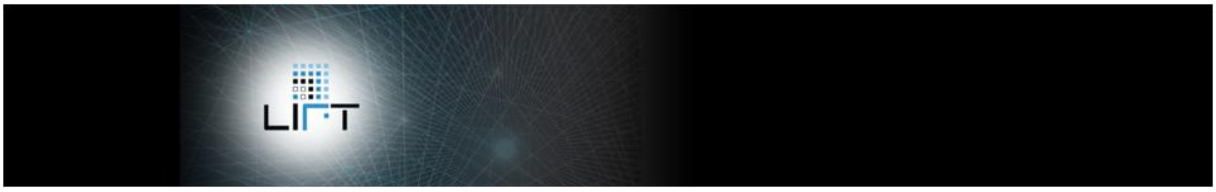


ICT, FET Open

LIFT ICT-FP7-255951

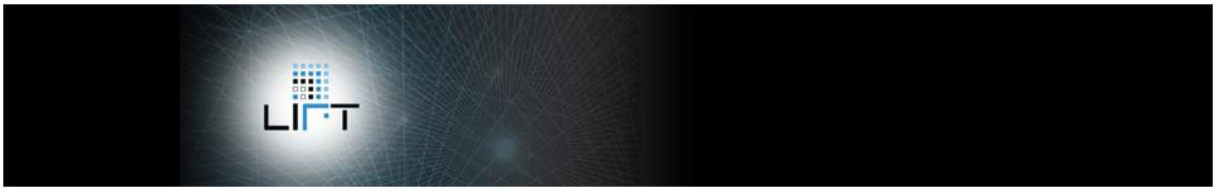
Using Local Inference in Massively Distributed Systems

Collaborative Project

D 4.2

Scenario Report

Contractual Date of Delivery:	31.07.2013
Actual Date of Delivery:	09.08.2013
Author(s):	C. Kopp (FHG), A. Deligiannakis (TUC), M. Gabel (Technion), M. Garofalakis (TUC), M. Kamp (FHG), D. Keren (HU), M. May (FHG), A. Monreale (CNR), M. Nanni (CNR)
Institution:	FHG
Workpackage:	WP 4
Security:	PU
Nature:	R
Total number of pages:	233



Project coordinator name: Michael May

Project coordinator organisation name:
Fraunhofer Institute for Intelligent Analysis
and Information Systems (IAIS)
Schloss Birlinghoven, 53754 Sankt Augustin, Germany
URL: http://www.iais.fraunhofer.de

Revision: 8

Abstract:

This document is the LIFT deliverable of WP4 for the third review period (01.10.2012 – 30.09.2013). The document provides a detailed report on the application results obtained during the project. It hereby points out the interconnection between the developed LIFT technologies (WP1) and their evaluation (WP5), taken privacy measures (WP2) as well as the used architecture (WP3).

Revision history

Administration Status		
Project acronym: LIFT		**ID:** ICT-FP7-255951
Document identifier:	D 4.2 Sceanrio Report (01.10.2012 – 30.09.2013)	
Leading Partner:	FHG	
Report version:	8	
Report preparation date:	26.07.2013	
Classification:	PU	
Nature:	REPORT	
Author(s) and contributors:	C. Kopp (FHG), A. Deligiannakis (TUC), M. Gabel (Technion), M. Garofalakis (TUC), M. Kamp (FHG), D. Keren (HU), M. May (FHG), A. Monreale (CNR), M. Nanni (CNR)	
Status:	-	Plan
	-	Draft
	-	Working
	-	Final
	x	Submitted

Copyright

Project funded by the European Community
under the Information and Communication
Technologies Programme
Contract ICT-FP7-255951

LIFT Deliverable 4.2
Scenario Report

C. Kopp, A. Deligiannakis, M. Gabel, M. Garofalakis, M. Kamp,
D. Keren, M. May, A. Monreale, M. Nanni

Contents

Chapter 1

Introduction

The objectives of WP4 are to design, implement, and evaluate the proof-of-concept applications of the LIFT approach to some key data processing and monitoring aspects of five high-profile real-world scenarios. Deliverable D4.2 provides a detailed report on the application results obtained during the project. It hereby points out the interconnection between the developed LIFT techniques (WP1) and their evaluation (WP5), taken privacy measures (WP2) as well as the used architecture (WP3). The deliverable thus connects the results of the other WPs and shows their applicability in a variety of real-world scenarios.

For the convenience of the reader, the report includes also the formalization, requirements analysis and data description of each scenario, which were already part of deliverable D4.1. The scenarios cover the following real-world applications:

1. Mobility Data Analysis
   (a) Distributed Monitoring of Driving Profiles
   (b) Monitoring Movement Behavior Using Stationary Scanners
   (c) Quantitative Monitoring of Visit Patterns
   (d) Distributed Estimation of Traffic Flows
2. Internet Scale Query Processing (ISQP)
3. Sensor Networks
4. Cloud Data Centers
5. Financial Monitoring

Note that we exchanged the former scenario 1a) "Distributed Density Map Computation" with a new scenario on the "Distributed Monitoring of Driving Profiles" (including a complete description of the new scenario). The reason for this exchange is twofold. First, the monitoring function of the former scenario turned out to be reducible to the linear case. The monitoring function of the new scenario is non-linear and part of a data mining task, namely the distributed monitoring of cluster quality. As the application of LIFT techniques to data mining algorithms was one main focus of year three, we decided in favor of the new scenario. Second, the new scenario offered the opportunity to explore a real-world problem formulated by the Italian Company *OctoTelematics*, which collects mobility data for insurance purposes.

Chapter 2

Mobility Data Analysis

2.1 Distributed Monitoring of Driving Profiles

2.1.1 Motivation

A key task in modern customer relationship management (CRM) is to understand the needs of each customer and to devise policies to harmonize them at the best with the company objectives. In most cases that translates into identifying a portfolio of *customer profiles*, each representing the needs and requirements of a reasonable number of customers. Then, each profile can be treated separately in order to devise the market strategies and business models that best fit its customers' peculiarities. In the business intelligence field, this process is best known as *customer segmentation*, and is traditionally implemented by applying predefined customer classification rules (for instance based on RFM indices such as recency of last contact with customer, frequency of transactions, monetary volume involved in the relation with the customer) or, in more recent times, by using clustering algorithms. This kind of process can be applied to any kind of business focused on services towards several customers, including the classical domain of retail selling as well as e-commerce and many others. In particular, in this section we describe an application of customer segmentation in the very actual and rather uncommon ground of car insurance business, where the main features that characterize a customer are related to how he/she drives, and therefore on his/her mobility. We adopt a data mining-oriented approach, working from a realistic *big data* perspective, where the application can benefit from a continuous flow of fresh information. In particular, we developed a framework where customer segments are built at the initialization step, and then they are continuously monitored to ensure that the available segmentation still fits well the customer behaviors. The definitions, methods and results we report here are a summary of Nanni et al. (submitted), where much more details can be found.

2.1.2 Scenario Description and Formalization

Scenario

We assume that each customer's vehicle is equipped with a localization device that can continuously update its position but also perform some computations. In particular, the device will take care of computing and updating the aggregate information about the vehicle/customer driving behaviors that will be used to build the customer segmentation. In principle, then, the device will continuously send the updated aggregates to the central site (the controller) where the segmentation is monitored and updated. As described later, the LIFT technology will play a role in reducing this systematic transmission of data into a lighter communication load.

Since we are interested in interpreting the driving behaviors of people, we model the characteristics of the customers through features extracted from their mobility. In particular, we analyze the trajectories that describe such mobility based on standard GPS traces, as those recently collected

by specialized companies for the car insurance sector.

For the description of the *user driving behavior* in a time window we identified four categories of measures: (i) *basic*, (ii) *space-time distribution*, (iii) *context-aware*, and (iv) *behavioral*.

- The *basic* category contains measures describing the simplest features of the trajectories in the time window such as the number of trips performed, their total distance and duration, and maximum acceleration and deceleration experienced.
- The category *space-time distribution* comprehends more complex measures that capture how the territory is used, both spatially and temporally: average distance of the user from his most frequent location, the radius of gyration of the user (i.e. the standard deviation from the center of mass of his movements), the radius of gyration w.r.t. to the user's most frequent location, the time spent by the user in the two most frequent locations (typically home and work place), and entropy measures related to the frequency distribution of places visited and time of travel (hour of the day).
- The third category is composed of the *context-aware* features, where information about the user's movement is related to the spatial and temporal context in which he moves: entropy of road segments traversed, distance traveled on highways, inside urban areas (cities) or on the most crowded roads (we selected the top 20% crowed roads), and distance travelled during night time.
- Finally, the *behavioral* category focuses on measuring the frequencies of some specific behaviors of the user, such as rapid accelerations/decelerations, exceeding the speed limits or driving along habitual routes (i.e., performing routine trips).

Following well-known precedents in the business intelligence literature and practice, we build customer segments through a clustering algorithm. The general requirements of our application can be met very simply by the standard K-means algorithm, which seeks globular clusters and has an almost linear complexity. In order to account outliers and yet keep the overall procedure efficient, we choose to first apply K-means on the input data, and afterwards apply a post-processing to remove objects too far from the cluster center. In particular, the distance threshold adopted in the experiments was computed separately for each cluster as $2 \cdot median_{x \in C_i}\{d(x, c_i)\}$, where C_i represents the cluster and c_i its centroid.

The mobility of individuals is a phenomenon that, in principle, can highly change with time. For instance, some routines might be affected by sudden and temporary variations of the environment (special events or road works forcing people to adapt their daily routes), or they might be influenced by seasonal factors (actually, the whole lifestyle might change from winter to summer). For these reasons, in our context it is advisable to have mechanisms that ensure a good fit between the actual segmentation (the one that the company is using to shape its business) and the real mobility behavior of the customers. When individual changes are small enough to have negligible effects on the corresponding segments, or at least not to impact on the overall structure of segments, it is advisable to simply keep the known segmentation, avoiding the practical troubles at the management level that would result from an excessively frequent redefinition of the segments. In order to implement these general principles, we define a monitoring process based on the safe zones approach continuously ensuring that the last computed segmentation still fits the data sufficiently well. When that does not happen, the existing segmentation is discarded, and a new one is computed from the most recent data.

Formalization

In terms of LIFT architecture, each customer/vehicle represents a node of the system, whose local vector contains the measures mentioned in the scenario description (plus an extra feature, which will be introduced later). Every node communicates its local vector to a central site that takes care of computing the initial customer segmentation and monitoring it. The central site plays the role of the coordinator of the system. In the following we introduce the global function to

be monitored (Formula 2.1), showing how the thresholds of the monitoring problem are decided (Formula 2.2), and how the problem itself can be treated by means of the LIFT tools (Formula 2.3).

In order to test the fit between existing segmentations and recent data we adopt a simple and very popular clustering quality measure called *Sum of Squared Error* (SSE in short), defined as follows:

$$SSE = \sum_{i=1}^{K} \sum_{p \in C_i} ||p - c_i||_2^2 \tag{2.1}$$

where C_i represents the i-th cluster, and c_i is its center (average vector). This measure evaluates the dispersion of each cluster around its centroid, and therefore gives emphasis to the compactness of clusters.

Our approach to deal with dynamic data consists in continuously checking that the dispersion of the objects within the clusters did not grow, or at least not significantly. That means computing the SSE at each time stamp t, which we will denote with SSE_t, and test that it stays below some threshold. That is summarized in the following problem definition:

Definition 2.1.1 (Cluster Monitoring Problem)**.** *Given a clustering $C = \{C_1, \ldots, C_K\}$ having initial SSE equal to SSE_0, and given a tolerance $\alpha \in \mathcal{R}^+$, we require to ensure that at each time instant t the following holds for the SSE of the (dynamic) dataset D_t:*

$$SSE_t \leq (1 + \alpha) SSE_0 \tag{2.2}$$

When that does not happen, an alarm is raised.

We should note that SSE describes all the clusters together, aggregating the dispersions of the single clusters. That means that in principle having a good SSE does not guarantee that each single cluster is compact, since some slightly over-dispersed cluster might be balanced in the sum by some virtuous one that adds very little to the SSE. From this perspective, it might happen that the end user of our application wants to ask for stronger requirements in the monitoring problem. Therefore, in our work we considered also variants of the problem where the constraints are imposed over each single cluster. Please refer to Nanni et al. (submitted) for details.

The cluster monitoring problem can be fitted to the geometric (safe zones) approach by properly rewriting it through a vector augmentation trick. Indeed, the formulation of SSE is very similar to a variance, though on d dimensions, which is a classical example of complex function that can be reduced to the framework of safe zones. To find the precise relation between the two, we observe that each cluster C_i, having centroid c_i, contributes to the SSE by the following value:

$$\begin{array}{ll} SSE^{(i)} & = \sum_{p \in C_i} ||p - c_i||_2^2 \quad = \quad |C_i| \sum_{j=1}^{d} var_{p \in C_i}(p^j) \\ & = |C_i| \cdot \left[avg_{p \in C_i} \left(||p||_2^2 \right) - ||avg_{p \in C_i}(p)||_2^2 \right] \end{array} \tag{2.3}$$

where p^j represents the j-th component of the d-dimensional vector p. This means that by augmenting the vector $v_i(t)$ of each node with the additional feature $||p||_2^2$ we can compute the variance for each component, as requested in (2.3).

Monitoring framework

The work in Nanni et al. (submitted) implements a monitoring framework based on the problem reformulation shown above, by partitioning the problem into K subproblems (one per cluster). Basically, the single global threshold for SSE_t in (2.2) is properly partitioned into K components, i.e. one threshold for the $SSE^{(i)}$ of each cluster.

Also, the framework is organized along three levels. At the lowest level, each node N_i monitors whether its local vector stays within the associated safe zone. Whenever a local violation occurs, the node communicates the new local vector to the controller, which tries to solve the violation *locally to the cluster*, i.e., applies a balancing procedure involving only (a subset of) the nodes

that belong to the same cluster as the violating node. This represents the middle- or cluster-level of the framework. When the balancing within the cluster fails (i.e., $SSE^{(i)}$ is really larger than its associated threshold), a global, inter-cluster balancing is operated, trying to see whether the threshold of a *virtuous* cluster can be slightly eroded in favor of the violating cluster. This represents the higher level of the framework. When everything fails, a complete synchronization of the system is operated, together with the re-computation of cluster assignments. The latter step is performed by executing the k-means algorithm on the recent data, followed by the outlier removal procedure previously described.

Since human mobility is known to have a substantial systematic component of behaviors that tend to repeat with some periodicity, our framework included a learning process that tries to identify possible periodic behaviors for each customer/node and exploit it in the monitoring task. In particular, we adopted the recent predictive model-based monitoring approach developed within the LIFT consortium (Giatrakos et al., 2012), extending it with a history-based model built on top of the customer's historical data values.

2.1.3 Requirements Analysis

The foremost requirements for the present application are about the communication load that the network might experience, due to the large number of customers to be monitored: the case study we will summarize in Section 2.1.4 already contains more than 10k users, and the data provider is actually collecting data for more than 900k customers nation-wide, with a growing trend.

A second requirement is the privacy of the individuals monitored. While the information sent in the process by each customer is of an aggregated nature, some privacy concerns still remain, due, for instance, to the possibility of identifying some extreme behaviors that might be linkable to single users.

2.1.4 Employed datasets

The application introduced above has been implemented on top of our monitoring framework and tested against a dataset of real GPS traces. The dataset is provided by an Italian company called *OctoTelematics* collecting data for insurance purposes. This dataset is composed of GPS observations of 11,470 private cars active in Tuscany in a period of 35 days between June and July 2011. Due to some pre-processing (i.e. aggregation and filtering) performed by the device on board, the sampling rate is reduced to an observation every 3 minutes and it is not regulated by any policy of synchronization.

We divided the dataset temporally in order to create a training and a test set, respectively using the first week and the remaining 4 weeks. The two datasets have been processed to extract the measures presented in Section 2.1.2 using a time window of 3 days with a time granularity of 15 minutes. Unfortunately, due to the specific data format adopted by the data provider and to the low sampling rate, some of the measures we discussed cannot be extracted. In particular, all the acceleration-based measures have been excluded from our experiments for this reason.

2.1.5 Results

Communication reduction

In this section we report a summary of the main results obtained in our case study. Further details, including a deep exploration of the customer segments obtained and a performance evaluation on synthetic data, can be found in Nanni et al. (submitted).

In Figure 2.1 our monitoring framework is evaluated varying the α parameter (percentage tolerance of SSE). The algorithm is applied with different variants presented in Nanni et al. (submitted). The variants considered are: apply a balancing with memory (**M**), which computes vector translations for a subset of nodes, such that all the safe zone violations encountered are removed, and such translations are also materially applied to vectors, thus they will have an effect

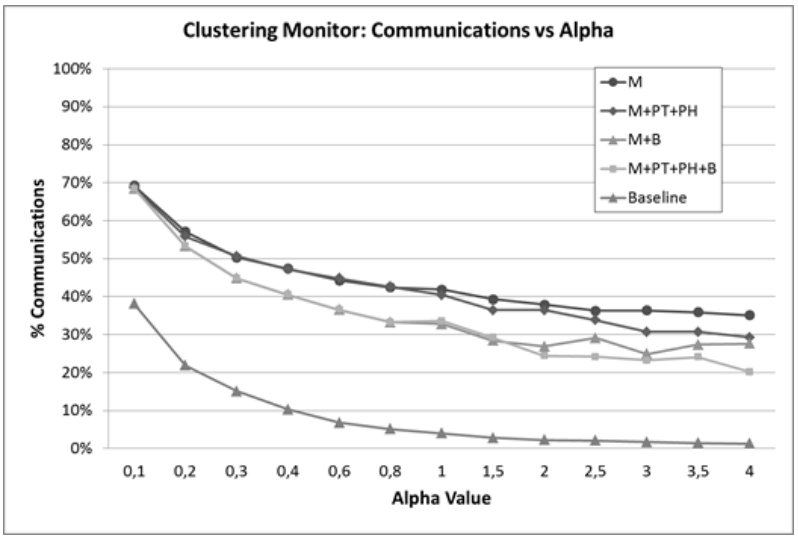


Figure 2.1: Communications during the clustering monitor by varying the α parameter.

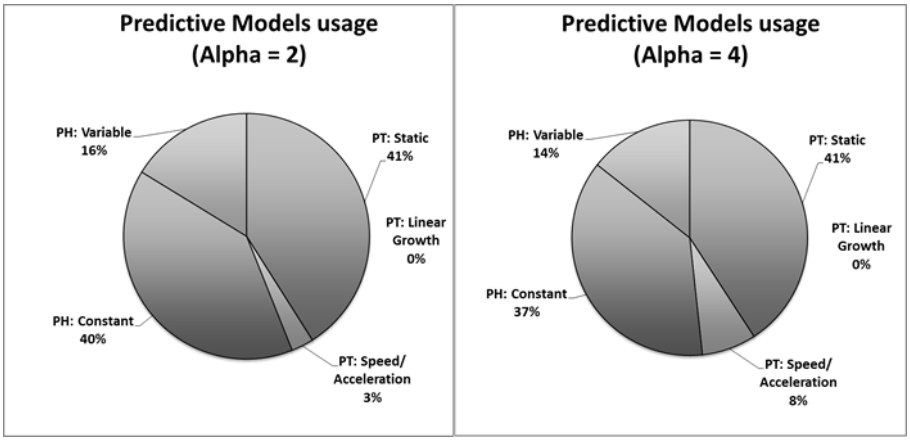


Figure 2.2: The percentage predictive model usage.

on the next iterations; some variants use the standard trend predictive models (**PT**) introduced in Giatrakos et al. (2012) and the novel history predictive models introduced in this work (**PH**); some variants consider a buffer in the balancing (**B**), i.e. they translate the node vectors in such a way to maximize the minimum distance from the border of the safe zone. We remark that the balancing is also operated at the level of clusters (the upper one of our three-level monitoring framework), and applying a buffer here basically means dynamically changing the SSE thresholds of the single clusters. Moreover, in the figure we report also the baseline representing a lower bound of the communications needed for monitoring, computed as the communications generated only by synchronizations.

Generally, the framework proves to be able to reduce the communication load substantially, cutting between 50% and 80% of messages for any reasonable value of α – we remark that the values $\alpha \in \{0.1, 0.2\}$ resulted (through empirical evaluation/exploration of the data) to be in practice very tight constraints, and are reported here just for reference. Typical *moderate* variations of SSE should be expected to be in the range $\alpha \in [0.6, 2.5]$. However, we can observe that the different variants of our framework yield significantly different performances. In particular, using a buffer in the node balancing procedure almost always reduces communications by ca. 10%. Then, the predictive models also improve performances, with communication reductions that increase with

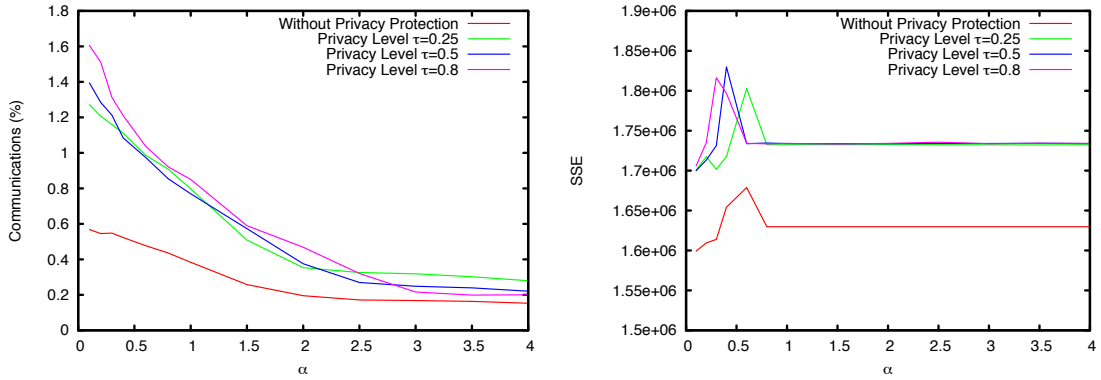


Figure 2.3: (left) Communications evaluation by varying α and for different levels of privacy protection. (right) SSE by varying α and for different levels of privacy protection.

larger error tolerance (α) values.

Figure 2.2 reports the usage of each predictive model, analyzed in terms of the amount of time in which it is used during the monitoring. **PH: constant** is a simple variant of the history-based predictive models that predicts a fixed, time-independent vector value (basically, it is the typical value assumed by the node vector during the learning period), whereas **PH: variable** is the basic variant whose predicted value depends on the time of the day (more precisely, the time granularity applied into these experiments are slots of 15 minutes). The family of trend-based models includes the **PT: static** one, corresponding to simply use the last communicated value (equivalent to have no predictive model at all); **PT: linear growth** and **PT: speed/acceleration**, that assume, respectively, that the vector values follow a linear behavior or that they maintain the same speed and acceleration observed in the recent updates. It is important to notice how, excluding the *PT: static* one, the constant history-based model is the most used outperforming the others. The second most used is the variable version, which is probably more affected by noisy values than the constant version. The least used one, is the linear growth model, which is never used and, as expected, proves to be not suitable to describe the trend of our data. Overall, we observe that the history-based predictive models have a much larger impact on the monitoring than the trend-based ones proposed in Giatrakos et al. (2012).

Privacy aspects

In Monreale et al. (submittedb) we reformulated our monitoring problem taking into consideration the privacy requirements, and we validated our approach with an extensive set of experiments on large and real data. In particular, we verified the empirical privacy guarantees, the impact of the data transformation on the number of communications and on the correctness of the monitoring function. We compared the amount of communications required by the monitoring process without any privacy guarantee and the number of communications required in the system when we use our privacy preserving method with different levels of privacy.

The approach followed to add a privacy protection consists in perturbing the original data vectors. This perturbation, quite naturally, introduces a level of noise in the data, making them less stable than the original ones, potentially making the communication reduction strategies describe in the previous sections less effective.

As expected, the number of communications obtained in experiments increases with the privacy protection level required. Figure 2.3 (left) shows that the privacy transformation adds a variable amount of communications to the original one, that amounts to around 30% for the reference case of $\alpha = 1.0$. We also analyzed the impact of the randomization on the quality of the SSE. The plot in Figure 2.3 (right) shows the behavior of the measure for different values of privacy protection. We can observe that the SSE value increases when the level of privacy in the data is

greater; however, the effect of the privacy enforcement is reasonable because we have an increasing of about 7% of the original value in the worst case. Finally, we also evaluated the level of privacy guaranteed at the coordinator site. We simulated an attacker by spectral filtering technique and we found that the level of privacy provided is much higher w.r.t. the theoretical level of privacy set as a parameter. This is a good result considering that the performance of the system and the correctness of the global function can be considered acceptable even with this high data distortion. A detailed description of the results can be found in Monreale et al. (submittedb).

2.1.6 Discussion

This application represents an innovative instance of the customer segmentation problem in the context of car insurances, characterized by a strong human mobility component and the large scale, distributed and streaming nature of its data sources. We developed and tested all the components of the application, exploiting a large dataset of car mobility data, showing the feasibility of the approach.

The LIFT technology proved to be very effective in reducing the communication load, also thanks to the flexibility of the safe zones approach, that allowed to incorporate rather easily domain-specific improvements, such as the multi-level partitioning of the problem (which made the standard node balancing procedures very effective) and the detection of periodicities in the customers' behaviors (exploited through predictive models). Experimental validation showed that not all improvements to the basic monitoring framework were actually beneficial. Some of them, such as the *PT: linear growth* predictive model, proved to be simply useless in our case. Others, for instance the use of buffers at the level of nodes, were actually detrimental – as opposed to buffers at the cluster level, which yielded large improvements. Finally, as Figure 2.1 shows, there is still some margin of improvement for the solution developed so far (see the gap between the lower bound (Baseline) and the other curves). The first next step required to fill such a gap will be a deeper understanding of the domain-specific factors that cause these (apparently) unnecessary communications.

We believe that the basic building block of the framework, i.e. the distributed monitoring of cluster quality, is a data mining problem with an applicability that goes beyond the application domain considered here, and we plan to validate it into an extensive array of clustering problems, in order to achieve a general methodology.

Finally, the privacy-aware variants of the problem we studied so far, though not covering the full framework, yielded empirical results that are very promising, indicating the possibility of finding a good trade-off between efficiency/quality of the application and privacy guarantees. Our future activities on this line include the extension of the approach to cover the full cluster monitoring framework developed for this application, as well as its generalization to other application domains that share a similar structure.

2.2 Monitoring Movement Behavior Using Stationary Scanners

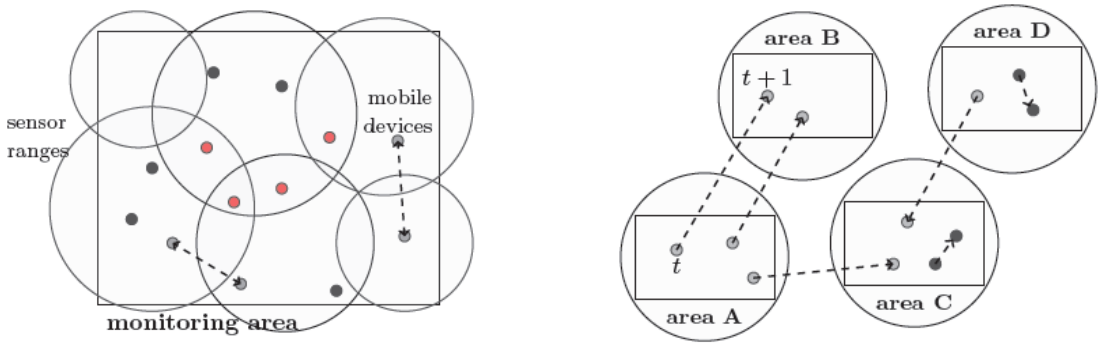
2.2.1 Motivation

With advancing sensor technology and spread of mobile devices, mobility modeling and monitoring is becoming a topic of high interest. Depending on the spatial extension of the monitored area, different parties are interested in mobility statistics. For example, organizers of local events such as open air concerts, soccer games or theme parks can apply monitoring to detect security problems (Is there an overcrowded area?) or to improve their services (Should the road between two attractions be enlarged?). On a global level, the statistics about the traffic within a city or between different communities are necessary for the management of the traffic network and reduction of traffic jams.

Using stationary sensors for mobility monitoring has the advantage that people do not have to be equipped with a separate tracking device. Examples of such stationary sensors are Bluetooth or GSM antennas. If the base technology is widely spread within the population, a high coverage can be reached. However, privacy protection is a great issue when using such passive monitoring techniques. First, people are typically not aware of passive monitoring technologies, and the barrier for an active opt out (e.g. switching off Bluetooth functionality) is therefore high or even impossible (e.g. in the case of cell phone towers). Second, mobility data is especially sensitive when making inference from the data because spatial locations are inherently connected to personal activities. Thus, it is especially important to apply privacy preserving techniques when evaluating mobility data from stationary scanners.

In this scenario we consider two major tasks from mobility mining, crowd monitoring and flow monitoring. The first scenario, crowd monitoring, is relevant for preventing overcrowding at major public events such as concerts, sport events or demonstrations. The goal of the application is to continuously monitor the amount of people in certain areas. A well-studied approach to monitoring the amount of people in an area is to use sensors that count the number of *mobile devices* in their sensor radius (Gibbons (2009), Gonçalves et al. (2011)). The actual amount of people can then be estimated assuming a stable average fraction of people owning such a device (Liebig et al., 2012). The sensors are distributed over the area so that the sensor ranges cover the whole space that is to be monitored (see Figure 2.4(a)). The sensors then send their data to a central coordinator that evaluates the data and presents it to the user. The second scenario, flow monitoring, estimates the number of people that move between distinct destinations. For a given set of locations, all flows between those locations can be combined in the form of an origin-destination matrix. These matrices are an important tool in traffic management (Ashok and Ben-Akiva, 1993).

In order to apply stationary sensors for flow monitoring (Barceló et al., 2010), the sensors are placed at each location of interest (see Figure 2.4(b)). For a given time interval, the flow between two locations denotes the amount of mobile entities that have been present at one location at the beginning of the time interval and present at other location at the end of the interval. An entity is present at a location, if it is staying within the range of the sensor placed at that location. In the scenario we assume that the sensor ranges do not overlap.



(a) (b)

Figure 2.4: The two mobility modeling tasks addressed in this paper: (a) *crowd monitoring* having the goal to detect the number of distinct devices in the sensor range with devices potentially in the range of multiple sensors (red circles) or moving from one sensor range to another during one time interval (gray circle); and (b) *flow monitoring*.

2.2.2 Formalization

During its life-time a sensor S observes a large number of mobile devices, each of which possesses a unique identifier from some global address space A. We denote the stream of **sensor readings**

of S by $\mathcal{R}_S \subseteq A \times \mathbb{R}_+$ where $(a,t) \in \mathcal{R}_S$ means that S has read a device address a at time stamp t. For some **measurement interval** $T = [b,e] \subseteq \mathbb{R}_+$ we denote by $\mathcal{R}_S^T = \{(a,t) \in \mathcal{R}_S \mid t \in T\}$. We represent the distinct devices observed by the sensor in some given measurement interval without explicitly storing their addresses. This representation is achieved by mapping the devices to a vector $f_S^T \in \mathbb{R}^{|A|}$. For a mapping $h : A \to \{1, ..., |A|\}$, the vector is defined as

$$f_S^T[i] = \begin{cases} 1, & \exists (a,t) \in \mathcal{R}_S^T : h(a) = i \\ 0, & otherwise \end{cases}$$

The number of mobile entities in an area covered by a set of sensors $\mathcal{S}$ is equal to the distinct number of MAC-addresses present at all sectors. We define a global vector as

$$f_{\mathcal{S}} = \bigvee_{S \in \mathcal{S}} f_S \ .$$

Now, the number of distinct mobile entities is

$$n_{\mathcal{S}} = \|f_{\mathcal{S}}\|_1 \ .$$

For crowd monitoring, the number of mobile entities in an area must not exceed a certain threshold ϵ. Thus, the non-linear function that is monitored in this scenario is

$$\|f_{\mathcal{S}}\|_1 \leq \epsilon \Leftrightarrow \left\| \bigvee_{S \in \mathcal{S}} f_S \right\|_1 \leq \epsilon \ .$$

Given two sensors $S, S' \in \mathcal{S}$ and two disjoint time intervals $T = [b,e], T' = [b',e'], b' > e$, the flow from S to S' in between those time intervals is

$$\begin{aligned} n_{S \Rightarrow S'} &= \left| \{\forall a : (a,t) \in \mathcal{R}_S^T \wedge (a,t') \in \mathcal{R}_{S'}^{T'}\} \right| \\ &= \left\| f_S^T \wedge f_{S'}^{T'} \right\|_1 \ . \end{aligned}$$

This result can be generalized to paths. For a given set of sensors $S_1, S_2, ..., S_k \in \mathcal{S}$ and a set of consecutive, disjoint time intervals $T_1, T_2, ..., T_k$, we define a path $\mathbf{P} = ((S_1, T_1), ..., (S_k, T_k))$. The amount of devices that have been present on this path is

$$n_{\mathbf{P}} = \left\| \bigvee_{(S,T) \in \mathbf{P}} f_S^T \right\|_1 \ .$$

For flow monitoring, the flow between two sensors S, S' should not exceed a threshold ϵ. Hence, the non-linear function that is monitored is

$$\left\| f_S^T \wedge f_{S'}^{T'} \right\|_1 \leq \epsilon \ .$$

The presented approaches to crowd and flow monitoring with stationary sensors rely on tracing unique identifiers through the sensor network. Hence, identifying and tracking a specific device, i.e., a specific person, is possible in monitoring systems as soon as the identifier can be linked to a person. This is in most cases a severe privacy violation. Thus, we propose using linear count sketches at each local sensor and combine them in a privacy-preserving manner to estimate crowds and flows. This way, the privacy-preserving properties of linear count sketches at each sensor are maintained when combining them and identifying and tracking of a single entity with absolute certainty becomes impossible.

2.2.3 Requirements Analysis

The application has to fulfill requirements with respect to privacy, accuracy and to a smaller extend to scalability and communication reduction. The main requirement for this application scenario is privacy.

Our approach is to use sketching in order to trade accuracy for privacy. In a sketch, many mobile entities are subsumed to a single bit of information from which no deterministic inference on a single mobile entity can be accomplished. The more entities are subsumed to one bit in the sketch, the more privacy can be guaranteed. In particular, we want to ensure that an attacker cannot deterministically infer upon the movement or position of a mobile entity, even if he has access to the sketches. Furthermore, we want to provide a bound on the amount of certainty an attacker can gain about a particular entity, following the notion of k-anonymity on estimate (Aggarwal, 2008).

For common privacy requirements (e.g., k-anonymity for $k = 5$) we want our approach to meet the accuracy requirements of its practical applications. Furthermore, the application is required to provide accuracy guarantees given certain privacy requirements.

Our approach should be scalable to all practically useful dimensions. Thus, the application should be able to monitor several hundred thousands to several million users using a handful to hundreds of sensors.

Finally, our method should provide means of communication reduction based on the Safe Zone approach in order to prolong sensor battery life and reduce communication costs.

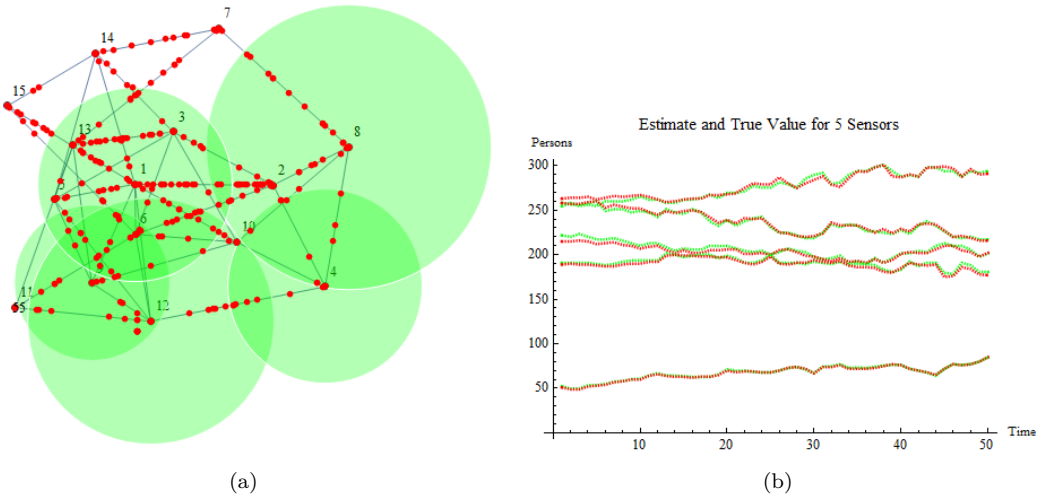
2.2.4 Employed Data Sets

For the evaluation of our method we use simulated movement data of up to 100.000 entities. The entities move on a path network and are monitored by simulated stationary sensor devices. In our simulation, we assume each entity to be tractable, as well as perfect sensor readings, i.e., no noise, each entity is directly captured and the sensor scans continuously. We implemented a simulation environment as follows. A random graph of k nodes with Bernoulli Graph Distribution ($p = 0.4$) is generated; the position of nodes in 2D-space is calculated using an automatic graph layout method. A number s of node locations is attached with sensors with a predefined range. In general, a sensor may cover more than one node and several edges. A number of n objects, i.e. the global population, is created. For each object a random sequence of tour stops (nodes of the graph) is generated. For every pair of tour stops the shortest path is determined using Dijkstra's method and inserted into the sequence between the stops. Finally, to each object a velocity, starting time and a step size is assigned (the latter because objects are not only at the node positions, but travel along the edges). During the simulation, for each time step the objects follow the tour with the assigned velocity and starting time, and their position along the edges is calculated. Each sensor monitors at each time step the objects in its sensor range. For each sensor and time period a new sketch is calculated and stored. As a ground truth, also the object address are stored. The simulation stops when the last object has completed its tour.

In addition, we plan to collect real-world Bluetooth data for the evaluation of this scenario.

2.2.5 Results

On the simulated data we conducted extensive experiments (Kamp et al., 2013) in which we have shown that our approach meets the requirements. For the crowd monitoring scenario, overlapping sensors are simulated. Fig. 2.5(a) shows a snapshot from a simulation run. For the flow monitoring we use non-overlapping areas. The distribution of mobile entities in the sensor ranges is generated by a process very similar to real traffic flow, so that we have realistic flow properties over time. Fig. 2.5(b) shows an example for crowd monitoring with 15 nodes, 5 sensors, 1000 objects and a load factor of 5. Evidently, the estimates closely track the true values, as expected from the theoretical analysis. For the flow monitoring scenario, Figure 2.6(a) shows the estimate (green, dashed) and true (red, dotted) value of the flows between 4 sensors over time. Next we simulated



(a) (b)

Figure 2.5: (a)Crowd simulation with Random Bernoulli Graph, 1000 persons (red dots), 5 Bluetooth sensors with overlapping range. (b) Estimated value (green dashed) and true value (red dotted) for each sensor for the first 50 time steps.

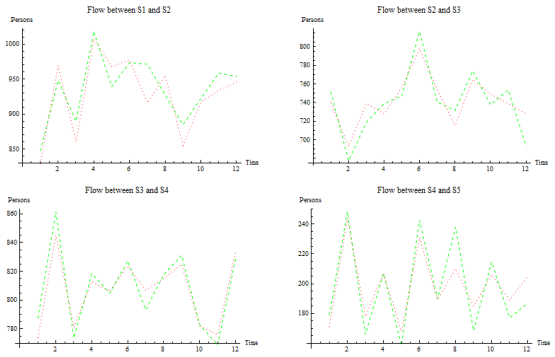
the OD-matrix construction problem. Table 2.6(b) shows the relative error of an OD-matrix construction for 20,000 persons moving over a period of 15 time steps in a system with 12 nodes, tracked by 5 sensors. Each sensor uses the improved estimator with 30 sketches and a load factor of 5. From these results we conclude that even for moderately sized sets, the error is very low. Overall, we conclude from the experiments that Linear Counting behaves in the complex setting of the simulation as expected from a theoretical point of view and is a very promising approach for deployment in real-world applications. The approach guarantees k-anonymity on expectation, i.e., an attacker cannot infer deterministically on a single entity but at most on k entities as a group (Kamp et al., 2013). The approach is highly accurate for common privacy requirements, i.e., $k = 5$ and scales well with the number of mobile entities so that monitoring even very large areas, e.g., entire cities, is possible. Sensor battery life is prolonged by sending only sketches of data instead of full sensor readings, thereby reducing communication.

2.2.6 Discussion

We successfully applied sketches to increase privacy in mobility monitoring scenarios with stationary sensors and have been able to improve existing sketching techniques to meet the application requirements. More specifically, we extended the Linear Counting approach to a general set of primitives for privacy-preserving mobility modeling. We showed theoretically and empirically that two challenging application scenarios can be solved using and combining this set of primitives (Kamp et al., 2013). To compensate the accuracy deficit of Linear Counting for strict privacy requirements we presented a method for increasing the accuracy, while maintaining the privacy. This method is also applicable to boost accuracy in flow estimation, allowing to monitor even tracks.

In contrast to many privacy-preserving approaches, this one is easy to implement, has excellent accuracy and can be implemented efficiently. Our experiments suggest that it is immensely useful in a practical settings and can have a real impact on how stationary sensor based data collection is done.

The developed techniques have a high potential of being applied in real-world applications, soon. However, it remains to reduce communication using safe-zones. The main difficulty we face is to express the binary OR function as a function of the average of local binary vectors. We found that, while transforming continuous, even highly non-linear functions to functions of



(a)

	S1	S2	S3	S4	S5
S1	0.005	0.008	0.007	0.018	0.001
S2	0.026	0.026	0.01	0.006	0.015
S3	0.004	0.003	0.016	0.002	0.014
S4	0.02	0.03	0.011	0.004	0.012
S5	0.014	0.005	0.004	0.017	0.027

(b)

Figure 2.6: Flow estimation (green) and true values (red) between four sensors (a) as well as relative error of an OD-matrix between five sensors (b). Results are from flow simulation with 20000 persons and 5 sensors.

the average of local values is feasible using the techniques and tools developed in this project, transforming discontinuous functions remains difficult. However, solving this problem yields not only benefits for this application scenario but will further improve the generality and applicability of the LIFT-approach.

2.3 Quantitative Monitoring of Visit Patterns

2.3.1 Motivation

The organization and planning of services (e.g. shopping facilities, movie theaters) requires up-to-date knowledge about visiting customers. Especially quantitative informations about the number of customers and their frequency of using the offered service or a combination of services are important. The goal of this application scenario is to provide such quantitative information in a privacy preserving way. We hereby assume that each service is connected with a specific location and that the usage of a service can be identified from the recorded movements of a person by detecting visits to that location, hence the name customer-location interaction. For example, assume that the owner of a supermarket chain with five supermarkets in Cologne, Germany, is interested in statistics about his customers. On a monthly basis he wants to know the total number of customer visits, the average number of visits per customer and the number of distinct customers that visit any of his stores. All three quantities can be derived if the movement trajectories of the population (or in the example of people living in Cologne and its surroundings) are known. The example also shows that quantitative information cannot only be provided for a single location, but also for alternative location choices. In this case, visits to any of the five supermarkets are of interest. Furthermore, sequences of visits (e.g. going to the cinema and subsequently to a pub), possibly refined by time constraints, can be monitored. We will refer to such composite customer-location interactions as *visit patterns*.

Clearly, the most important requirement for the scenario is the protection of personal movement information. From a privacy point of view it is out of question to collect the movement trajectories of the population at a central server. Therefore, we will evaluate all trajectories locally (at the personal devices) in our approach. Only aggregated statistics of visits or visit patterns will be communicated to the central server. In addition, communication reduction is important because the application shall scale to geographic regions as cities, federal states or even the whole nation. For Germany with about 82 million inhabitants this means to handle between several tens of thousands and tens of millions of local nodes. Local data processing has advantages for both requirements: it allows to encapsulate sensitive mobility data at the nodes and to reduce

communication.

2.3.2 Scenario Description and Formalization

Scenario Description

In our scenario each person carries a mobile device which determines the person's position in regular time intervals. We hereby assume that each person is represented by exactly one device. The device possesses processing power to analyze the position records. In particular it is given a point of interest (POI) database which it utilizes to detect location visits from the stream of position records. In addition, the device receives a set of visit patterns. For each pattern it monitors the number of occurrences over periodic time intervals (e.g. one week or one month). At the end of each time interval the device communicates the number of occurrences of each pattern to the coordinator. The coordinator aggregates the counts in the distribution of k-visiting entities for each pattern, which states how many persons had 0, 1, 2, ... pattern occurrences in the given time interval. The coordinator monitors this distribution as well as further quantities (gross visits, average visits, entity coverage) derived from it and sends an alert if a change in the distribution or in any derived quantity exceeds a given threshold value. Figure 2.7 shows the information flow and communication architecture of the application. Between the local nodes and the coordinator an intermediate layer for the anonymization of source identifiers is inserted. The anonymizer follows e.g. the principle of onion routing (Goldschlag et al., 1999) and prevents that the coordinator reconstructs movement histories from several messages of a person based on the communication protocol. The anonymizer itself is prevented from reading message contents by encryption technology. Further details on privacy aspects of the distributed monitoring scenario can be found in (Kopp et al., 2012).

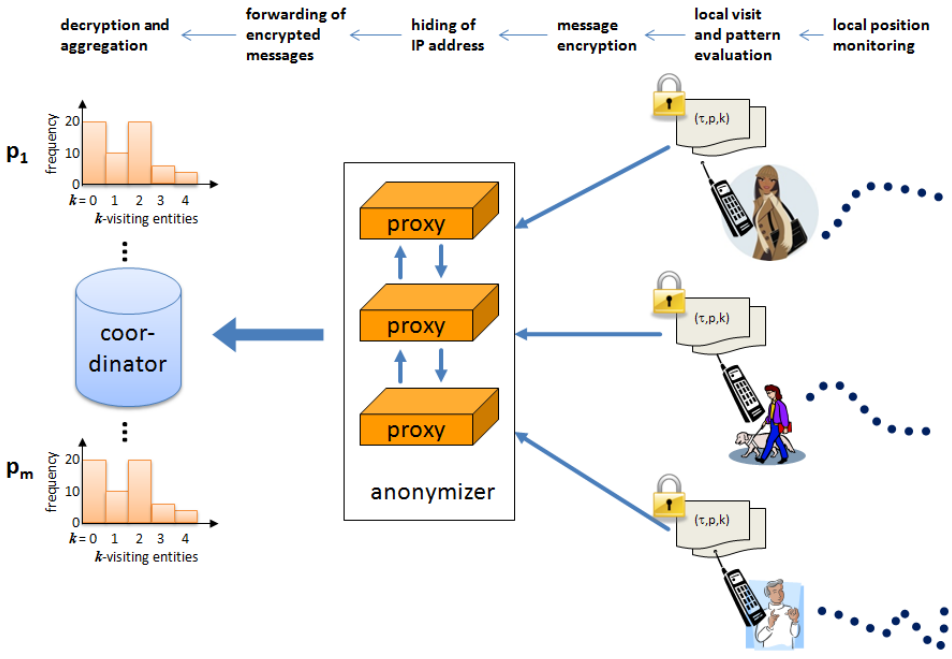


Figure 2.7: Communication architecture for monitoring customer-location interactions

Formalization

Our local nodes are mobile devices which each represent one person. The coordinator is a central database which maintains the distribution of k-visiting entities for each visit pattern. The visit

patterns are monitored independently of each other, we therefore confine the following formalization to a *single* visit pattern.

Given a set of local nodes $i = 1..n$, a visit pattern and a time interval $\tau = (t_{start}, t_{end}]$, let $x_0, x_1, ..., x_j, ..., x_m$ with $j = 0..m-1$ denote frequency counts specifying how many local nodes observed j occurrences of the visit pattern during τ.

The global data vector contains the frequency distribution of k-visiting entities and is defined as

$$\vec{v}(\tau) = (x_0, x_1, ..., x_j, ..., x_{m-1})^T.$$

The global data vector is the sum of the n local data vectors

$$\vec{v}(\tau) = \sum_{i=1}^{n} \vec{v_i}(\tau) = \sum_{i=1}^{n} (x_{i0}, x_{i1}, ..., x_{i(m-1)})^T.$$

Each local data vector states the frequency of the visit pattern at the local node. As each local node belongs to one frequency class, exactly one of the frequencies x_{ij}, $j = 0..m-1$ in each local data vector has a value of one while all other frequencies are zero, i.e. let node i follow the visit pattern k times, then

$$x_{ij} = \begin{cases} 1 & \text{if } j = k \\ 0 & \text{else.} \end{cases}$$

On the global data vector we want to monitor four different quantities. First, we want to monitor the difference between the current global statistics vector and a given distribution (e.g. from previous observation). Second, we want to monitor three quantities that can be derived from the global data vector, namely gross visits, average visits and entity coverage as previously defined in (Körner, 2012; Körner et al., 2010).

In order to monitor the difference between two distributions, we can use the Kolmogorov–Smirnov test or Chi-Square goodness of fit test. The Kolmogorov–Smirnov test relies on the greatest difference between the cumulative distribution functions (*cdf*) of two distributions. Let $F_0(x)$ denote the *cdf* of some base distribution and let $F_1(x)$ denote the *cdf* of the global data vector. The Kolmogorov-Smirnov test statistic D is defined as

$$D = \max_{x \in \mathbb{R}} |F_1(x) - F_0(x)|.$$

The test statistic is compared to the critical value $D_\alpha = K_\alpha \sqrt{(n_0 + n_1)/n_0 n_1}$ where K_α depends on the level of significance (e.g. $K_\alpha = 1.36$ for $\alpha = 0.05$) and n_0 and n_1 denote the number of items in the empirical distributions. Applied to our scenario, we can monitor the Kolmogorov-Smirnov test statistic as follows. Let $\vec{v}^* = (y_0, y_1, ..., y_j, ..., y_{m-1})^T$ denote the base distribution to which the global data vector shall be compared.The monitored function then is

$$g_1(\vec{v}(\tau)) = ||\vec{v}(\tau) - \vec{v}^*||_\infty = \max_{i=0..m-1} \left| \sum_{k=0}^{i} x_k - \sum_{k=0}^{i} y_k \right|. \tag{2.4}$$

If the function value of the global data vector is smaller or equal to D_α both distributions are equal. The Chi-Square goodness of fit test compares distributions based on the sum of squared (relative) distances between frequency classes. Its test statistic leads to the following monitoring function

$$g_2(\vec{v}(\tau)) = \sum_{i=0}^{m-1} \frac{(x_i - y_i)^2}{y_i}. \tag{2.5}$$

The threshold value χ^2_α is given by the Chi-Square distribution using $m-1$ degrees of freedom at an appropriate level of significance. If the function values is smaller or equal to χ^2_α, the global statistics vector has the same distribution as the given base distribution. Clearly, both the Kolmogorov-Smirnov and the Chi-Square goodness of fit test statistics are non-linear functions.

In addition to the distribution, we want to monitor the quantities gross visits, average visits and entity coverage as defined in (Körner, 2012; Körner et al., 2010). The authors have shown that all three values can be derived from the distribution of k-visiting entities, which is the global data vector in the scenario. Note that the original definitions are based on *visits*, however, the notion of a visit can interchangeably be used with *visit pattern*. The three quantities can be monitored for different visit classes $vc \in \mathbb{Z}$, which restrict the quantities to nodes with a minimum frequency count. For example, the number of average visits per entity for visit class $vc = 2$ states the average number of pattern occurrences for persons who followed the pattern at least twice.

Before we proceed to the definition of the monitoring functions, we introduce the following vector $\vec{c}_{vc}$ of length m

$$\vec{c}_{vc} = (c_0, ..., c_i, ..., c_{m-1})^T \quad \text{with} \quad \begin{cases} 1 & \text{if } i \geq vc \\ 0 & \text{else.} \end{cases}$$

For example, for $vc = 2$ the vector has the form $\vec{c}_{vc} = (0, 0, 1, ..., 1)$. Further we will use the diagonal matrix $A = diag(0, 1, 2, ..., m-1)$ as well as the identity matrix I of size m.

Gross visits state the total number of visit patterns that occur in the given time interval. Given a visit class vc, the gross visit monitoring function is defined as the following linear function:

$$\begin{aligned} g_3(\vec{v}(\tau), vc) &= \sum_{i \geq vc}^{m-1} i \cdot x_i \\ &= \vec{c}_{vc}^{\,T} \cdot A \cdot \vec{v}(\tau). \end{aligned} \tag{2.6}$$

Average visits state the average number of pattern occurrences per node. For visit class $vc = 0$ the monitoring function is again linear because the divisor is a constant stating the total number of nodes. For visit classes $vc > 0$ this number can change over time, and the function becomes non-linear. Given a visit class vc, the average visit monitoring function is defined as

$$\begin{aligned} g_4(\vec{v}(\tau), vc) &= \frac{\sum_{i \geq vc}^{m-1} i \cdot x_i}{\sum_{i \geq vc}^{m-1} x_i} \\ &= \frac{\vec{c}_{vc}^{\,T} \cdot A \cdot \vec{v}(\tau)}{\vec{c}_{vc}^{\,T} \cdot E \cdot \vec{v}(\tau)} \ . \end{aligned} \tag{2.7}$$

Finally, entity coverage states the proportion of the nodes which observe at least vc occurrences of the given visit pattern. Given a visit class vc, the entity coverage monitoring function is defined as the following linear function:

$$\begin{aligned} g_5(\vec{v}(\tau), vc) &= \frac{\sum_{i \geq vc}^{m-1} x_i}{\sum_{i=0}^{m-1} x_i} \\ &= \frac{\vec{c}_{vc}^{\,T} \cdot E \cdot \vec{v}(\tau)}{\vec{c}_{0}^{\,T} \cdot E \cdot \vec{v}(\tau)} \ . \end{aligned} \tag{2.8}$$

For monitoring functions 2.6 - 2.8 the threshold values are user defined, and threshold crossing can be monitored in both directions.

2.3.3 Requirements Analysis

The application has to fulfill requirements with respect to scalability, accuracy of results and privacy.

First, in order to be successfully applied in practice, the application must be able to handle a very large number of visit patterns. We retrieved about 800,000 points of interest (POI) from the

Open Street Map (OSM) geo-service (Maps) for Germany's 15 most relevant location types (e.g. restaurant, shop, bank, school, cinema). Even though it is likely that a person visits POI only in a geographically restricted region, an exponential number of POI combination can be formed for mobility patterns. The large number of POI and visit patterns have an impact on the local nodes because they have only limited resources for the identification of visits and visit patterns.

Second, the application has to be scalable to a very large number of local nodes. For example, when applied in Germany a potential of 82 million local nodes (the inhabitants) exist. In consequence, the communication and aggregation at the coordinator form bottlenecks to the application. As our application relies on mobile phones, it has to face less severe communication restrictions than typical sensor networks, and we will therefore not focus on this issue. However, we want to apply sketching methods to enable fast aggregation and storage reduction at the coordinator.

Third, the results obtained at the global coordinator should stay within given (probabilistic) accuracy bounds. For most analyses it is not necessary to obtain exact results. However, the error variance of the results should be known and within reasonable bounds.

Finally, as mobility data is very sensitive, it is important to ensure the protection of user privacy. In particular we want to ensure that an attacker a) cannot infer the current position of a user and b) cannot infer historic movements or movement patterns of a user. In addition, we assume that the coordinator is untrusted, which is opposed to most privacy approaches currently used in trajectory data analysis (e.g. relying on k-anonymity (Monreale et al., 2010; Nergiz et al., 2009; Abul et al., 2008; Monreale, 2011)).

2.3.4 Employed Data Sets

The Arbeitsgemeinschaft Media-Analyse e.V. (ag.ma) is a joint industry committee of German advertising vendors and customers. Starting in 2006 it commissioned a yearly nationwide mobility survey as basis for an objective performance evaluation of outdoor advertisements in Germany. About 22% of the surveved persons were provided with GPS devices to record their movements over one week, while the mobility of the other persons was captured in a computer assisted telephone interview (CATI).

The GPS data has been collected in three waves in the years 2007, 2009 and 2011 and contains in total 14.763 persons (Arb, 2012). The GPS measurements are concentrated on 42 selected cities in Germany including the 15 largest cities in Germany, major cities above 100,000 inhabitants as well as a number of cities with inhabitants between 50,000 and 100,000 inhabitants (see Figure 1 left). The duration of GPS recording is one week per person, however, the measurement periods are distributed over one year such that a representative mobility sample for the whole year is obtained. The participants of the survey are equipped with GPS devices, which they are asked to carry along during the measurement period. The GPS devices record positions every second as long as the devices are turned on. Measurement gaps may arise if participants forget to charge the devices or if they turn them off deliberately. In addition, faulty measurements may arise if participants forget to carry the GPS device along with them. In order to distinguish between days without mobility of the participant and between days with erroneous measurements, a follow-up survey is conducted where the participants can specify the validity of measurement days with low movement or few position records.

In addition to movement information, the survey provides detailed socio-demographic information about the participants including gender, age group, size of household, level of education, type of profession, household income as well as mobility characteristics such as possession of a driving license, use of private vehicles or use of public transport.

2.3.5 Results

Results on Scalability

In order to apply LIFT techniques in the scenario, fast and resource-preserving methods have to be devised for the local evaluation of mobility data. In (Florescu et al., 2012; Florescu, 2012) we

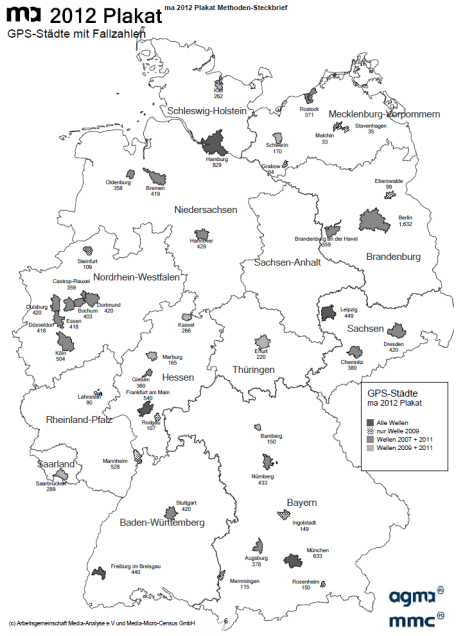


Figure 2.8: Municipalities with GPS measurements (source: Arbeitsgemeinschaft Media-Analyse e.V. and Media-Micro-Census GmbH, ma 2012 Plakat Methoden-Steckbrief zur Berichterstattung, 29.08.2012)

have shown that the implementation of the local nodes can be done efficient, handling 800,000 locations and up to 10,000 complex patterns in less than one second on a Samsung Galaxy SII (I9100). For handling more locations or more patterns, measures can be taken to reduce the number of GPS position updates by configuring the Android Location Provider appropriately or by adding intermediate GPS smoothing filters. For example, for a frequency of 5 seconds per position update request, our application can efficiently scale up to at least 800,000 locations and over 300,000 complex patterns.

We further investigated methods for scaling the proposed method to thousands and potentially millions of user nodes. Consider, for example, that every person traverses more than 200 street segments per day. If we assume further that each person visits 10 different locations (e.g. work location, shops, bus stops) per day and 20 million persons participate in data collection, about 4.2 billion visit events occur every day. In order to cope with this number of events, sophisticated analysis and storage algorithms as well as a sophisticated system architecture have to be devised. The key component of our approach is to use exponential histograms for data compression. This data structure has the advantage that it can be applied in a distributed setting as previously shown in work of LIFT partners (Papapetrou et al., 2012) and thus allows for scalability when the number of users increases. In addition, exponential histograms offer sliding window query capabilities with a guaranteed maximum relative error and can thus be used in an online setting.

We performed experiments using a flat and hierarchical architecture as shown in Figure 2.9. The lowest layer holds the user nodes, which collect the users' GPS data and extract visit events. The visit events are forwarded either directly to the central coordinator (flat setting) or to a layer of intermediate nodes (hierarchical setting). In the flat setting, the coordinator aggregates the visit information of each point of interest (POI) in an exponential histogram. In the hierarchical setting the exponential histograms reside already at the intermediate nodes. In regular time intervals the intermediate nodes submit the exponential histograms to the coordinator, which merges them and answers user queries. The layer of intermediate nodes was introduced for horizontal scalability and to avoid an overload of the coordinator. However, it also serves a privacy purpose given that the intermediate nodes do not collude (see (Kopp et al., 2012)). As a user can freely select an

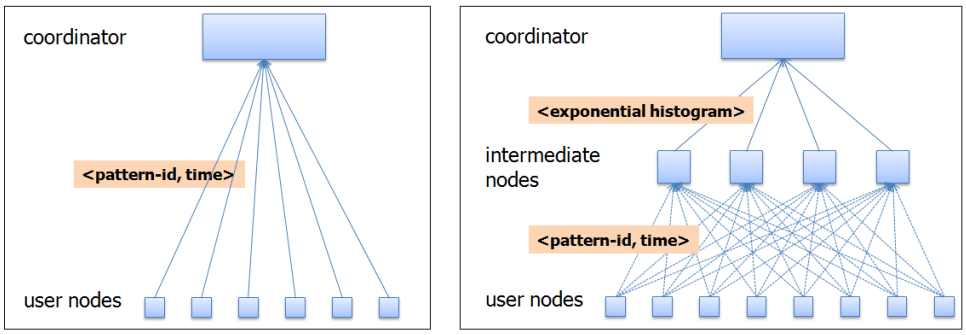


Figure 2.9: System architecture; left: flat setting; right: hierarchical setting

intermediate node when submitting a visit event, no intermediate node will obtain the whole event history of a single user.

We performed experiments using a subset of the above introduced data set containing 2,967 persons with a total of 304 million GPS points as well as 400,988 POI from OSM marked with the keys *shop, amenity, leisure, tourism, historic, sport, public transport, railway*. From the data we extracted 23,508 visit events to 11,809 different POI. As this number was considerably below our expectations (see also (Kopp et al., 2013) for further explanations), we replicated the visit data by a factor of 1,000 adding Gaussian noise in the temporal dimension. In our experiments we conducted point queries in a sliding window fashion. I.e., we queried the number of events per POI in the past Δt seconds. The selected query windows were of length 30, 600, 1800, 3600 or 86400 seconds. We performed those queries every 10 minutes (in the hierarchical setting this coincides with the time interval of the force action). For our observation period of one week this resulted in $n_t = 1{,}008$ queries per query window for each of the $n_p = 11{,}809$ visited POI. Further, we varied the maximum acceptable relative error ε to take the values 0.01, 0.02, 0.04, 0.08 and 0.16. In the hierarchical setting we used 10 intermediate nodes which submitted their data structures every 600 seconds to the coordinator. We measured the error for each experiment using the mean absolute percentage error (MAPE), which is defined as follows:

$$MAPE = \frac{\sum_{i=1}^{n_p} \sum_{j}^{n_t} \left| \frac{x_{ij} - \widehat{x}_{ij}}{x_{ij}} \right|}{n_p \cdot n_t}$$

where x_{ij} denotes the true number of visit events at POI i in query window j and $\widehat{x}_{ij}$ denotes the number of events returned from the exponential histogram.

The results are depicted in Figure 2.10. Our experiments show that the resultant error is very low. For all settings of ε the mean error (MAPE) is less than 1/10 of the maximum acceptable error. This is a very good result. Especially we can be sure for small total number of visits that the query results are always correct. For example, setting $\varepsilon = 0.01$ will result in no errors if less than 100 events occur per POI. This is an important characteristic because the visit frequency of POI is right-tailed, containing only few POI with very high frequencies. Further the experiments show that our setting scales horizontally. By introducing a layer of 10 intermediate nodes, the MAPE was on average 7.5% higher and at most 23% higher than in the flat setting. Both numbers are considerably below the maximum acceptable error as well as the maximum relative error guaranteed for the join of exponential histograms. Finally, to evaluate the memory usage, we can compare the numbers to the following baseline scenario. Whenever a visit event occurs, the POI identifier and timestamp are stored at the coordinator using two 4 Byte integers. As our sliding window covers only one day, we will assume that we have to store 1/7 of the total visit events. For the original 23,509 events this results in 0.026 MB. For the multiplied data set it results in 25.6 MB. The storage amount for the original events using exponential histograms varied between 0.54-4.7 MB. In this case we did not save on memory. However, using the more realistic multiplied data set with exponential histogram sizes between 2.5-14.5 MB, our experiments require only 9.7-

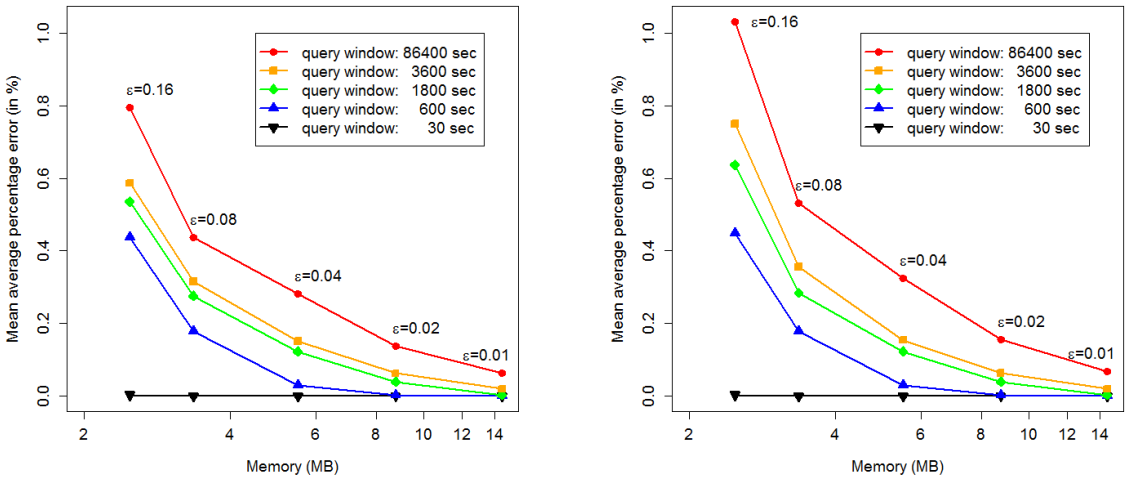


Figure 2.10: Mean average percentage error and memory usage for different maximum relative errors (ε) and query window sizes; left: without intermediate nodes; right: hierarchy with 10 intermediate nodes

56.6% of the baseline storage space depending on the selected ε. Considering the entire approach including exponential histograms and loca evaluation, the storage reduction is even much higher compared to a naive centralized setting where users submit a GPS position every second to some central coordinator.

In conclusion, local evaluation combined with exponential histograms allows to substantially reduce storage space while maintaining a high accuracy of query results. Due to the composability of exponential histograms the technique can be applied in a distributed setting, which reduces the load on the coordinator and thus ensures scalability to a large number of local nodes.

Results on Privacy

In the requirements analysis we stated that we assume an untrusted coordinator and that we want to ensure that an attacker a) cannot infer the current position of a user and b) cannot infer historic movements or movement patterns of a user.

The current position of a user is not revealed in our setting because only aggregated statistics without user identifiers are transmitted to the server. Aggregation takes place over time and possibly over space. Clearly, the longer the monitored time interval is, the stronger the aggregation. Spatial aggregation can be achieved, for example, by a typification of locations. I.e. instead of monitoring specific locations, all locations of a certain type (in a given region) can be specified in a pattern. Typification can also be performed locally if the coordinator shall not know the exact position of some location. For example, the "home" of a user might locally be resolved for visit detection, but a location identifier referring only to the type of location (i.e. "home") and not to its position is transmitted to the coordinator.

The movement history of a user is protected by independently submitting aggregated statistics for each observed POI or visit pattern. Recall from the description in section 2.3.2 that the communication to the coordinator is anonymized w.r.t. the sender addresses (e.g. IP number) in such a way that different messages originating from the same sender cannot be related to each other. Hence, a simple aggregation of information from different messages just based on the sender addresses is prevented by technical infrastructure means. Hence, the adversary may only rely on the contents of individual messages for trying to retrieve information that would allow to set up a personal mobility profile. Thus, it has only to be ensured that the monitored visit patterns are not too specific and become a mobility profile by themselves.

Further, an attacker might try to use the visit information as quasi-identifier. For example, a

location set containing the "White House" would basically reveal some of the mobility behavior of a very restricted set of persons. In addition, identification may be possible if a user has very extreme visit counts which only very few people may reach, such as an animal keeper in a zoo. In fact, the level of anonymity related to this kind of attack can be directly specified from our model in terms of k-anonymity. In order to distinguish between the parameter in the anonymity model and our visit model, we will use the calligraphic letter $\mathcal{K}$ to refer to the former. Given the global aggregate $\vec{v}(\tau)$ of some location set, the level of $\mathcal{K}$-anonymity is given by $\mathcal{K} = min\{x_i \mid x_i > 0\}$. I.e. the smallest positive frequency of k-visiting entities determines the anonymity threshold. Translated to a database representation, the derivation of $\mathcal{K}$ becomes obvious. Each distribution $\vec{v}(\tau)$ can be disaggregated in a table with one column containing for each user the value k of occurrences of a given visit pattern. As the table contains only a single column, the minimum number of users with the same k correspond to the anonymity level. The maximal information that an adversary may retrieve from a violation of some anonymity threshold, however, is the frequency count of a user for the visit pattern and time interval in question. It is not possible to retrieve mobility profiles involving any other (spatially distant) locations.

In future work we plan to implement and test algorithms that ensure that each global data vector conforms to $\mathcal{K}$-anonymity. As the anonymization requires knowledge about all visit pattern occurrences, it has to be performed in a centralized way. If the coordinator is still assumed to be untrusted, a semi-trusted coordinator can be introduced between the nodes and the coordinator for this purpose. In order to ensure $\mathcal{K}$-anonymity the semi-trusted coordinator needs to know only the global data vectors but not their semantic interpretation. Thus a higher level of privacy is achieved compared to an anonymization via the original coordinator.

2.3.6 Discussion

When applying LIFT technologies to the monitoring of mobility patterns we discovered that the local evaluation principle provides a very natural way for the privacy-preserving handling of large sets of mobility data. By local aggregation a massive reduction of data can be achieved which ensures at the same time user privacy. In addition, we could show that the scenario is applicable to real-world situations where an efficient local processing of mobility data has to be ensured and a large number of local nodes has to be served.

However, we also discovered situations in which current LIFT technology is not optimal and requires further research efforts. First, the monitoring functions of our scenario rely on the *sum* of local data vectors and not on their average. Although the difference lies only in the division by a constant factor (per use case), it means that the safe zones depend on the number of user nodes and become very small with increasing users. A change at node level therefore leads inevitably to a violation. In order to solve this problem, a combination of safe zone and sampling techniques may be investigated to submit alerts only if a large portion of user nodes violates their safe zone.

Second, we experienced that some local summarization techniques are not applicable in our scenario because their error guarantees are too lose. Typically, the error of local summarization techniques is bounded by the sum (of squares) of the global pattern frequencies. This is sufficient to find e.g. heavy hitters. However, in our scenario we have comparably evenly distributed visits or visit patterns. For example, supermarkets or shops of a given size have a natural upper bound on their visitors. Given a large number of patterns with comparably low frequency makes it therefore hard to answer queries precisely. In the application scenario we therefore applied a simpler summarization technique. Instead of ECM sketches we used exponential histograms, which are one building block of ECM sketches. This technique proved to have a high accuracy while reducing the storage space considerably.

Finally, we discovered that local summarization techniques cannot be efficiently applied at the node level in this scenario because pattern events of a single user occur too seldom. However, this finding turned out to be no restriction for the scenario. On the one hand, the main data reduction takes place due to the local evaluation of visit patterns (e.g. reducing 304 million GPS points to about 23,500 visit events). On the other hand, communication takes place over the mobile phone network and the resultant messages on the occurrence of visit patterns are comparably infrequent

and small. Nevertheless, distributed summarization techniques are crucial in our scenario to scale the architecture to thousands or millions of user nodes by reducing the load on the coordinator.

In summary, the monitoring of visit patterns is well suited for the application of LIFT techniques and profits from the data reduction and privacy achievements of local evaluation.

2.4 Distributed Estimation of Traffic Flows

2.4.1 Motivation

The understanding of the human mobility behavior in a city is important for improving the daily human mobility, for managing the traffic network and for reducing traffic jams.

Thanks to the recent advent of inexpensive positioning technologies, data about movement of various mobile objects are collected in rapidly growing amounts. Potentially, these data are a source of valuable knowledge about behavioral and mobility patterns. To gain an understanding of these patterns, an analyst needs a visual representation of the data, which is the most effective way to support human perception, cognition, and reasoning. An aggregated view enables the traffic managers to evaluate the situation of mobility by means of an effective visualization. The generalization of trajectories can be used to provide a high level description of traffic flows in a given time interval. The flows can be rendered directly on the screen to highlight the critical paths of the road network. Each flow is rendered with a stroke height proportional to the flow. Andrienko and Andrienko (2011) propose a method for trajectory generalization that requires to collect in a central station all the raw trajectories. This centralized setting highlights two main problems: a) the large mass of information to be transmitted and processed might be too much, and b) raw trajectory data to be transmitted describe mobility behavior of the users with great detail that could enable the inference of very sensitive information related to the personal private sphere of individuals. In order to solve these two very important problems we propose a privacy-preserving distributed computation framework for the aggregation of movement data for a comprehensive exploration of them. In order to reduce the amount of information to be communicated, we propose the application of sketches to the data. The data transformation based on sketches allows us to obtain a preliminary level of data privacy that is the first step towards the realization of a privacy-preserving framework.

2.4.2 Scenario Description and Formalization

The goal of our application is to estimate traffic flows and correlations between traffic flows. In our scenario the coordinator is responsible for computing the flows on a territory by combining the information received by each node. In order to obtain traffic flows in the centralized setting we need to generalize all the trajectories by using the cells of a partition of the territory. In our distributed setting we assume that the partition of the territory, i.e., the set of cells $C = \{c_1, \ldots, c_m\}$ useful for the generalization, is known by both all the nodes and the coordinator. Each node, that represents a vehicle that moves in this territory, in a given time interval observes a sequence of spatio-temporal points (trajectory), generalizes it and contributes to the computation of the global function.

Before describing the local and global functions computed by the nodes and the coordinator, we introduce some basic notions that help the understanding of our problem.

Definition 2.4.1 (Trajectory)**.** *A Trajectory or spatio-temporal sequence is a sequence of pairs* $T =< l_1, t_1 >, \ldots, < l_n, t_n >$*, where* t_i *(*$i = 1 \ldots n$*) denotes a timestamp such that* $\forall_{1<i<n}\ t_i < t_{i+1}$ *and* $l_i = (x_i, y_i)$ *are points in* $\mathbf{R}^2$*.*

Intuitively, each pair $< l_i, t_i >$ indicates that the object is in the position l_i at time t_i.

We assume that the territory is subdivided in cells $C = \{c_1, c_2, \ldots, c_t\}$ which compose a partition of the territory. During a travel a user goes from a cell to another cell. We use g to

denote the function that applies the spatial generalization to a trajectory. Given a trajectory T this function generates the generalized trajectory $g(T)$, i.e. a sequence of *moves* with temporal annotations, where a *move* is an pair (l_{c_i}, l_{c_j}), which indicates that the moving object moves from the cell c_i to the cell c_j. Note that, l_{c_i} denotes the pair of spatial coordinates representing the the centroid of the cell c_i; in other words $l_{c_i} =< x_{c_i}, y_{c_i} >$. The *temporal annotated move* is the quadruple $(l_{c_i}, l_{c_j}, t_{c_i}, t_{c_j})$ where l_{c_i}, is the location of the origin, l_{c_j} is the location of the destination and the t_{c_i}, t_{c_j} are the time information associate to l_{c_i} and l_{c_j}. As a consequence, we define a generalized trajectory as follows.

Definition 2.4.2 (Generalized Trajectory)**.** *Let $T =< l_1, t_1 >, \ldots, < l_n, t_n >$ be a trajectory. Let $C = \{c_1, c_2, \ldots, c_t\}$ be the set of the cells that compose the territory partition. A generalized version of T is a sequence of temporal annotated moves $T_g = (l_{c_1}, l_{c_2}, t_{c_1}, t_{c_2})(l_{c_2}, l_{c_3}, t_{c_2}, t_{c_3}) \ldots (l_{c_{m-1}}, l_{c_m}, t_{c_{m-1}}, t_{c_m})$ with $m <= n$.*

Now, we show that a generalized trajectory can be represented by a frequency distribution vector. First, we define the function *Move Frequency* MF that given a generalized trajectory T_g, a move (l_{c_i}, l_{c_j}) and a time interval computes how many times the move appears in T_g by considering the temporal constraint. This function is the base for the computation of the local data vector calculated by each node. More formally, we define the *Move Frequency* function as follows.

Definition 2.4.3 (Move Frequency)**.** *Let T_g be a generalized trajectory and let (l_{c_i}, l_{c_j}) be a move. Let τ be a temporal interval. The* Move Frequency *function is defined as:*

$$MF(T_g, (l_{c_i}, l_{c_j}), \tau) = |\{(l_{c_i}, l_{c_j}, t_i, t_j) \in T_g | t_i \in \tau \wedge t_j \in \tau\}|.$$

This function can be easily extended for taking into consideration a set of generalized trajectories $\mathcal{T}^{\mathcal{G}}$. In this case, the information computed by the function represents the total number of movements from the cell c_i to the cell c_j in a time interval.

Definition 2.4.4 (Global Move Frequency)**.** *Let $\mathcal{T}^{\mathcal{G}}$ be a set of generalized trajectories and let (c_i, c_j) be a move in the graph G. Let τ be a time interval. The* Global Move Frequency *function is defined as:*

$$GMF(\mathcal{T}^{\mathcal{G}}, (c_i, c_j), \tau) = \sum_{\forall T_g \in \mathcal{T}^{\mathcal{G}}} MF(T_g, (c_i, c_j), \tau).$$

The the number of movements between two cells computed by either the function MF or GMF describes the amount of traffic flow between the two cells in a specific time interval. This information can be represented by a frequency distribution vector.

Definition 2.4.5 (Vector of Moves)**.** *Let $C = \{c_1, c_2, \ldots, c_t\}$ be the set of the cells that compose the territory partition. The* vector of moves *M with $s = 2 \times \binom{t}{2}$ positions is the vector containing all possible pair of cells. The element $M[i] = (c_i, c_j)$ is the* move *from the cell c_i to the cell c_j.*

Definition 2.4.6 (Frequency Vector of a Generalized Trajectory)**.** *Let $C = \{c_1, c_2, \ldots, c_t\}$ be the set of the cells that compose the territory partition and let M be the vector of moves. Let τ be a time interval. Given the generalized trajectory $T_g = (l_{c_1}, l_{c_2}, t_{c_1}, t_{c_2})(l_{c_2}, l_{c_3}, t_{c_2}, t_{c_3}) \ldots (l_{c_{m-1}}, l_{c_m}, t_{c_{m-1}}, t_{c_m})$, for $i = 1, \ldots, s$ we can construct the corresponding* frequency vector *f with size $s = 2 \times \binom{t}{2}$ where each $f[i] = MF(T_g, M[i], \tau)$.*

Clearly, the frequency vector of a generalized trajectory is the local data vector computed by a node by using the Move Frequency function MF.

Given the set of vehicles $V = \{V_1, \ldots V_k\}$, the global data vector is the sum of the k local data vectors (with $s = 2 \times \binom{t}{2}$ positions) in a time interval τ:

$$f^{\tau}(V) = \sum_{j=1}^{k} f^{V_j} = [f_1, f_2, \ldots, f_s],$$

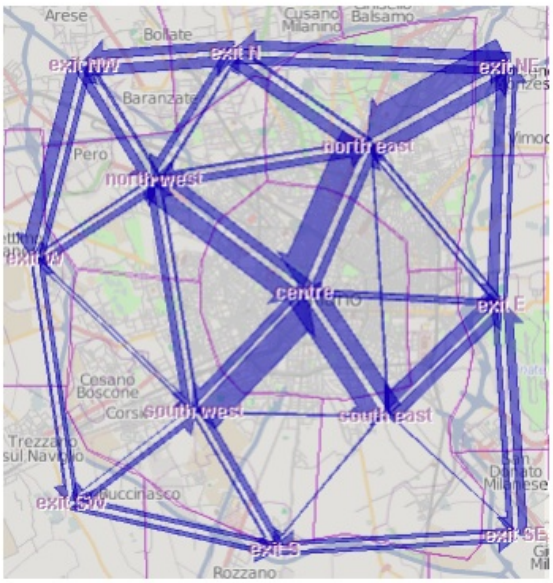


Figure 2.11: Visual summarization of a set of trajectories

where each

$$f_i = \sum_{j=1}^{k} f^{V_j}[i] = GMF(\mathcal{T}^{\mathcal{G}}, M[i], \tau).$$

Here, $\mathcal{T}^{\mathcal{G}}$ is the set of generalized trajectories related to the k vehicles V in the time interval τ and f^{V_j} is the frequency vector of the trajectory observed by the vehicle V_j in the time interval τ.

Our problem can be defined as follows.

Definition 2.4.7 (Problem Definition)**.** *Given a set cells $C = \{c_1, \ldots, c_m\}$ and a set $V = \{V_1, \ldots V_k\}$ of vehicles distributed, the* distributed estimation of traffic flows problem *consists in computing in a specific time interval τ the $f^{\tau}_{DMAP}(V) = [f_1, f_2, \ldots, f_s]$, where each $f_i = GMF(\mathcal{T}^{\mathcal{G}}, M[i], \tau)$ and $s = 2 \times \binom{m}{t}$. Here, $\mathcal{T}^{\mathcal{G}}$ is the set of generalized trajectories related to the k vehicles V in the time interval τ and M is the vector of moves defined on the set of cells C.*

Solving this problem means to define a process describing the computation of each node and of the coordinator. In the following we describe the main steps of the computation of both.

Node Computation. Each node represents a vehicle that moves in a specific territory and this vehicle in a given time interval observes a sequence of spatio-temporal points (trajectory) and computes the frequency vector to be sent to the coordinator. The steps to compute this data vector are:

Trajectory generalization. The vehicle computes the generalized version of its trajectory by substituting spatial points with centroids of cells and transforms the trajectory in a sequence of *moves* annotated with the temporal information. So, the node given the original trajectory T obtains the generalized version T_g (see Definition 2.4.2)

Frequency Vector Construction. Starting from the generalized trajectory T_g the node constructs a frequency vector of this generalized trajectory (Definition 2.4.6) by counting how many times each move occurs in its trajectory, by using the function introduced in Definition 2.4.3. In particular, the vehicle populates the local data vector f^{V_j} according to the generalized trajectory observed. So, the element $f^{V_j}[i]$ contains the number of times that the vehicle V_j moved from m to n if $M[i] = (m, n)$.

Sketching of the Frequency Vector. The node applies a sketching technique to represent the frequency vector f^{V_j} using a much smaller *sketch vector* and reducing the amount of information to be communicated.

Coordinator Computation. The coordinator computes the global vector of traffic flows in a given time interval by combining all local sketches. The global sketch thus summarizes all local flows and can be queried for traffic analysis. In the case of correlation analysis between traffic flows, the coordinator has to reconstruct the matrix of local traffic flows. When reconstructing each local vector from the sketch vector, the coordinator obtains only an approximate version of the original data vector. From these vectors it computes the global flow matrix where each row represents a vehicle and each column a specific flow. In addition to correlations, other functions can be monitored based on the matrix as, for example, the relations between the amount of traffic in different cells (traffic share).

2.4.3 Requirements Analysis

Privacy. The main requirement in the above application is the protection of the users' privacy. We consider sensitive information any information from which the precise location or mobility behavior of an individual at a given time can be inferred. Each node sends the number of times that he/she goes from an area to another area. By combining different cell transitions it is possible to identify typical movements of a specific user. An obvious privacy measure is to anonymize the communication of the vector by removing identifiers such as network addresses. This approach is insufficient, however, because drivers can often be re-identified by correlating anonymous location traces with identified data from other sources. So, we need to find privacy measures to avoid the re-identification. In general, we assume that the coordinator is not a trusted entity, therefore, the idea is to hide the real count associated to each move, in order to generate uncertainty during the reconstruction of the user movements. The use of the sketching procedure introduces a first preliminary form of privacy protection due to the approximation derived from the vector reconstruction. In order to increase the level of privacy, before the sketching transformation another data vector transformation able to introduce more confusion and so more data privacy should be applied. One possible direction is a suitable perturbation of the vector data that allows us both local data protection and global data quality. During the third year we developed a suitable privacy-preserving technique based on the well known differential privacy. Details on that can be found in Deliverable on D2.2 "Privacy-preserving Methods".

Communication bottleneck. The application described above involve hundreds of thousands of vehicles that are located in any place of the territory. Each vehicle submits the information contained in the local vector to the coordinator, which has a size depending on the number of cells that represent the partition of the territory. The number of cells in a territory can be very huge and this can make each local vector too big. For example, in one of our dataset of trajectories on Tuscany, we have a number of cells that subdivide the territory equally into about 2600 cells. Therefore, the application has to be able to handle both a very large number of nodes and a large amount of the information to be communicated. These considerations make the reduction of the information communicated necessary. However, the application of sketching methods allows us to apply a good compression of the information introduced.

Data Utility. Lastly, the results obtained at the global coordinator should stay within given accuracy bounds. Unfortunately, data transformations due to both privacy requirements and the necessity of reducing the transmitted information introduce some approximation that describes the information loss and so, allows to define the quality and utility of the data computed. This approximation should be kept under control by identifying specific bounds.

2.4.4 Employed Data Set

The dataset used for testing this application are two:

- a dataset of trajectories of around 38k vehicles located in the western part of Tuscany, thus covering a large territory and mixed land usages (residential areas, industrial zones, countryside, suburbs, etc.), and spanning over 5 weeks across June and July 2010.
- a dataset of trajectories that contains a set of trajectories obtained by 17,000 private cars with on-board GPS receivers and moving during one week in April 2007 in the city of Milan (Italy).

The dataset was collected by Octotelematics S.p.A. (Octotelematics).

2.4.5 Results

For the evaluation of our approach that includes a privacy transformation we applied it to the real movement data set described above. We assumed that on-board location devices in vehicles continuously trace the positions of the vehicles and periodically send statistical information about their movements to a central station. The coordinator will store the received statistical information and compute a summary of the traffic conditions of the whole territory, based on the information collected from individual vehicles. We deeply studied the trade-off between privacy and data utility of each privacy transformation and the impact of the further transformation required by the sketch-based transformation. Concerning the sketching algorithm we tested the following algorithms: *Count*, *Count-min* and *AGMS*. Since the coordinator reconstructs the flows among the zones of the territory division, we can represent such data as a directed graph, where the nodes represent the zones and an edge between two nodes represents the flows from one zone to the other. Therefore, we compared the transformed data with the original data by varying the transformation parameters and analyzed the cumulative distribution of some network measures and their linear correlations by means of the Pearson Correlation Coefficient (PCC). The results obtained and discussed in Monreale et al. (submitteda, 2013a,b); Pratesi (February, 2013); Pratesi et al. (2013) show that most of the network-based measures are preserved. The good results are also confirmed in our experiments about the mobility analyses on the transformed data. We found that an analyst can use the differential private and sketched data for important mobility data analyses obtaining results with good quality. In Figure 2.12 we show two plots highlighting how the privacy transformation impacts the preservation of the most important measures: the *flow per link*, i.e., the preservation of the amount of flow between the areas, and the *flow per zone*, i.e., the preservation of the amount of traffic flow within each area. While Figure 2.13 provides the PCC of the same measures after the application of the *Count* sketching algorithm.

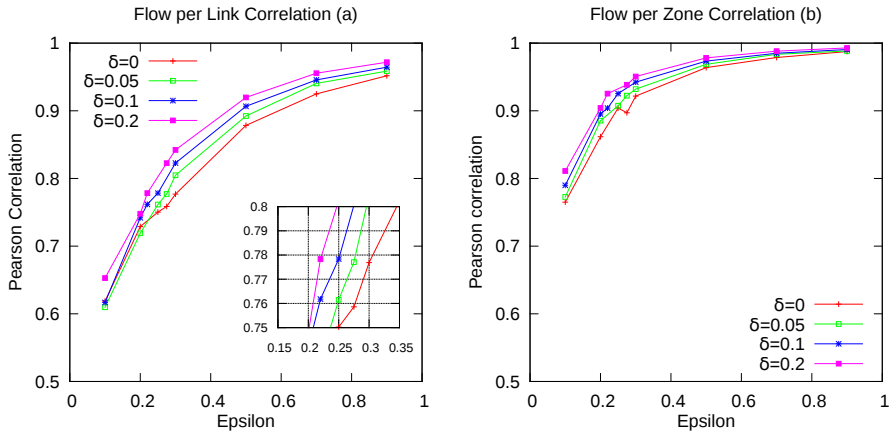


Figure 2.12: PCC for flow per link and flow per zone measures after privacy transformation

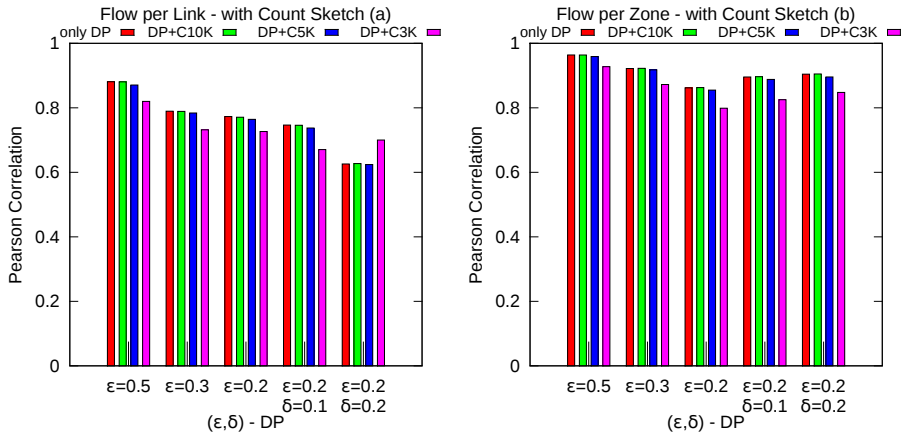


Figure 2.13: PCC for flow per link and flow per zone after Count Sketch application

2.4.6 Discussion

This application represents an innovative distributed framework for privacy-aware sharing of user movement data. A typical centralized setting for generalization and aggregation of movement data entails two important problems: a) the amount of information to be collected and processed may exceed the capacity of the storage and computational resources; and b) the raw data describes the mobility behavior of the individuals with great detail that could enable the inference of very sensitive information related to the personal private sphere.

We developed a distributed setting for this important task by exploiting the properties of the sketching algorithms for reducing the amount of data to be communicated and by integrating privacy technologies in the analytical process. Therefore, the LIFT technology allows us to solve one of the most important problems that characterize distributed frameworks as ours. The application of both sketching and privacy transformations lead to some data distortion due to the level of approximation introduced by these two steps. This requires to manage a trade-off among data utility, privacy and reduction of communications. In our experiments performed on real-world data we deeply evaluated the impact of these two steps on the data utility and we found that some important analyses on the transformed data are still applicable.

This application with the integration of privacy and sketching is compatible with the *user-centric model* for the personal data management supported by the World Economic Forum in its last report. The basic idea is to introduce high levels of transparency and full control for the user on the life cycle of his personal data (e.g., collection, storage, processing, sharing). Thus, the user has to have a crucial and active role in a righteous and fruitful ecosystem based on personal data and should be encouraged to give access to his own data (with the level of privacy desired) to receive advantages once he discloses them to a company or a public authority, according to his wills and needs. In our application the private traffic data, collected by the central station, are still useful for important collective mobility and traffic analyses and each user can release the data after setting the desired privacy level, compatible with his privacy expectation.

In the current version of the application we did not integrate the safe zones technology, due to the fact that it does not consider a specific thresholding function to be monitored on the basis of the user movement data. Therefore, our future activities on this line include the extension of the approach by defining one or more thresholding functions and by considering also the safe zones methods in the system.

Chapter 3

Internet Scale Query Processing

3.1 Motivation

With the advent of the Internet as the main tool for information sharing and publishing, it is now obvious that the largest "big data" collections are going to be generated inside the network, and, thus, will inevitably be *dynamic* (i.e., streaming) and *massively distributed*. Specific example scenarios include:

- *Distributed network-traffic analysis monitors* that try to track the correlation of traffic patterns to the same destination IP addresses across two collections of routers (possibly, in different administrative domains), e.g., for tracking network health (e.g., potential DDoS activity) or improving network-load balancing in real time (Huang et al., 2007). Another important task involves identifying IP addresses with abnormal behavior, and isolating them before they can bring down the network.

- *Peer-to-Peer (P2P) systems* that want to track content and query similarity across a large collection of peers to enable effective peer clustering and formation of semantic online user communities.

The sheer size of such *Internet-scale* systems combined with the volumes and dynamic nature of the underlying data streams put it well beyond the capabilities of any existing distributed/parallel data-management architecture (Huebsch et al., 2005). Centralizing all this data to enable intelligent global data analysis is simply infeasible, due to the enormous volume and dynamic nature of the data, as well as numerous administrative, ownership, and privacy constraints. Instead, the only possible way to enable intelligent analysis and monitoring of such network-bound streams is by querying the data *in situ*, without the need for traditional database design, integration, and maintenance (Hellerstein et al., 2007). This, in turn, implies the need for novel, massively-distributed query processing tools that rely on localized data-stream processing, and can efficiently scale up to thousands of participating nodes (i.e., work at Internet-scale). Data processing in an ISQP system will mostly be data- and event-driven – in other words, it must rely on communication-efficient techniques for massively-distributed, intelligent event processing, triggers (i.e., threshold-crossing conditions), and continuous queries.

3.2 Scenario Description and Formalization

Focusing on a network-traffic monitoring scenario, let R and S denote the union of the packet streams across two distinct sets of routers (e.g., at the edge of an enterprise network). A natural measure of the correlation (or, similarity) of the two destination-IP traffic streams can be expressed through the size of an SQL join query:

```
SELECT COUNT(*) FROM R, S WHERE R.destinationIP = S.destinationIP
```

Letting f_R and f_S denote the frequency distribution vectors of the R and S streams (respectively) across all possible destinations, it is easy to see that the above join size essentially computes the similarity of the R and S destination traffic as the *inner product* of f_R and f_S (i.e., the (un-normalized) cosine similarity of the two traffic vectors). That is, assuming N-dimensional distribution vectors f_R and f_S, it computes the non-linear function

$$g(f_R, f_S) = |R \bowtie_{destIP} S| = f_r \cdot f_s = \sum_{destIP} f_R[destIP] f_S[destIP]$$

over the $2N$-dimensional vector $[f_R, f_S]$ A continuous distributed *threshold query* (or, trigger) would monitor the event that the similarity of the R and S traffic patterns crosses a certain threshold – that is, $g(f_R, f_S) > T$, or, after normalizing by L_2 norms, $g(f_R, f_S) > T||f_R||_2||f_S||_2$ – in order to trigger some appropriate course of action. A generalization of this scenario is a *continuous approximate tracking* query where the goal is to continuously track the value of the similarity of the distribution vectors to within a θ approximation; that is, continuously maintain an estimate $\hat{g}$ at the coordinator such that, at any point in time, $\hat{g} \in (1 \pm \theta) g(f_r, f_s)$. Similar join queries and threshold/approximate-tracking conditions (based on cosine vector similarity) could be used to monitor the semantic similarity of two collections of peers in a P2P system; of course, in this case, the underlying distributed dynamic vectors would correspond to the term frequency distributions of the peers' query streams or document contents. These queries can be naturally generalized to the broader class of multi-way distributed stream joins (corresponding to inner products of multi-dimensional tensors) (Cormode and Garofalakis, 2008). Special cases of such distributed join queries are also of interest – for example:

- L_2 **norm queries (i.e., self-join sizes)**: In this case, $f_R = f_S$, and the target function essentially computes the (squared) L_2 norm (or, self-join size) of a distributed, dynamic data-distribution vector, i.e., $g(f_R, f_S) = g(f_R) = ||f_R||_2^2 = \sum_i (f_R[i])^2$. The self-join size represents important demographic information about a data collection; for instance, its value is an indication of the degree of skew in the data (Alon et al., 1996).

- **Range Queries**: In this case, the vector $f_S = I_{[a,b]}$ is a constant vector representing a "1-pulse" over a range $[a, b]$ of the data distribution domain, and $g(f_R, f, S) = \sum_{i=a}^{b} f_R[i]$ is a range-sum query over the $[a, b]$ range. Such range queries are critical in estimating histogram or wavelet representations of the data, and approximating the values of *heavy-hitter* data elements.

Another important application scenario involves tracking the set of *skyline (i.e., dominating or Pareto-optimal) points* across the massive distributed collection of router traffic statistics for all IP addresses in a large ISP domain. A typical configuration involves installing monitoring code at the edge routers of the ISP to collect workload statistics over sliding windows for a set of IP addresses served by the ISP. Skyline queries on the data *aggregated* over all edge routers are powerful tools for network administrators for to quickly identifying problematic IP addresses or interesting network events, such as DoS attacks. For example, the skyline of the average (over all routers) number of packets and transfer volume, per target IP (data shown in Fig. 3.1(b)), helps an administrator to quickly focus on the IPs under attack. The skyline dimensions can even be defined through *complex, non-linear functions on the aggregated data*, such as the variance on the workload per IP, collected by the edge routers (Fig. 3.1(c)) – a key indicator for sites under a DoS attack.

Our goal is to continuously monitor distributed skylines over complex functions of fragmented multi-dimensional objects. More formally, at time t, the local state of each object o_j at site p_i is captured by a dynamic d-dimensional *local statistics vector* $\vec{v}(o_j, p_i, t)$. The global state of o_j is defined as the average of o_j's local statistics vectors across all sites in $\mathcal{P}(o_j)$, i.e., the *global statistics vector* $\vec{v}(o_j, t) = \frac{1}{|\mathcal{P}(o_j)|} \sum_{p_i \in \mathcal{P}(o_j)} \vec{v}(o_j, p_i, t)$. The dimensions of our skyline space are

router	target IP	#packets	vol.
1	121.11.*.*	134	1226
1	110.1.*.*	60	72
2	121.11.*.*	180	1280
2	110.1.*.*	80	100
3	121.11.*.*	160	1301
4	201.7.*.*	627	4874
...	...	...	...

target IP	#packets	vol.
121.11.*.*	158	1269
110.1.*.*	70	86
201.7.*.*	627	4874
117.3.*.*	884	982
...	...	...

Aggregation (average)

target IP	#packets	Var(vol.)	skyline
121.11.*.*	158	1497	YES
110.1.*.*	70	392	NO
201.7.*.*	627	0	NO
117.3.*.*	884	1208	YES
...	...	...	...

Skyline space

Dimensions: #packets, Var(vol.)

Figure 3.1: Monitoring ISP network traffic skyline: (a) the raw-distributed data, (b) the aggregated data, (c) the skyline space.

defined through a d'-dimensional *function vector* $\boldsymbol{f} : \mathbb{R}^d \rightarrow \mathbb{R}^{d'}$, where each dimension $\boldsymbol{f}[k](\vec{v}(\cdot))$ is a possibly complex, non-linear, arbitrary function over the original d-dimensional global statistics vectors of our objects. We define the notion of *functional dominance* (or, $\boldsymbol{f}$*-dominance*) over fragmented data objects as follows. (Wlog., the definition assumes that lower values are preferred for the skyline.)

Definition 3.2.1 ($\boldsymbol{f}$-dominance)**.** *Let $\vec{v}(o_i)$, $\vec{v}(o_j)$ denote the global statistics vectors of objects o_i and o_j. We say that o_i* $\boldsymbol{f}$-dominates *o_j (denoted as $o_i \prec_{\boldsymbol{f}} o_j$) if and only if $\boldsymbol{f}[k](\vec{v}(o_i)) \leq \boldsymbol{f}[k](\vec{v}(o_j))$ for all $k = 1, \ldots, d'$, and $\exists k \in \{1, \ldots, d'\}$ such that $\boldsymbol{f}[k](\vec{v}(o_i)) < \boldsymbol{f}[k](\vec{v}(o_j))$.*

The $\boldsymbol{f}$*-skyline* of the set of objects $\mathcal{O} = \{o_1, \ldots, o_n\}$ fragmented over the remote sites $\mathcal{P}$ is then simply defined as the subset of objects in $\mathcal{O}$ that are not $\boldsymbol{f}$-dominated by any other object in $\mathcal{O}$. That is, $o \in \mathcal{O}$ belongs in the $\boldsymbol{f}$-skyline if and only if $\nexists o' \in \mathcal{O}$ such that $o' \prec_{\boldsymbol{f}} o$. Our goal is to continuously maintain (at the coordinator) the $\boldsymbol{f}$-skyline over a large collection of fragmented multi-dimensional objects $\mathcal{O}$ that are dynamically updated across multiple remote sites $\mathcal{P}$.

3.3 Requirements Analysis

In large-scale complex query monitoring, the size of the local data-stream distributions N can be huge, which means that supporting local data summarization techniques (for ensuring space/time efficiency and, also, improving communication costs) is critical. To address this challenge, we have incorporated appropriate *sketching* tools in the general geometric monitoring framework. Naturally, sketching also implies that local and global statistics vectors are only approximated to within some ϵ error bound. Further reductions in communication can be achieved through the use of *concise models* of the local data-stream dynamics. These models are constructed locally at each node, and can be communicated from the nodes to the central monitoring station (along with the local stream summaries) in an attempt to accurately predict the anticipated behavior of local data streams. Our initial results have clearly demonstrated the benefits of coupling the geometric method with sketch synopses (Papapetrou et al., 2012; Garofalakis et al., 2013) and prediction models (Giatrakos et al., 2012) for large-scale distributed-stream monitoring tasks. Furthermore, in a recent paper Papapetrou and Garofalakis (submitted), we have proposed novel algorithms for continuous fragmented skyline monitoring in large-scale distributed systems. Both algorithms rely on effectively *decomposing* the continuous fragmented skyline computation into *a collection of threshold-crossing queries*, which can be efficiently monitored at the participating sites using the geometric method. Another important concern here is scalability with respect to the number of distributed sites and fragmented objects being tracked. Our skyline tracking algorithms carefully optimize the number of threshold queries that need to be monitored, and, as our results demonstrate, can provide substantial communication benefits and scale to several thousands of objects/sites.

3.4 Employed Data Sets

We have been benchmarking the algorithms and tools we develop through extensive simulations on a number of real-life data sets, including publicly-available network-traffic data from the Internet Traffic Archive (`http://ita.ee.lbl.gov`) and the Internet2 Network Observatory (`http://www.internet2.edu/network/`). The HTTP data set is available from `http://ita.ee.lbl.gov/html/contrib/WorldCup.html` and contains HTTP requests made to the World Cup 1994 web site. A second data set we have used is the CRAWDAD data (available from `http://crawdad.cs.dartmouth.edu/meta.php?name=ibm/watson#N100AD`) containing SNMP records about network users such as number of packets and bytes from/to each user's machine. (The data has been collected from a corporate research center (IBM Watson) over a period of several weeks.) For our continuous fragmented skyline monitoring application, we have also employed two additional data sets: (1) WEATHER is a massive weather-related data set, with 93.6 million readings from 5423 sensors distributed in 257 countries, with the weather statistics per country determined as the average statistics read by all its sensors. (2) MOVIES is the largest of the MovieLens data sets, and is one of the standard data sets used in recommendation research. The data set contains 10 million ratings of 10681 movies provided by 71567 users, and is frequently used for evaluating recommender systems. (In our work, we employ these data sets to simulate scenarios where (weather or user-ratings) information needs to be aggregated across large collections of distributed sites in order to maintain a set of useful skylines on the observed streaming data.)

3.5 Results

As a small sample of our experimental results with real-life data, Figures 3.2-3.3 depict the communication cost savings when enriching the basic geometric method with AMS sketches (Alon et al., 1996) and simple linear prediction models. More specifically, Figure 3.2 compares the sketch-data transmission costs of the state-of-the-art **CG** technique of (Cormode and Garofalakis, 2008) for tracking self-join size queries using AMS sketches (whose cost is at point "1" of the y-axis) with our sketch-based generalizations of the geometric method (Garofalakis et al., 2013), for two different values of the sketching error ϵ. (The figure also depicts the performance of an unrealistic "oracle" strategy, showing that little improvement is possible over our techniques).

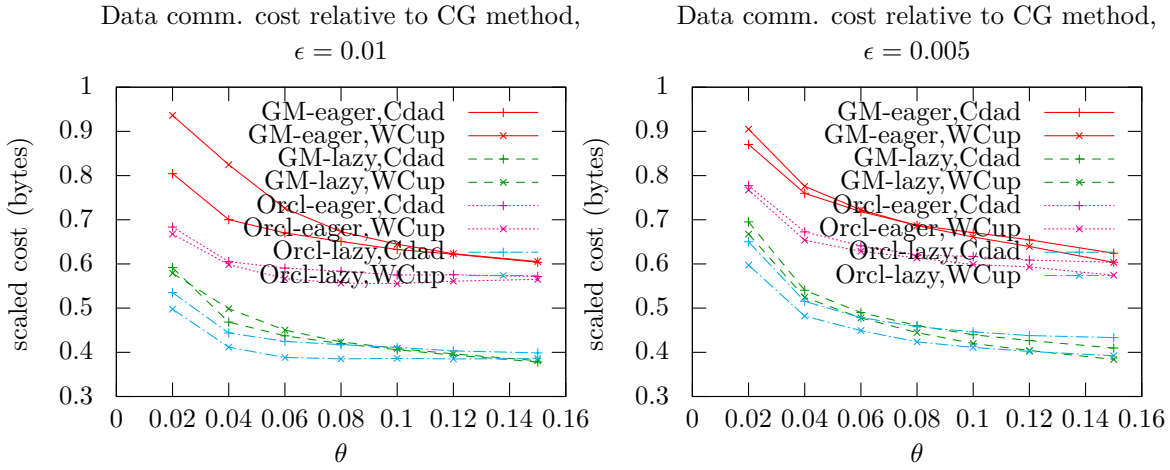


Figure 3.2: Self-join data communication costs as a fraction of the cost of **CG** (network traffic data).

Figure 3.3 shows the communication benefits of intelligently coupling prediction models with geometric monitoring ("CAA" curve) compared to the baseline geometric method ("Model 0"

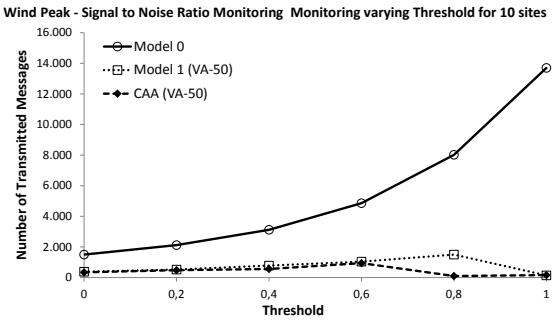


Figure 3.3: Number of messages as a function of the threshold for signal-to-noise ratio using prediction models (Weather data).

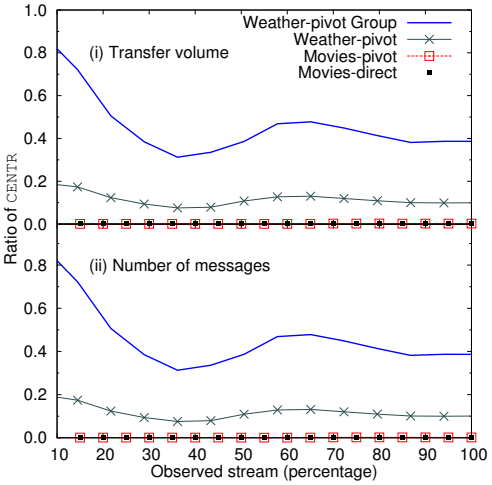


Figure 3.4: Communication cost savings of geometric skyline monitoring vs. "push-all-updates" (Weather and Movies data).

curve) (Giatrakos et al., 2012). Finally, Figure 3.4 depicts the communication benefits (in terms of both traffic volume and number of messages) of our novel geometric skyline monitoring techniques vs. a naive "push-all-updates" tracking approach.

For a full discussion of our experimental setup and results, please refer to the relevant papers Garofalakis et al. (2013); Giatrakos et al. (2012); Papapetrou and Garofalakis (submitted).

3.6 Discussion

The primary focus of our experimental studies is the *communication cost* incurred by our geometric monitoring techniques. We distinguish two parts in the total communication traffic. The first part, *data communication*, comprises updates transmitted from remote sites to the coordinator, whereas the second part, *monitoring overhead*, consists of additional control messages exchanged during local/global rebalancing operations. As our results have shown, in all cases, our geometric tracking techniques offer significant benefits compared to the naive strategy of centralizing all updates to the coordinator — the reduction in communication costs compared to centralization can often reach several orders of magnitude over a wide range of values for the number of sites/objects (e.g., Papapetrou and Garofalakis (submitted)). On the other hand, when compared to more clever techniques (e.g., the **CG** method of Cormode and Garofalakis (2008)), the monitoring overhead of the basic geometric approach can become a problem and pose limitations to the scalability of our techniques. A problem here is that, as the number of sites increases, opportunities for rebalancing among sites increase commensurably. The protocols of the basic geometric method exhaust every opportunity to ensure that a global violation (triggering update flushes) does not occur, without regard for the cost incurred by this rebalancing. Our experimental results indicate an important direction for further research, namely, attempt to capture (at least some of) the benefit in data communication with a truly scalable rebalancing approach.

Another important, and closely related, direction for research comes from the fact that the sheer scale of our target application environments implies that the conventional single-tier architecture (with a coordinator node directly linked to all remote sites) is no longer a realistic assumption. Extending geometric and safe-zone (SZ) monitoring to more general network architectures raises interesting issues. For instance, the underlying network (e.g., Internet or P2P overlay) architecture implies a natural, well-defined notion of "network locality" that could be exploited for reducing the amount of communication when communication does need to occur.

Abstractly, the idea is that, when a node discovers that its local SZs are violated, it should first attempt to resolve the situation within its network locality (i.e., the node's "neighborhood") before escalating communication exchanges further in the network — this, of course, would also lead to much more scalable rebalancing approaches. These are important issues that will need to be addressed when developing a practical ISQP architecture. The nature of the application will not ask for strict guarantees on correctness (false negatives), so in this application we can investigate probabilistic error guarantees. This is also enforced by the use of local summarization techniques that may give rise to some local probability of error.

Chapter 4

Sensor Networks

4.1 Motivation

Wireless sensor networks (WSNs) are an important application domain for the technology that LIFT is developing. WSNs are a collection of spatially distributed autonomous devices composed of sensors coupled with a broadcast mechanism, which cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants, at different locations. Some WSNs consist of thousands of sensors distributed over a large area - usually on the ground but also at sea.

WSNs play a major role in monitoring of both artificial and natural surroundings. Recently, efforts are being made to apply WSNS to tasks such as river flooding, tsunami, and earthquake warnings. These tasks require the system to very quickly identify imminent danger and report it. Typically, a WSN sends periodical updates to the coordinator, which studies the inputs and decides whether to issue an alert.

Warning from natural disasters is a very delicate problem: it is imperative to issue a warning, but on the other hand, false alarms can cause the needless evacuation of many people from their homes. For example, on November 16, 2006, a false tsunami alert caused the evacuation of hundreds of thousands of people in northern Japan. Our algorithms are rooted in a probabilistic framework, and the function we optimize is the probability of the local measurements to remain in their "safe zones", thus minimizing the number of false alarms, while conforming to the constraint that a real alarm will be reported and dealt with immediately.

4.2 Scenario Description and Formalization

In order to evaluate and demonstrate the feasibility of using geometric monitoring (GM) and the more efficient safe zone (SZ) approach, we have considered several interesting combinations of sensor node organizations, monitored functions and used real data sets. In all of our subscenarios, we consider sensor nodes that collect measurements about their environment, such as air pollutant measurements, light, temperature, humidity and solar irradiation measurements etc. More specifically:

1. In our first scenario we seek to evaluate the scalability and efficiency of calculating safe zones for real sensor network applications, as well as the effectiveness in reducing the number of violations (and, thus, also reduce the number of transmitted messages) using safe zones, rather than the simpler GM approach. In this first scenario we will investigate a large variety of monitoring functions, such as ratios or chi-square monitoring.

2. In our second scenario we consider the problem of detecting nodes with outlier readings in sensor networks. In this scenario we seek to use the GM and SZ approach in order to handle complex functions used for testing the similarity of nodes, such as the L_∞, L_1, L_2 norms, the

Similarity Metric	Function based on vectors X, Y of two nodes	Transforming the Function Calculation into a Function over Average Quantities that each node may compute individually	Local Statistic Vectors X', Y' (may contain more elements than X, Y)	Value of function on any point Z
$\|\| X - Y \|\|_\infty$	$dist(X,Y) = max_i \|X_i - Y_i\|$	$2\, max_i\left(\left\|\frac{X_i - Y_i}{2}\right\|\right)$	$X' = \begin{bmatrix} X_1 \\ \dots \\ X_W \end{bmatrix}$ $Y' = \begin{bmatrix} -Y_1 \\ \dots \\ -Y_W \end{bmatrix}$	$2\, max_i(\|Z_i\|)$
$\|\| X - Y \|\|_1$	$dist(X,Y) = \sum_{i=1}^{W} \|X_i - Y_i\|$	$2\sum_{i=1}^{W}\left\|\frac{X_i - Y_i}{2}\right\|$	$X' = \begin{bmatrix} X_1 \\ \dots \\ X_W \end{bmatrix}$ $Y' = \begin{bmatrix} -Y_1 \\ \dots \\ -Y_W \end{bmatrix}$	$2\sum_{i=1}^{W}\|Z_i\|$
$\|\| X - Y \|\|_2$	$dist(X,Y) = \sqrt{\sum_{i=1}^{W}(X_i - Y_i)^2}$	$\sqrt{4\sum_{i=1}^{W}\left(\frac{X_i - Y_i}{2}\right)^2}$	$X' = \begin{bmatrix} X_1 \\ \dots \\ X_W \end{bmatrix}$ $Y' = \begin{bmatrix} -Y_1 \\ \dots \\ -Y_W \end{bmatrix}$	$2\sqrt{\sum_{i=1}^{W} Z_i^2}$
$\|\| X - Y \|\|_k$	$dist(X,Y) = \sqrt[k]{\sum_{i=1}^{W}(\|X_i - Y_i\|)^k}$	$\sqrt[k]{2^k\sum_{i=1}^{W}\left(\left\|\frac{X_i - Y_i}{2}\right\|\right)^k}$	$X' = \begin{bmatrix} X_1 \\ \dots \\ X_W \end{bmatrix}$ $Y' = \begin{bmatrix} -Y_1 \\ \dots \\ -Y_W \end{bmatrix}$	$2\sqrt[k]{\sum_{i=1}^{W}\|Z_i\|^k}$
Cosine Similarity	$cos(\theta(X,Y)) = \frac{XY}{\|X\|_2\,\|Y\|_2}$	$\frac{2\sum_{i=1}^{W}\left(\frac{X_i + Y_i}{2}\right)^2 - \frac{\|X\|_2^2 + \|Y\|_2^2}{2}}{2\left(\frac{\|X\|_2 + \|Y\|_2}{2}\right)^2 - \frac{\|X\|_2^2 + \|Y\|_2^2}{2}}$	$X' = \begin{bmatrix} X_1 \\ \dots \\ X_W \\ \|X\|_2^2 \\ \|X\|_2 \end{bmatrix}$ $Y' = \begin{bmatrix} Y_1 \\ \dots \\ Y_W \\ \|Y\|_2^2 \\ \|Y\|_2 \end{bmatrix}$	$\frac{2\sum_{i=1}^{W} Z_i^2 - Z_{W+1}}{2\,Z_{W+2}^2 - Z_{W+1}}$
Extended Jaccard Coefficient	$J(X,Y) = \frac{XY}{\|X\|_2^2 + \|Y\|_2^2 - XY}$	$\frac{2\sum_{i=1}^{W}\left(\frac{X_i + Y_i}{2}\right)^2 - \frac{\|X\|_2^2 + \|Y\|_2^2}{2}}{2\frac{\|X\|_2^2 + \|Y\|_2^2}{2} - \left(2\sum_{i=1}^{W}\left(\frac{X_i + Y_i}{2}\right)^2 - \frac{\|X\|_2^2 + \|Y\|_2^2}{2}\right)}$	$X' = \begin{bmatrix} X_1 \\ \dots \\ X_W \\ \|X\|_2^2 \end{bmatrix}$ $Y' = \begin{bmatrix} Y_1 \\ \dots \\ Y_W \\ \|Y\|_2^2 \end{bmatrix}$	$\frac{2\sum_{i=1}^{W} Z_i^2 - Z_{W+1}}{2\,Z_{W+1} - (2\sum_{i=1}^{W} Z_i^2 - Z_{W+1})}$
Correlation Coefficient	$corr(X,Y) = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$	$\frac{\left(\frac{2}{W}\sum_{i=1}^{W}(\frac{X_i + Y_i}{2})^2 - \frac{1}{W}\frac{\|X\|_2^2 + \|Y\|_2^2}{2}\right) - \left(2\left(\frac{E[X] + E[Y]}{2}\right)^2 - \frac{(E[X])^2 + (E[Y])^2}{2}\right)}{2\left(\frac{\sigma_X + \sigma_Y}{2}\right)^2 - \frac{\sigma_X^2 + \sigma_Y^2}{2}}$	$X' = \begin{bmatrix} X_1 \\ \dots \\ X_W \\ \|X\|_2^2 \\ E[X] \\ (E[X])^2 \\ \sigma_X \\ \sigma_X^2 \end{bmatrix}$ $Y' = \begin{bmatrix} Y_1 \\ \dots \\ Y_W \\ \|Y\|_2^2 \\ E[Y] \\ (E[Y])^2 \\ \sigma_Y \\ \sigma_Y^2 \end{bmatrix}$	$\frac{\left(\frac{2}{W}\sum_{i=1}^{W} Z_i^2 - \frac{1}{W} Z_{W+1}\right) - (2Z_{W+2}^2 - Z_{W+3})}{2Z_{W+4}^2 - Z_{W+5}}$

Figure 4.1: Expressing common similarity functions between two nodes with W-dimensional measurement vectors X and Y using the geometric method. The function must be expressed as a general function over the average of local statistics vectors X' and Y', with the elements of X' (Y') being computed based only on the elements of X (Y). X' (Y') may have a different dimensionality than X (Y).

cosine similarity and the Extended Jaccard Coefficient. The definition of these functions, as well as the way that they are transformed so that they can be expressed using the GM and SZ approach are depicted in Figure 4.1. The sensor nodes are asked to continuously monitor whether their similarity to neighboring nodes lies above/below a given threshold. Extensions to monitoring minimum support queries (i.e. when a node has fewer than minSupp similar neighbors, which may point to a node that is either malfunctioning or has started monitoring an interesting event) are also going to be studied.

3. In our third scenario we well consider the case of hierarchically organizing the sensor nodes, as is the case in large-scale sensor networks. In this scenario we seek to adapt the GM and SZ approaches and optimize them in order to take advantage of this hierarchical organization of sensor nodes. Several complex monitoring functions used in the first two scenarios will be evaluated.

4.3 Requirements Analysis

Bottleneck and Limitations. Typically, the major algorithmic challenge in WSNs is to minimize the communication overhead. To make sensors efficient, their power source - typically a battery - should function for a long time (even years). Replacing or recharging the batteries is a timely and expensive procedure. Moreover, sensors are sometime scattered in remote areas (e.g., dropped from airplanes) and when the batteries run out, the only solution is to deploy a new WSN. The most important factor of energy drain in WSNs is the energy spent during data communication. Since communication minimization is a major goal of LIFT, WSNs are an important application domain.

Used LIFT techniques. In this scenario we plan to investigate the use of both the geometric monitoring (GM), as well as the more advanced Safe Zone (SZ) approach. Our goal is to allow

the efficient (in terms of the number of required transmissions) monitoring of complex functions over the sensor data, while at the same time minimizing the number of false alarms, while also maintaining correctness.

Software/Hardware Requirements. The three scenarios are going to be evaluated with extensive simulations that use several real data sets. In addition, we plan to demonstrate the efficiency of our approaches in the second scenario with a real implementation on Iris sensor nodes.

4.4 Used Data Sets

In our three Sensor Networks scenarios we plan to utilize a variety of real data sets. We have currently used the following three data sets:

1. Air pollutant measurements taken from "AirBase - The European air quality database"[1]. Concentrations were measured in micrograms per cubic meter. Nodes correspond to sensors at different geographical locations. The data at different nodes greatly varies in size and shape and is highly irregular as a function of time; see Fig. 4.2.

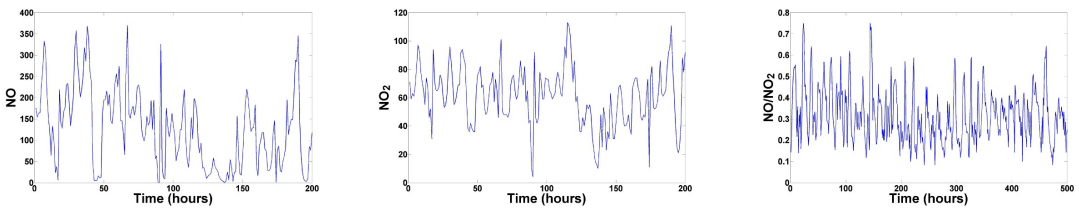


Figure 4.2: Left and center: typical local concentrations of NO and NO2 as a function of time. Right: typical behavior of the local ratio between NO and NO2 as a function of time.

2. The Intel Lab data set includes temperature, humidity and light measurements collected by 54 motes in the Intel Research, Berkeley Lab between February 28th and April 5th, 2004. The measurements are collected approximately every 30 seconds. [2].

3. The Weather Data, includes air temperature, relative humidity and solar irradiance measurements, collected on a minute base, from the station in the University of Washington and for the year 2002[3].

We are also making efforts to gain access to more real data sets, such as real tsunami data.

4.5 Results

4.5.1 Results - Scenario 1

In our initial set of experiments, we monitored the ratio between two pollutants, NO and NO2, measured in distinct sensors. Formally, each of the n node holds a vector (x_i, y_i) (the two concentrations), and the monitored function is $\frac{\sum y_i}{\sum x_i}$ (in Gupta et al. (2010) ratio is monitored but over aggregates, while here the monitoring is of the instantaneous ratio). An alert must be sent whenever this function is above a threshold T, and also when the average concentration of NO2 is above 250 (as that also indicates poor air quality). We also measured the improvements in running that the hierarchical clustering of nodes provides during the safe zone computation. In our second set of experiments we explored the bandwidth savings when monitoring the chi-square distance between

[1]The European air quality database: http://dataservice.eea.europa.eu/dataservice/
[2]Data available at: http://db.csail.mit.edu/labdata/labdata.html
[3]Data available at: http://wwwk12.atmos.washington.edu/k12/grayskies

histograms, which is a non-linear and non-monotonic function, defined by $\chi(f,g) = \sum \frac{(f_i - g_i)^2}{f_i + g_i}$ for histograms f, g. The histogram was defined as the concentration levels of five pollutants, and the monitored function was the chi-square distance between the hourly average of two nodes and their average calculated over the previous week (i.e., a measure of how much the hourly distribution deviates from last week's average).

The main findings of our experimental study on this scenario can be summarized as follows:

- The use of safe zones (of triangular shape in the ratio monitoring case) can dramatically reduce the number of violations, compared to GM monitoring. Figure 4.3 demonstrated that on average a 17.5 fold reduction is achieved when varying the number of nodes from 1 to 10.
- Hierarchical clustering of nodes can reduce the running time of computing the optimal safe zones by up to two orders of magnitude, compared to a flat (non-hierarchical) optimization, as Figure 4.4 demonstrates.
- The reduction in the number of violations for the chi-square monitoring function using the safe zone approach compared to geometric monitoring is significant, resulting in up to a 60-fold reduction (Figure 4.5).

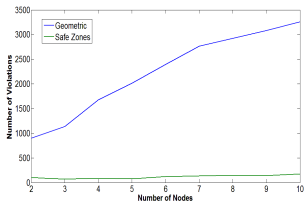


Figure 4.3: Comparison of safe zones (green line) to GM (blue line) in terms of the number of violations, varying the number of nodes.

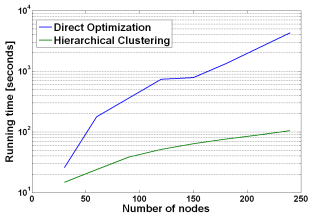


Figure 4.4: Running time (in logarithmic scale) for "flat" – direct optimization over all the nodes (blue) vs. hierarchical clustering (green).

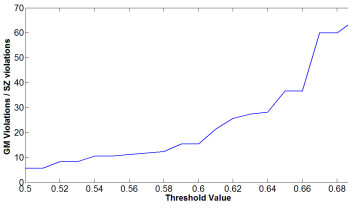


Figure 4.5: Comparing the number of violations between GM and safe zones of 5-dimensional axis-aligned boxes.

The detailed results can be found in (Keren et al., submitted).

4.5.2 Results - Scenario 2

For Scenario 2 we developed techniques for monitoring complex similarity functions between nodes. The full description of the used techniques, as well as a detailed experimental evaluation of several different setups related to Scenario 2, can be found in Burdakis and Deligiannakis (2012). The basic idea is to appropriately transform the similarity functions and express them in a format that is expressed as the average of carefully selected local estimate vectors maintained at the sensor nodes. In this section we present some of these results.

Main Findings. In our experiments we use the real Intel Labs and Weather data sets. The main findings of our experimental evaluation in this scenario can be summarized as follows:

- The use of LIFT techniques can offer a one-order of magnitude reduction in the number of transmitted messages. The improvements are even more significant when exploiting the broadcast capabilities of sensor nodes.
- The improvements are significant for all monitoring functions and for a wide variety of different settings (used data set, size of the network, number of neighbors for each node etc).
- The use of safe zones can further reduce the communication cost, compared to the geometric monitoring technique.

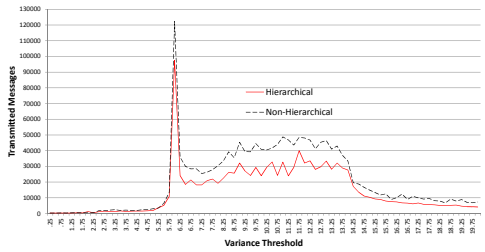


Figure 4.6: Comparison of hierarchical vs non-hierarchical operation of safe zone monitoring of variance. Depicting total number of transmitted messages when varying the threshold value. Intel Labs data set, 100 nodes, 1000 epochs.

4.5.3 Results - Scenario 3

We also tested the performance of the LIFT techniques when adapted to perform under a hierarchical setting. Sensor nodes are not expected to be able to directly communicate with a coordinator node, due to a limited transmission range. Our algorithms are based on the following principle: when a node detects a local violation, it pushes its delta vector to its neighbors in the hierarchical routing tree that is formed in the sensor network for propagating the results. This technique offers the hope that violations may be balanced at other nodes in the hierarchy, based on the data that other nodes possess. We devised a hierarchical processing of sensor nodes that operates in two waves. In the first wave any local violations are pushed to the children of the violating node. In the second wave any remaining local violations are pushed to the parent of the violating node. If a violation reaches the root of the hierarchy, then a synchronization needs to take place.

We compared our new hierarchical organization of sensor nodes, where each node acts as a mini coordinator for its children nodes, with the original LIFT approach, where local violations would have to be transmitted and resolved at the coordinator node through multihop communication. We tested our techniques in the Intel Labs data set and at a network of 100 nodes. Figure 4.6 demonstrates that the non-hierarchical approach may require substantially more transmitted messages (up to 50% in some cases) when compared to our novel hierarchical processing algorithms.

4.6 Discussion

We applied LIFT techniques in a variety of scenarios for sensor network applications. When developing our techniques, we set three important goals for our techniques:

- Increased bandwidth savings for sensor nodes, which ultimately lead to increased lifetime. This was achieved in all cases, using the LIFT approach and our newly developed techniques for our scenarios.

- Feasibility of our techniques for the (reduced) computational capabilities of sensor nodes. To achieve this goal, we needed to devise techniques in which sensor nodes could easily perform their local monitoring tasks and can easily compute new safe zones upon a global violation. Safe zones provide us with the appropriate tool for the first requirement, since the test for local violations that each sensor node needs to perform is very simple. On the other hand, the geometric approach would not be feasible to use in a sensor network application, since it would require the sensor nodes to compute the minimum and maximum values of a complex monitoring function in a ball (or ellipsis). For the second requirement, the hierarchical clustering of the sensor nodes allowed the efficient *centralized* computation of new safe zones and the transmission of the safe zones to each node. This techniques allows the sensor nodes to refrain from this computationally expensive task.

- Ease of use in sensor network applications. Sensor nodes are often programmed to transmit in certain time periods, while sleeping in the majority of time. The majority of past work has devised transmissions schedules that follow the tree hierarchy (bottom-up from the leaves to the root, or top-down). In our work, we took this requirement into consideration when designing our hierarchical organization algorithm.

Our results demonstrate that LIFT techniques can be applied for a large variety of monitoring and outlier detection tasks in sensor networks, while minimizing the amount of CPU time required by the sensor nodes in order to test for local violations. The hierarchical clustering safe zone computation approach offers a scalable technique for computing safe zones in large scale networks. Moreover, the hierarchical clustering of sensor nodes can offer further significant bandwidth savings, compared to the distributed approach.

Chapter 5

Cloud Data Centers

5.1 Motivation

Large scale cloud services run on top of data centers comprising thousands of computers. It is unreasonable to assume that so many machines are working properly and are well configured. Unnoticed faults might accumulate to the point where redundancy and fail-over mechanisms break. Therefore, early detection and handling of *latent faults* is essential for preventing failures and increasing the reliability of these services.

Machines are usually monitored by collecting and analyzing performance counters (Bodík et al., 2010; Isard, 2007; Sahoo et al., 2003). Hundreds of counters per machine are reported by the various service layers, from service-specific information (such as the number of queries for a database) to general information (such as CPU usage). The large number of machines and counters in data centers makes manual monitoring impractical.

Existing automated techniques suffer from several key issues. Most are inflexible: they rely on static rules, past examples or deep domain insights. When the system or workload changes, new rules, labeled examples or insights are required. Existing techniques are also reactive, identifying failures rather than proactively look for them. Finally, most existing approaches require centralization of data. For very large data centers, this may be impossible, as the data is too large to centralize and process in one location.

In our previous work (Gabel et al., 2012) we provide evidence that latent faults are common. We show that these faults can be detected using domain independent techniques, and with high precision. Our general framework can adapt to different situations using "pluggable" tests, and guarantee the false positive rate. The general scenario and approach are described in detail in that work.

This current work (Gabel et al., 2013) extends our previous work using LIFT machinery to overcome the challenge of the centralizing and processing the huge amount of data generated by monitored nodes. Our previous work relied on a parallel "Map-Reduce" style architecture to process all the data, which amounted to 12TB per day – far too big to centralize. Parallel processing may not always be feasible in all situations, however. In this scenario we wish to apply LIFT-style techniques to greatly reduce the amount of data that needs to be sent. New sketch-based tests fit well within the existing framework, and can reduce the amount of data by an order of magnitude or more. We use the safe zone approach (Keren et al., 2011) to monitor both the global mean and the global variance of each counter so that they do not deviate too much from their last known values.

5.2 Scenario Description and Formalization

Both the general framework and the communication-efficient adaptation are described in detail in (Gabel et al., 2012, 2013), part of the project deliverables. This section is a summary.

$\mathcal{M}$ denotes the set of all machines in a test, m, m', m^* denote specific machines, and $M = |\mathcal{M}|$ denotes the number of machines. $\mathcal{C}$ is the set of all counters selected by the preprocessing algorithm, c denotes a specific counter, and $C = |\mathcal{C}|$. $\mathcal{T}$ are the time points where counters are sampled during preprocessing (for instance, every 5 minutes for 24 hours), t, t' denote specific time points, and $T = |\mathcal{T}|$.

Thus each machine outputs C counters every time point. We denote the *vector of counter values* for machine m at time t as $x(m,t)$. In LIFT terms, $x(m,t)$ is the *local statistics vector* for node m at time t. Each counter is identically distributed across machines: we assume that $x(m,t)$ is a realization of a random variable $X(t)$ whenever machine m is working properly. Given a test S, at any point t, the input $x(t)$ to the test S consists of the vectors $x(m,t)$ for all machines m. The test $S(m, x(t))$ analyzes the data and assigns a *score* (either a scalar or a vector) to machine m at time t. x and x' denote sets of inputs $x(m,t)$ and $x'(m,t)$, respectively, for all m and t.

One difference from the usual LIFT setting is that in our case, rather than monitoring a single function of the global aggregate and a threshold, here we monitor one function per node, with differing threshold. Despite this difference, we can use the sketch approach to achieve significant savings in communication and processing costs.

Given a test S, and a significance level $\alpha > 0$ that determines the false positive rate, we can present the framework as follows:

1. Preprocess the data to automatically select useful counters and scale to unit variance (can be done once, after collecting some data).
2. Compute for every machine m the vector $v_m = \frac{1}{T}\sum_t S(m, x(t))$ (integration phase). In LIFT terms, this is the *global aggregate* for node m.
3. Using the vectors v_m, compute p-values $p(m)$ for every machine.
4. Report every machine with $p(m) < \alpha$ as suspicious.

The p-value for a machine m is a bound on the probability that a random healthy machine would exhibit such aberrant counter values. If the p-value falls below a predefined significance level α, the null hypothesis is rejected, and the machine is flagged as suspicious.

In the context of the general framework in our previous work (Gabel et al., 2012), we derived and evaluated 3 different tests within the framework (different S functions). The sign test accumulates the average normalized direction from machine m to the rest of the machines. The Tukey test measures the average depth of $x(m,t)$ compared to the vectors of other machines at the same time. The LOF test similarly compares the local density of points around $x(m,t)$ to the local density of its neighbors.

To put this in LIFT terms, the *monitored function* for node m in the sign test would be:

$$g_m = (M+1)\exp\left(\frac{-TM\gamma^2}{2\left(\sqrt{M}+2\right)^2}\right) \quad ,$$

where $\gamma = \max\left(0, \|v_m\| - \mathrm{E}_{m\in\mathcal{M}}\left[\|v_m\|\right]\right)$, v_m defined as above, and

$$S\left(m, x(t)\right) = \frac{1}{M-1}\sum_{m'\neq m}\frac{x(m,t) - x\left(m',t\right)}{\|x(m,t) - x\left(m',t\right)\|} \quad .$$

5.2.1 Sketches

Rather than applying test S to the set of all local statistic vectors $x(m,t)$, each machine m will first apply a sketching function f to its vectors, and send only the sketch $\hat{x} = f(x(m,t))$. The modified test $\hat{S}$ will be applied to the sketches rather than the original vector: $v_m = \frac{1}{T}\sum_t \hat{S}(m, \hat{x}(t))$.

One well-suited sketch is the AMS sketch (Alon et al., 1999), which involves a random linear projection to k dimensions. In our setting, each machine would project its counter vectors to k dimensions using a specially constructed projection matrix: $\hat{x}(m,t) = f(x(m,t)) = Rx(m,t)$ where R is a random $C \times k$ matrix constructed as described in (Alon et al., 1999).

Online Sign Test on Linear Sketches. The sign test function depends only on the normalized direction from $x(m,t)$ to the other vectors. Since the sign-test uses normalized directions and the projection matrix R preserves their symmetry around $x(m,t)$, we can apply the sign test directly to the sketched vectors $\hat{x}(m,t)$. Moreover, the sign test p-value does not depend on the dimensionality of the vectors, and so we can use it as is.

The integration phase in stage 2 of the framework computes $v_m = \frac{1}{T}\sum_t S\left(m, x(t)\right)$. Computing $S\left(m, x(t)\right)$ only requires the data from time t. When new data arrives at time t, the coordinator updates the current v_m by computing and adding $\frac{1}{T}S\left(m, \hat{x}(t)\right)$, and subtracting the least recent stored test result, $\frac{1}{T}S\left(m, \hat{x}(t-T-1)\right)$. The p-value for each machine in the time window can then be computed in the usual manner. Since the test function S need only be computed for the most recent time, and since the sketches are of low dimension k, processing and memory costs are low. This allows the computation to be done on a single coordinator machine on time, before the next round starts. More details can be found in (Gabel et al., 2013).

5.2.2 Scaling By Monitoring Variance with Safe Zones

We use the *safe zones* approach (Keren et al., 2011; Sharfman et al., 2007) to monitor both the global mean and the global variance of each counter. Given the last known global mean and variance of the last T samples, we define some lower and upper threshold, for example 0.5 and 2 times the last known values. If there is any violation, the coordinator polls each node for the current mean and variance, and distributes the new global mean and variance to all nodes. We can trade-off accuracy and communication by adjusting the high and low thresholds when monitoring. Violations are less likely if global values are allowed to drift further from their last known values – reducing communication but also decreasing accuracy. We monitor each counter independently, so it is enough to show how we monitor a single counter X. Further note that all tests described in (Gabel et al., 2012) are invariant to data translation, and so we do not monitor the global mean explicitly. The full details of the approach can be found in (Gabel et al., 2013).

Notation. The set of values of counter X over the last T times and over M nodes (machines) is denoted by $X(t)$. We denote by $X_i(t)$ the values of X at node i for the last T times up to t. Thus $\mathrm{E}[X_i(t)]$ is the mean of the last T values at node i in time t, while $\mathrm{E}[X(t)]$ is the global mean of the last values at all nodes. Denote $\mu_i(t) = \mathrm{E}[X_i(t)]$ the local means, and $\mu(t) = \mathrm{E}[X(t)]$ the global mean. Similarly, we denote $\lambda_i = \mathrm{E}\left[X_i(t)^2\right]$, the local mean of the squares, and $\lambda = \mathrm{E}\left[X(t)^2\right]$ the global mean. Let $V(t) = (\mu(t), \lambda(t))$, and $V_i = (\mu_i(t), \lambda_i(t))$, the global and local monitored vectors, respectively.

Monitoring. We wish to monitor the global variance $\mathrm{Var}(X)$ at each time t. Recall that:

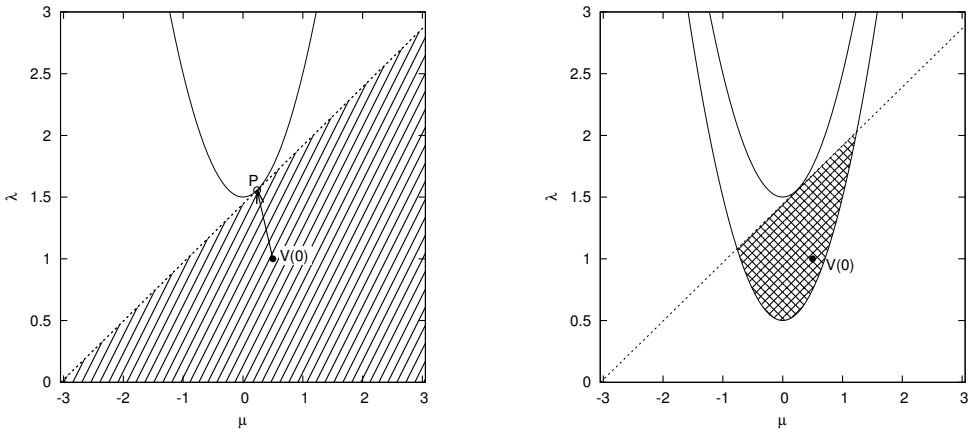
$$\mathrm{Var}(X) = \mathrm{E}\left[X^2\right] - \left(\mathrm{E}\left[X\right]\right)^2 = \lambda - \mu^2 \quad .$$

We therefore monitor the conditions $L \leq \lambda - \mu^2 \leq H$, for some lower and upper variance thresholds L and H. We aim to find a convex safe zone G which is contained within the admissible region. Since convex sets are closed under averaging, when all local vectors are inside the safe zone, the global mean is guaranteed to be inside as well.

Let $t = 0$ be the last global synchronization time, and let $V(0) = (\mu(0), \lambda(0))$ be the *reference point*, the last known global mean and mean-of-squares, computed that time. For each node i we define the local drift vector $d_i(t)$ as the drift of the current vector from the node's vector during the last synchronization: $d_i(t) = V_i(t) - V_i(0)$.

Since we wish to monitor that the global $V(t)$ is within some convex set G, we define equivalent local conditions on the drift vectors. The current local vectors can be written in terms of drift vector d_i: $V_i(t) = V_i(0) + d_i(t)$. Note that the global vector is the mean of the local vectors, and can thus be written as the mean of drifts and the reference point:

$$V(t) = \frac{1}{M}\sum_i V_i(t) = V(0) + \frac{1}{M}\sum_i d_i(t) \quad . \tag{5.1}$$



(a) Upper threshold.

(b) Upper and lower.

Figure 5.1: Safe zones for $L = 0.5, H = 1.5$ where $V(0) = (0.5, 1)$.

Let $W_i(t) = V(0) + d_i(t)$ be the local drift from the last reference point. Note that $V(t) = \frac{1}{M}\sum_i W_i$, recall G is convex, and from (5.1) we arrive at the local conditions: if $\forall i, W_i \in G$ then $V(t) \in G$.

To monitor that the variance is between L and H, we derive separate safe zones: one for variance above L and another for variance below H. As long as the local conditions for both safe zones are maintained in all nodes, we are guaranteed that the variance is within the allowed range.

Variance Above Lower Threshold. We wish to define a convex safe zone G_L so that as long as $V(t) \in G_L$ then $\mathrm{Var}(X) \geq L$. This corresponds to monitoring that $\lambda - \mu^2 \geq L$, which is already a convex set – the area above a parabola – and can be directly used as safe zone. Therefore the local condition for each node i is trivial: $I_i(t) \in G_L$: $I_i(t) = V(0) + d_i(t) = (a, b)$ and monitor that $b - a^2 \geq L$.

Variance Below Upper Threshold We wish to define a convex safe zone G_H so that as long as $V(t) \in G_H$ then $\mathrm{Var}(X) \leq H$. This area is the area below a parabola, which is not a convex set. However, we can find a tangent half-plane I below this parabola. This half-plane is a convex set, and since $I \subset G_H$, then as long as $V(t) \in I$, $V(t) \in G_H$ and therefore $\mathrm{Var}(X) \leq H$.

We use the reference point $V(0)$ to find the optimal hyperplane. The thresholds H and L are reset during synchronization, so obviously $V(0) \in G_H$. We can choose any half-space I such that $V(0) \in I$, but to avoid future unnecessary synchronization we choose I such that $V(0)$ is far from the boundary of G_H. Doing so ensures that drift has to be large to cause a violation. Consequently, we choose I as the tangent at point P, where P is the closest point to $V(0)$ on the parabola $\lambda - \mu^2 = H$, and the local condition is $W_i \in I$. We can find P numerically, or by minimizing the distance from the parabola to $V(0)$. For example, if $V(0) = (0.5, 1)$ and $H = 1.5$, then the closest point on the parabola, where $\frac{dD}{d\mu} = 0$, is $\mu \approx 0.237$. This yields the point $P = (0.237, 1.556)$, and finally the induced safe zone I: the half-plane $\lambda - 0.474\mu < 1.443$. Figure 5.1(a) shows $V(0)$, P and the resulting safe zone, and Figure 5.1(b) shows the intersection with the safe zone for the lower limit $L = 0.5$.

Threshold Selection. During each synchronization we set the new safe zone around the reference point. Large safe zones mean less communication, but they also allow the true variance to deviate further from the last global estimate, increasing the chance of errors when applying the test function S to the scaled and sketched data.

Let σ_0^2 be the last known global variance, given the current reference point $V(0)$: $\sigma_0^2 = \mathrm{Var}(X(0)) = \lambda(0) - \mu(0)^2$, and denote the current global variance $\sigma = \mathrm{Var}(X) = \lambda - \mu^2$. We set the the lower threshold to $L = \frac{\sigma_0^2}{f^2}$ and the upper threshold to $H = f^2\sigma_0^2$, where $f > 1$ is a

constant that determines the allowed deviation. These thresholds ensure that $\frac{\sigma_0^2}{f^2} < \sigma^2 < f^2\sigma_0^2$ as long as there is no safe zone violation, and therefore the standard deviation σ used to scale the counter is bounded by $\frac{\sigma_0}{f} < \sigma < f\sigma_0$.

Handling Violations. If one of the local conditions $W_j \in G$ is violated, it may be because $\mathrm{Var}(X)$ is no longer in the range, or due to a false alarm. The simplest way to deal with a violation is to perform a global synchronization: each node sends its current $V_i(t)$ to the coordinator. The coordinator "resets the time" to $t = 0$, computes the new global reference point $V(0)$, and sends it to the nodes, where it is used for monitoring and scaling. These synchronizations also also improve the accuracy of scaling, since nodes have fresh global mean and variance.

There are safe zone techniques that allow partial synchronization for further communication reduction, for example by predicting changes in the global reference point, or by balancing a node with local violation with another node that has enough slack. In the next section we explore approaches to preventing violations.

5.2.3 Preventing Safe Zone Violations

We describe 3 types of violations possible in the monitoring algorithm: true violations, global violations and local violations. A true violation is when the global aggregate is outside the admissible region (variance larger than H or smaller than L), and therefore synchronization must be performed. A global violation happens when the global aggregate is outside the safe zone, but still inside the admissible region (recall that the safe zone is a convex subset of the admissible region). In this case, synchronization is unavoidable but is not strictly necessary. A local violation is when the global aggregate is inside the safe zone, but the vectors of some of the nodes are outside their local safe zone. Ideally, we would like to avoid these violations since they trigger unnecessary additional synchronizations which increase communications. We explore two approaches to reduce the amount of spurious safe zone violations.

Local Slack. The safe zones described in Section 5.2.2 are uniform – all nodes share the same safe zones. One well known way to reduce violations would be to adjust the safe zone of each node to match its local statistics. Naturally, we must still preserve the guarantee that if all local node vectors are in their respective safe zones, then the global aggregate is inside the admissible region.

In our case, we elected to assign local safe zones to each node. The local safe zones are simply scaled versions of the original uniform safe zone G, centered as before around the reference point. Each node has a local slack factor β_i. When testing for local violation, node i first scales its drift vector d_i by β_i before checking for violations: $W_i(t) = V(0) + \frac{1}{\beta_i} d_i(t)$. This effectively scales the safe zone by $\frac{1}{\beta_i}$. The global drift is now a weighted mean of the local drift vectors, rather than a simple arithmetic mean. By making sure local slacks always sum to M we preserve our global guarantee: the convex hull of local drifts is still inside the weighted Minkowski mean, and therefore the global aggregate is inside the safe zone G.

Each node begins with a slack of $\beta_i = 1.0$. At each round, let $\mathcal{K}$ be the set of nodes that reported a local violation to the coordinator this round. If $|\mathcal{K}| \geq \frac{M}{2}$, we assume that current slacks and reference point are inadequate, and reset all slacks to 1.0. Otherwise, all nodes that reported a violation have their slack increased by some constant factor w: $\forall i \in \mathcal{K} : \beta_i \leftarrow w\beta_i$. This extra slack is then balanced across all non-violating nodes: $\forall i \in \mathcal{M} \setminus \mathcal{K} : \beta_i \leftarrow \gamma\beta_i$, where

$$\gamma = \left(M - \frac{\sum_{i \in \mathcal{K}} \beta_i}{\sum_{i \in \mathcal{M} \setminus \mathcal{K}} \beta_i} \right) \quad .$$

Finally, the coordinator distributes the new slacks to all nodes.

Our slack allocation scheme is motivated by several observations. First, we observe that in our setting, counters of healthy nodes are assumed to be identically distributed, and therefore their direction from the reference point is random. Simply translating the safe zone does not reduce

local violations. This phenomenon was also observed in practice during preliminary experiments. Second, we do assume there is a small number outlier nodes, whose counters might cause frequent local violations. Over time our balancing scheme distributes more slack (meaning larger safe zones) to such outlier nodes, balanced across the greater number of healthy nodes. Finally, this scheme entails sending just one extra value to each node, meaning it has the smallest impact on communication.

Reference Point Prediction. Section 5.2.2 describes a static scheme: nodes always use the last known global reference point $V(0)$ to monitor and scale counters. If nodes are able to correctly predict how the reference point changes over time, they can adjust the safe zone accordingly, thus reducing safe zone violations and communication costs. The prediction model must ensure that all nodes make the same prediction, which is used to monitor and scale the counters.

We incorporate a simple linear prediction model. Given current reference point $V(0)$, let Δt be the time difference between the last global synchronization ($t = 0$) and the previous synchronization, and let $V(-\Delta t)$ be the reference point at that time. The predicted reference point at current time t is simply:

$$V'(0) = V(0) + t\frac{V(0) - V(-\Delta t)}{\Delta t} \quad .$$

Once $V'(0)$ is known, each node can calculate the resultant safe zone as described in Section 5.2.2.

Observe that each node can remember Δt, $V(0)$ and $V(-\Delta t)$ during normal operation, and so this model requires no extra communication. Moreover, the model only depends on global data made available during synchronization, and so it yields the same prediction all nodes. Finally, this geometric model can result in "negative variance": $\mu'(0)^2 > \lambda'(0)$. In such cases we set $\lambda'(0) = \mu'(0)^2 + \epsilon$, where ϵ is some small constant.

One downside of this method is that it can impact accuracy. Since we allow the predicted reference point and the resulting safe zone to move outside the original safe zone, we are no longer guaranteed that $\frac{\sigma_0}{f} < \sigma < f\sigma_0$. Inaccurate predictions can increase our variance estimation error, and consequently the outlier detection tests.

5.3 Requirements Analysis

With many machines in the data center, each reporting many counters every minute or so, the amount of data produced by the monitored machines is too large to simply send it to a centralized location. This problem increases with the number of machines, reported counters, and the reporting frequency.

Rather than send all this data, the use of sketches can help reduce communication volume by an order of magnitude or more, while still preserving accuracy guarantees.

Before applying the sketch function, all counters must be scaled to uniform variance. Safe zone techniques allow us to efficiently monitor the global variance of each counter, enabling nodes to locally scale their counters before sketching.

5.4 Employed Data Sets

We have evaluated our previous work in-situ on historical data from several real live cloud services from Microsoft, currently in production. The main experiments were performed with up to 4500 machines, tested for over 60 days, with each machine reporting around 500 counters at various rates, from every minute up to every several hours. Health data was also provided by the existing monitoring and management system. The dataset is further described in (Gabel et al., 2012). The size of the logs (12TB per day) and their commercial nature prevents their use outside Microsoft.

Since the performance of the general framework has already been established, the evaluation of the communication-efficient online adaptation can employ a smaller subset of the above dataset.

We plan to continue our experiments using data from real, live services.

5.5 Results

Using a subset of the real world dataset in (Gabel et al., 2012), we performed a series of simulations to explore the behavior of the online communication-efficient algorithm. During simulation we run through the counter logs, and simulate the algorithm described in Gabel et al. (2013). We simulate the operation of nodes and the coordinator, and keep track of communications and results. The dataset consists of counter logs for $M = 110$ machines, each reporting $C = 216$ counters that were automatically selected during the pre-processing phase described in the general framework (Gabel et al., 2012). The window length was set to $T = 144$. Unless otherwise noted, our simulation includes the local slack and reference point prediction mechanisms described in Section 5.2.3. The slack factor was set to $w = 0.7$ by tuning.

We evaluate performance with three metrics:

Communication fraction The fraction of floating point values sent by the system, out of the centralized cost of sending C counters from each machine at each round, for a total of $T \cdot M \cdot C$ values. Note that we count the total number of discrete values, rather than number of messages sent. We include all sent values: those sent by the nodes, the coordinator, and the sketched values sent for the test.

Error The mean absolute different between the order of magnitude of machine p-values in the centralized sign test and its communication-efficient adaptation. Formally, let $p(m)$ and $p'(m)$ be the mean p-values assigned to machine m across all time windows in the centralized test and the communication-efficient adaptation, respectively. Our error measure is: $\frac{1}{M}\sum_{m\in\mathcal{M}} \log_{10} p(m) - \log_{10} p'(m)$. We take the log of the p-values because significance thresholds are normally set based on magnitude, for example 0.01 or 0.0001.

Detection error The fraction of classification differences across all machines and time windows. This measure indicates the ability of the adapted sign test to give identical classifications (suspect or healthy) as the centralized test. We set the significance level to $\alpha = 0.01$.

5.5.1 Performance

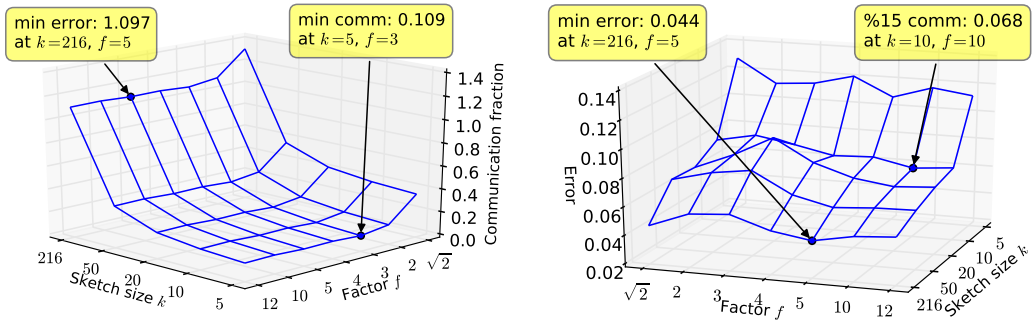
We performed a parameter sweep over several values of sketch size k and safe zone threshold factor f. The results are shown in Figure 5.2.

Figure 5.2(a) shows the fraction of communications performed for each k, f combination. Minimal communication (10%) is achieved when $k = 5, f = 3$ with an error of 0.068, while minimal error (0.044) is achieved at $k = 216, f = 5$ with communication costs of 110%. Sketch size k has the largest effect on communication. Without sketching, when $k = C = 216$, communication fraction is greater than 1.0, since in addition to monitoring overhead, each nodes sends 216 counters at each round. When sketching is applied, communication drops to 50% – 10%, depending on the exact values of k and f. Very tight safe zones ($f \in \{\sqrt{2}, 2\}$) also result in an increase in communication. When the safe zone scaling factor is more permissive ($f \geq 3$) monitoring cost is relatively constant, and communication due to larger sketch sizes dominates communication cost.

Figure 5.2(b) shows the resulting error (mean difference between magnitude of average machine p-value). Smaller sketches result in larger errors, but the effect of safe zone size are not as clear, with an odd peak around $f = 3, k = 20$.

One interesting observation is that more permissive safe zones $f \geq 3$ do not result in lesser communication. We attribute this phenomenon to the effectiveness of the reference point prediction. Figure 5.3 shows performance metrics with the reference point prediction mechanism disabled. Communication costs (Figure 5.3(a)) are now more predictable: both sketch size k and safe zone size f affect communication. Increased safe zone size means less communication, with the minimum achieved as expected for the largest safe zone factor $f = 12$ and the smallest sketch size $k = 5$. Error behavior (Figure 5.3(b))is now more clear: smaller sketches induce greater errors, while larger safe zones barely have little effect.

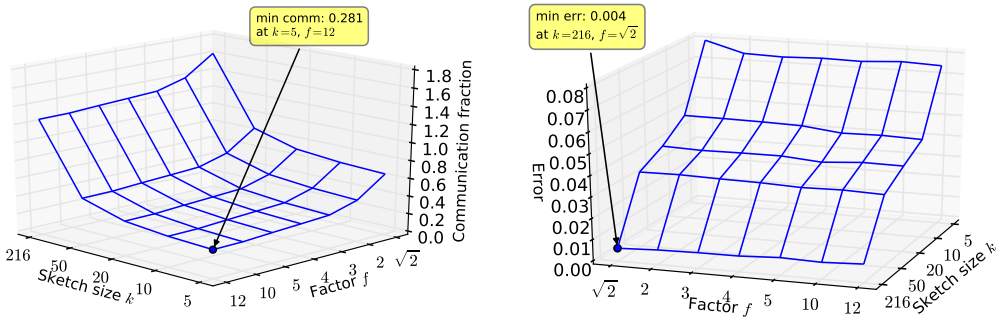
Another anomaly, the increased error around $f = 3, k = 20$, also disappears when prediction is disabled. We further investigate the effects of prediction in Section 5.5.2.



(a) Communication fraction.

(b) Error.

Figure 5.2: Sign test performance at different k, f combinations, compared to centralized test.



(a) Communication fraction.

(b) Error.

Figure 5.3: Sign test performance with reference point prediction disabled.

Detection Performance

We explore the practical effect of different k, f combinations on the latent fault detector. Recall that detection error is defined as the fraction of different machine classifications (healthy, suspect) between the centralized approach and the adapted, communication-efficient online approach. Figure 5.4 shows detection error for each k, f configuration. As with 5.2(b), there is a peak around $k = 20, f = 3$, but in general we see that for $k \geq 10$ there are little to none detection error. We can therefore conclude that the adapted latent fault detector is robust, and matches the centralized approach very well despite an order of magnitude reduction in communication.

Communication-accuracy Tradeoff

To strike a balance between accuracy and communication, Figure 5.5 plots the resulting communication fraction vs. error of different k, f combinations. The best configurations are those near the lower-left corner: minimal communication (X axis) and minimal error (Y axis). Note that $k = 5, f = 3$ results in minimum communication, but in larger error. Indeed, Figure 5.4 shows this configuration results in actual detection errors, where a healthy machine is flagged as suspect or vice versa. Conversely, $k = 216, f = 5$ results in the least error, but entails sending all data to the coordinator, requiring 110% communication. We select $k = 10, f = 10$ as the best configuration, resulting in 15% communication cost with little error, and indeed no detection errors at all.

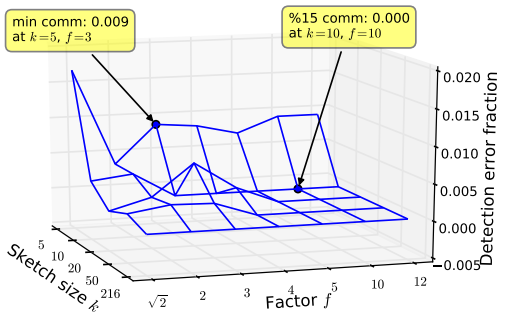


Figure 5.4: Sign test detection errors at different k, f combinations.

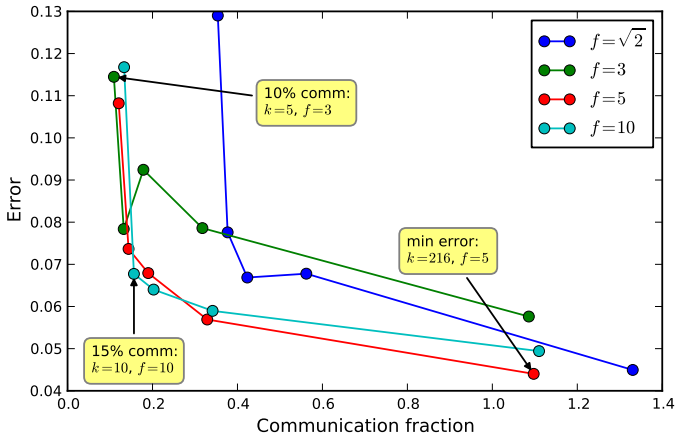


Figure 5.5: Performance at different k, f combinations, each line represents different safe zone scaling factor f.

5.5.2 Slack and Reference Point Prediction

Section 5.2.3 describes local slack and reference point prediction, two techniques we use to reduce communication, though with a possible increase in error. To explore how these techniques affect performance, we repeated the k, f parameter sweeps with different versions of the online, communication-efficient algorithm.

Basic Without local slack and without reference point prediction.

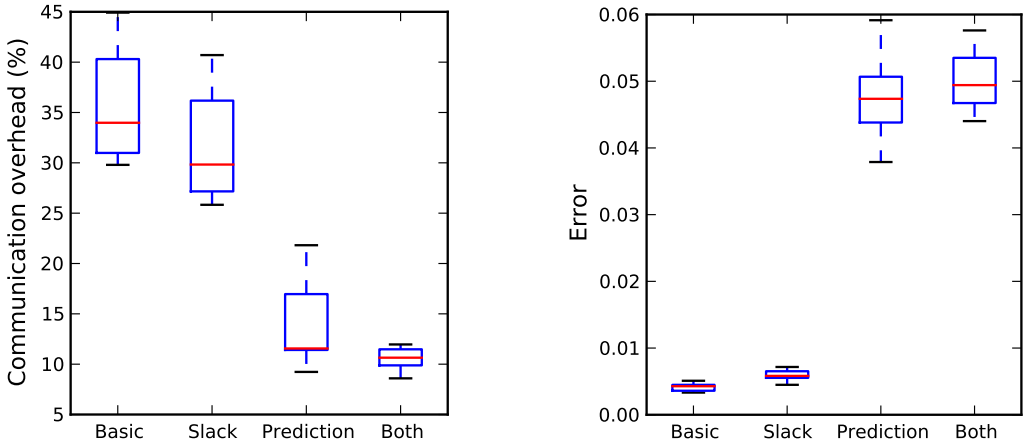
Slack Slack factor set to $w = 0.7$, without reference point prediction.

Prediction With reference point prediction but without local slack.

Both Both slack ($w = 0.7$) and reference point prediction are used.

Figure 5.6 compares the four versions across different f values. Sketching is turned off and the relevant communication is not counted, in order to focus on the impact of the monitoring methods. The box-plots describe the range and distribution of data, with the median marked in a red line. As

shown in Figure 5.6(a), local slack has only modest contribution, because the savings due to avoided local violations are offset by the need to transmit the slack factors. Indeed the best improvement is produced by reference point prediction, which requires no additional communication. Reference point prediction reduces the overhead from 30% to 10%. This reduction, however, is not free. Figure 5.6(b) shows that reference point prediction greatly increases the error. This is both because there are less synchronizations, and because we allow the predicted reference point to move outside the safe zone of the original reference point. Conversely, local slack distribution has very small impact on accuracy.



(a) Communication overhead over centralized version.

(b) Error relative to centralized (no sketch) version.

Figure 5.6: Performance of different monitoring versions, compared to the centralized version, at different f values, and without sketching ($k = C$).

Types of Violations

Recall the 3 types of safe zone violations possible in the monitoring algorithm: true violation, global violation and local violations. Local slack and reference point prediction are designed to reduce synchronization by reducing the amount of local violations. To asses the effectiveness of the two violation prevention mechanisms, we plot the number and composition of different types of violations in Figure 5.7. By itself, local slack reduces the total amount of violations to 70%, while reference point prediction reduces the total amount to 39%. Combined, the number of violations is reduced to 29%.

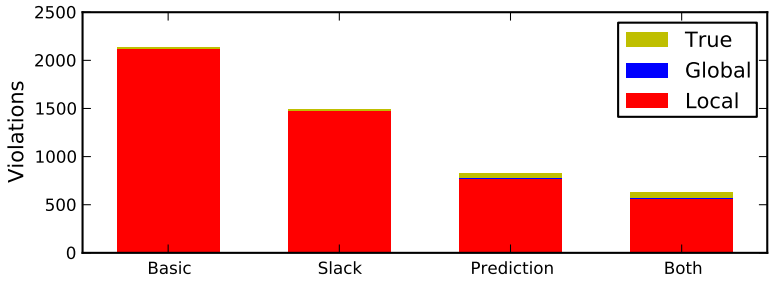


Figure 5.7: True, global and local violations in version of the monitoring algorithm with $f = 10$.

Figure 5.7 suggest that in practice, global violations are almost non-existent. Our experiments show that on average, when not using prediction and slack, only 0.2% of violations are global violations. Furthermore, the average ratio of global violations to true violations is always below 0.001. This suggests that our choice of convex safe zone is indeed close to optimal, since practically every time the global aggregate crosses the safe zone, it also crosses the admissible region.

5.6 Discussion

We began with a general framework and derived tests designed to detect latent faults. This framework, though easy to implement on a data-parallel systems, does not directly address the large amount of communication and processing it requires. We applied the principles and techniques of local inference to reduce the amount of transmitted data: sketching is used to reduce the amount of data that must be sent, and safe zones are used to monitor the data range and scale it to approximately uniform variance. Following the principle of local inference, our nodes make use of already available information to avoid communication by predicting future data behavior. Beyond that, the coordinator balances available slack across the nodes, so that each node is allowed more deviation based on its local data.

We set out to reduce communication by about an order of magnitude, and we conclude that we have achieved this goal. Our experiments on data from a real-world production system show that we are able to achieve 15% reduction in communication without any new false positives or false negatives. If a small amount of incorrect classifications (1%) is allowed, we can reduce communication to 10% of the original centralized communication. We can conclude that the LIFT techniques and principles apply are well-suited to this setting.

The adapted latent fault detector is robust. In practice, it turns out to be is very resilient to scaling errors. Even when the true variance is allowed to drift quite far from the last reference point, latent fault detection performance seldom drops. Similarly, even at small sketch sizes (216 counters reduced to 5 dimensions) the adapted latent fault detector performs close to the centralize version. We plan to further explore the reasons for and limits of this robustness, and how it depends on the data.

Another interesting observation that may be relevant in other LIFT applications is that once prediction is used, larger safe zones do not reduce communication costs. We are not yet sure what causes this. It is possible that prediction is so effective that when it is successful, the error is low, and when it fails, the error is so large that even the large safe zones do not prevent synchronization.

How much further can we reduce communication, or alternatively, reduce error without increasing communication? Future work will concentrate on additional techniques. Local violation resolution can be used to avoid synchronizations by having nodes contact each other directly to exchange slack. Since we know the adapted detector is robust to scaling errors, nodes or the coordinator could simply avoid some global synchronizations if the last one was too recent. Another idea, potentially applicable in other LIFT applications, is to assign different monitoring mechanisms to different counters. In many settings, each monitored counter is completely independent from the others, and can behave differently. Some counters may change often, other may be relatively constant except for major epoch changes, and so on. We can explore different strategies to automatically determine the best monitoring algorithm for each counter.

Chapter 6

Financial Monitoring

6.1 Motivation

In this scenario we monitor the fundamental values of assets in financial markets in order to detect speculative bubbles. The goal is to design a monitoring system that determines the deviation of asset prices from their fundamental value based on asset price correlations and financial news. The vast amount of financial assets and asset prices as well as financial news requires an amount of computation power that can only be provided cost-efficiently by large-scale distributed systems, e.g., a cloud. To process a realistic amount of data on a real-world distributed systems, the communication overhead of distribution has to be drastically reduced in order to comply with network capacities. We employ LIFT techniques, i.e., the safe zone approach, to reduce communication by processing data locally and inferring global values without communication.

According to mainstream economics, on a perfect market, the fundamental value of an asset is equal to its price (Fama, 1970). However, actual markets never fully fulfill the criteria of a perfect market, such that there is a discrepancy between the fundamental value of an asset and its price (Grossman and Stiglitz, 1980). This value can be calculated by summing up all future cash flows the asset generates, discounted to the present day. This method is called discounted cash flow and the estimated value is denoted fundamental value (Kruschwitz and Löffler, 2006).

If the fundamental value of an asset is known, the deviation of the current price from it can be calculated and thereby it can be determined if an asset is well priced, undervalued or overvalued. This information is highly important to investors for the evaluation of investments in terms of return and risk. It is also important to market authorities to detect and prevent adverse market behavior. An example for such adverse behavior are speculative bubble, i.e., trade in high volumes at prices that considerably deviate from the fundamental value (King et al., 1993). Being able to predict the fundamental value of assets for the present day enables market authorities to detect speculative bubbles and take measures to prevent, or decrease their adverse effects.

In retrospective, the fundamental value can be computed with high accuracy, as the cash flows of a long period are known and the discounting factor ensures that the unknown cash flows have only little impact on the overall value. However, predicting the fundamental value, even for the present day, is a difficult problem, because not only future cash flows are unknown but also present cash flows are usually unaccessible (e.g., the earnings of a company are only published quarterly).

In preliminary experiments we found solid indication that correlations of asset prices as novel features yield predictive power to detect speculative bubbles in the stock market. We now want to investigate their predictive power as features in a dynamical model of the asset market, where the goal is not only to detect bubbles, but to predict the deviation of an asset's price from its fundamental value. Furthermore, (financial) news provide an insight to real-world events affecting financial markets. We want to investigate the possibility of improving the prediction by including features derived from financial news, e.g., topics or keywords in related articles.

Once the model has been trained, the task is to monitor the deviation of asset prices from

their fundamental value. One application is to find sets of assets that are undervalued and thus are a good investment. Another application is to find sets of assets that are overvalued, i.e., that form or are part of a speculative bubble. As the number of subsets that have to be monitored is potentially the size of the power set of assets in the market, an efficient method of monitoring is essential for this task.

6.2 Scenario Description and Formalization

Given a set of assets, their prices for a larger timespan and their fundamental values deduced from published fundamental indicators (e.g., balance sheets of companies), we learn a function that predicts the deviation of the current price of an asset from its fundamental value. To learn this function we use ridge regression (also known as regularized regression) on a novel feature space of correlations as well as features derived from financial news. Once the function is learned, we monitor the function with respect to a threshold on the deviation of asset prices from their fundamental value. If the deviation is larger than a given threshold for a set of assets, this set is affected by a speculative bubble.

Because in todays markets, assets are highly interdependent, even for different markets (e.g., the impact of the housing market on the stock market in the 2007 financial crisis), it is not sufficient to monitor markets individually. To reliably model even only a single market, as many assets from as many different markets as possible have to be taken into consideration. The ultimate goal is to monitor all financial assets and their interdependencies simultaneously and in real-time. This task requires large amounts of computing power, which can only be provided by large-scale distributed systems, like clouds. Even for smaller markets, the advantages in terms of costs per computing power of large-scale distributed systems over workstations are considerable, further supporting the development of a distributed monitoring system.

Using such a large-scale distributed system increases the overall computation power on the cost of communication overhead. As the amount of communication essentially grows quadratically with the number of sets of assets in a market that are to be monitored, which is exponential in the size of the market, communication reduction is vital for distributed monitoring of large markets, because communication can otherwise exceed the network capacity in a distributed system, rendering most of its computation power useless. Using the safe zone approach (Sharfman et al., 2007), we are able to monitor all interesting sets of assets in a market on a distributed system (e.g., a cloud) while reducing the communication overhead significantly.

By imposing a hierarchical structure on the sets of assets using the attributes of the assets (e.g., country of origin, asset type, or sector), we can further reduce computational complexity. By exploiting the hierarchical structure to resolve safe zone violations we can further reduce the amount of communication.

Formally, given a market $\mathcal{A} = \{a_1, ..., a_n\}$, we want to learn a function

$$\mathbf{f} : (2^{\mathcal{A}}, \mathbb{N}) \rightarrow \mathbb{R} \ ,$$

where for a set of assets $A \subseteq \mathcal{A}$ and a point in time $t \in \mathbb{N}$, $\mathbf{f}(A, t)$ is the average deviation of asset prices of assets in A from their fundamental value. As speculative bubbles are defined as large overvaluations, we can formalize the task of detecting speculative bubbles as monitoring the function $\mathbf{f}$ according to a threshold θ:

$$\mathbf{f}(A, t) > \theta \Rightarrow \text{The set of assets } A \text{ is affected by a speculative bubble at time } t \ .$$

After having formally defined the scenario, we now want to present an approach to modeling our objective function. Therefore, we approximate the objective function by a linear model on a non-linear feature space of asset prices and pairwise correlations. Using a feature map Φ, we can represent our function as

$$\mathbf{f}(A, t) \approx f(A, t) =< \Phi(A, t), \vec{w} > \ .$$

Now, assume we are given a set of examples

$$E = \{((A,t),y) | t \in \mathbb{N}_+, A \subseteq \mathcal{A}, y \in \mathbb{R}\} \ ,$$

where y is the target value. We can learn our linear model by using regression techniques, such as ridge regression. This linear model is then monitored.

6.3 Requirements Analysis

The monitoring of the deviation of asset prices from their fundamental value on a large financial market requires an amount of computation power that can only be provided cost-efficiently by a large-scale distributed system (e.g., a cloud). The local nodes in the distributed system get sets of assets assigned and receive a stream of asset prices for those assets. Asset prices are updated daily, minute-wise or even in real-time. While in preliminary experiments, only a part of the stock market is used, the goal is to include as many asset markets as possible to our monitoring framework, such as the bond market, resources and foreign exchange rates. Furthermore, each node receives news articles related to the assigned assets. These are locally preprocessed in order to extract additional features which then supplement the stream of asset prices.

Because our method requires an amount of communication quadratic in the number of sets of assets that are monitored, which is exponential in the number of assets on the market, network capacities in the distributed system can be exceeded quickly. LIFT provides the means to reduce communication down to a level where networks are able to handle the amount of communication without delay.

The main technique we use for communication reduction is the safe zone approach, respectively the geometric approach. We construct geometric safe zones in the domain of our function which ensure that, as long as no local safe zone is violated, the global function value does not cross a given threshold.

While the safe zone approach is applicable and led to communication reduction in preliminary experiments, the amount of reduction of this technique alone might not suffice. Therefore, we want to implement a hierarchical structure on sets of assets in order to be able to resolve safe zone violations on a low level of the hierarchy. This way, full synchronizations of the whole system can be avoided, greatly increasing the amount of communication reduction. However, further research on the hierarchical safe zones has to be conducted.

6.4 Employed Data Sets

For our experiments we use stock market data, as it is publicly available and the fundamental value of stocks can be estimated in retrospective using fundamental indicators from the balance sheets of the corresponding companies.

We acquired all publicly available stock prices from Google finance (www.google.com/finance), for which historical prices are available. This dataset comprises daily stock prices of 7949 stocks from 1.1.1970 until now. We also obtained the annually earnings per share of all companies listed in the DAX (German stock index) from their published balance sheets for the years 2007, 2008, 2009 and 2010.

6.5 Results

In (Kamp, 2011) we have shown that speculative bubbles, i.e., a large set of stocks for which their prices deviate significantly from their fundamental value, can be detected efficiently from stock prices using a heuristic based on the time series of prices and their correlations. We developed a distributed monitoring system that scales with huge numbers of stocks, i.e., for n stocks, partitioned into m subsets and r machines in the distributed system, the time complexity is $O(n/r+m/r \log r)$.

By applying the safe zone approach to the monitoring system we were furthermore able to reduce overall communication by 30%.

In order to improve the method from a heuristic to a theoretically sound detection technique, we investigated the predictability of the fundamental value, or fair value of stocks. To obtain training data, we defined the fundamental value of stocks in retrospective as a linear combination of the earnings of the respective company and market key data, i.e., the key interest rate, the annual average inflation and the annual average GDP growth. This value has then be predicted by our method based on price data as well as past, already published but deprecated earnings data. The prediction model here is a regularized least squares regression. We compared this method to a simulated business insider that has access to the current earnings of the company. Experiments on the Standard& Poor's 100 index and the German stock index DAX 30 showed that the method is competitive with the business insider, i.e., for over 90% of stocks the root mean square error of the predictions generated by our model does not deviate more than two times the root mean squared error of the business insider (Kamp and Gärtner, 2012).

However, experiments have also shown that, while the fundamental value of stocks has been predicted accurately for companies with small variance in their quarterly earnings, the prediction has been significantly less accurate for stocks of company with high variance in their earnings (see fig. 6.1 for an exemplary comparison of Apple Inc. and Dell Inc.).

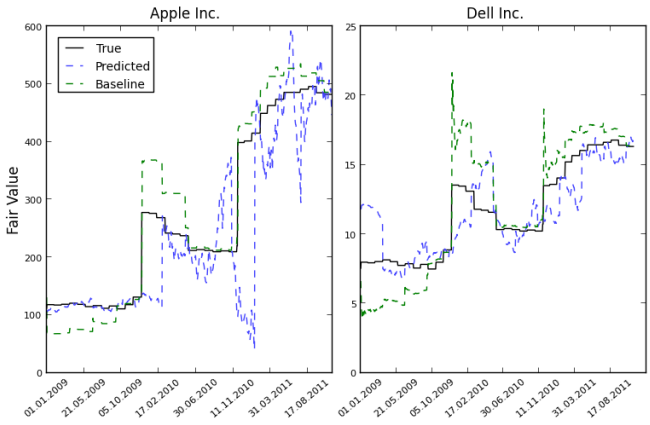


Figure 6.1: Plot of predicted value, baseline and true fundamental value for Apple Inc. (left) and Dell Inc. (right) from 1.1.2009 to 31.12.2011, both taken from S&P 100. The first 500 days are omitted, because this period is used as training set for our method. The predictions are very accurate for Dell Inc. with an RMSE of 1.91 and outperforms the baseline that has an RMSE of 2.32.. For Apple Inc., the baseline has a high RMSE of 40.65, indicating that the fundamental value is hard to predict for Apple. However, our approach performs even worse with an RMSE of 90.02. This high error can be explained by the two very large earnings jumps in the third quarter of 2009 and the fourth quarter of 2010.

From these experiments we have seen that it is necessary to have a good estimation of the current earnings of a company to be able to predict the fundamental value accurately. We found that by modeling the relationships between stock price and earnings of a company as well as the stock prices of related companies enables to predict corporate earnings from public stock price data. By incorporating features derived from the correlation of the stock price time series, short-term changes in the relationship between stock price and earnings, as well as between stock prices of related companies can be modeled. Using this correlation feature space, we were able to not only predict corporate earnings accurately from public price data but to outperform human experts, i.e., analysts in the form of published consensus forecasts (Kamp and Gärtner, submitted).

The method of monitoring correlations using the LIFT approach as developed in (Kamp, 2011) can be applied to the earnings prediction scenario as well. It remains to combine the developed methods to obtain a market monitoring system that reliably detects speculative bubbles. The

LIFT-approach has then to be applied to this monitoring system as a whole.

6.6 Discussion

After developing a preliminary distributed and scalable market monitoring system that detects speculative bubbles, we provided a theoretically founded method for detecting deviations of stock prices from their fundamental value, thereby laying the basis of an improved market monitoring system. We furthermore developed a method for predicting corporate earnings from public stock price data that outperforming experts analysts. This method further improves the fundamental value prediction. While these results are a tangible progress in data driven economics, the methods have yet to be combined into one distributed monitoring system. Also further potential improvements from incorporating additional data, e.g., news articles, twitter tweeds or financial statements, should be investigated.

The LIFT-approach in the form of safe zones has successfully been applied to the initial market monitoring system. Because of the rapidly changing stock prices and their correlations, reaching a quiescence in the distributed system is only possible for very short periods. The achieved communication reduction rate of 30% can further be improved using a more elaborate synchronization protocol. Preliminary experiments with hierarchical synchronization have been promising. We will apply these synchronization schemes to the final monitoring system.

Bibliography

O. Abul, F. Bonchi, and M. Nanni. Never walk alone: Uncertainty for anonymity in moving objects databases. In *Proc. of the 24th International Conference on Data Engineering (ICDE'08)*, pages 376–385. IEEE, 2008.

C. Aggarwal. On unifying privacy and uncertain data models. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 386–395. IEEE, 2008.

N. Alon, Y. Matias, and M. Szegedy. "The Space Complexity of Approximating the Frequency Moments". In *Proceedings of ACM STOC*, pages 20–29, Philadelphia, Pennsylvania, May 1996.

N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137 – 147, 1999.

N. Andrienko and G. Andrienko. Spatial generalization and aggregation of massive movement data. *IEEE Transactions on Visualization and Computer Graphics*, 17:205–219, 2011. ISSN 1077-2626. doi: http://doi.ieeecomputersociety.org/10.1109/TVCG.2010.44.

ma 2012 Plakat Methoden-Steckbrief zur Berichterstattung. Arbeitsgemeinschaft Media-Analyse e.V. (ag.ma) and Media-Micro-Census GmbH (mmc), 2012. URL `http://www.agma-mmc.de/publikationen/methodische-berichte/methoden-steckbriefe.html?eID=dam_frontend_push&docID=1792`.

K. Ashok and M. Ben-Akiva. Dynamic origin-destination matrix estimation and prediction for real-time traffic management systems. In *International Symposium on the Theory of Traffic Flow and Transportation (12th: 1993: Berkeley, Calif.). Transportation and traffic theory*, 1993.

J. Barceló, L. Montero, L. Marquès, and C. Carmona. Travel time forecasting and dynamic origin-destination estimation for freeways based on bluetooth traffic monitoring. *Transportation Research Record: Journal of the Transportation Research Board*, 2175(-1):19–27, 2010.

P. Bodík, M. Goldszmidt, A. Fox, D. B. Woodard, and H. Andersen. Fingerprinting the datacenter: Automated classification of performance crises. In *Proc. EuroSys*, 2010.

S. Burdakis and A. Deligiannakis. Detecting outliers in sensor networks using the geometric approach. In *Proc. of the 28th Int. Conf. on Data Engineering (ICDE'12)*, pages 1108–1119, 2012.

G. Cormode and M. Garofalakis. "Approximate Continuous Querying over Distributed Streams". *ACM Trans. on Database Systems*, 33(2), June 2008.

E. Fama. Efficient capital markets: A review of theory and empirical work. *The journal of Finance*, 25(2):383–417, 1970.

S. Florescu. Efficient retrieval of mobility patterns on mobile devices. Master's thesis, RTWH Aachen Uiversity, 2012.

S. Florescu, C. Körner, M. Mock, and M. May. Efficient mobility pattern stream matching on mobile devices. In *Proc. Ubiquitous Data Mining Workshop (ECAI2012)*, 2012.

M. Gabel, A. Schuster, R.-G. Bachrach, and N. Bjørner. Latent fault detection in large scale services. In *Proc. of the 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2012.

M. Gabel, D. Keren, and A. Schuster. Communication-efficient outlier detection for scale-out systems. In *First International Workshop on Big Dynamic Distributed Data (BD3), held at the 39th International Conference on Very Large Data Bases (VLDB)*, 2013.

M. Garofalakis, D. Keren, and V. Samoladas. "Sketch-based Geometric Monitoring of Distributed Stream Queries". In *Proc. of the 39th Intl. Conference on Very Large Data Bases*, Trento, Italy, August 2013.

N. Giatrakos, A. Deligiannakis, M. Garofalakis, I. Sharfman, and A. Schuster. "Prediction-based Geometric Monitoring of Distributed Data Streams". In *Proceedings of ACM SIGMOD*, Scottsdale, Arizona, May 2012.

P. Gibbons. Distinct-values estimation over data streams. In *In Data Stream Management: Processing High-Speed Data*. Springer, 2009.

D. Goldschlag, M. Reed, and P. Syverson. Onion routing for anonymous and private internet connections. *Communications of the ACM*, 42:39–41, 1999.

N. Gonçalves, R. José, and C. Baquero. Privacy preserving gate counting with collaborative bluetooth scanners. In *On the Move to Meaningful Internet Systems: OTM 2011 Workshops*, pages 534–543. Springer, 2011.

S. Grossman and J. Stiglitz. On the impossibility of informationally efficient markets. *The American Economic Review*, 70(3):393–408, 1980.

R. Gupta, K. Ramamritham, and M. K. Mohania. Ratio threshold queries over distributed data sources. In *ICDE*, pages 581–584, 2010.

J. M. Hellerstein, T. Condie, M. Garofalakis, B. T. Loo, P. Maniatis, T. Roscoe, and N. Taft. "Public Health for the Internet (PHI) – Towards a New Grand Challenge for Information Management". In *Proceedings of CIDR*, Asilomar, California, January 2007.

L. Huang, X. Nguyen, M. Garofalakis, J. M. Hellerstein, M. I. Jordan, A. D. Joseph, and N. Taft. "Communication-Efficient Online Detection of Network-Wide Anomalies". In *Proceedings of IEEE INFOCOM*, Anchorage, Alaska, May 2007.

R. Huebsch, B. N. Chun, J. M. Hellerstein, B. T. Loo, P. Maniatis, T. Roscoe, S. Shenker, I. Stoica, and A. R. Yumerefendi. "The Architecture of PIER: an Internet-Scale Query Processor". In *Proceedings of CIDR*, Asilomar, California, January 2005.

M. Isard. Autopilot: automatic data center management. *SIGOPS Oper. Syst. Rev.*, 2007.

M. Kamp. Automatic detection of anomalous stock market states using local inference, 2011. Diploma Thesis.

M. Kamp, C. Kopp, M. Mock, M. Boley, and M. May. Privacy-preserving mobility monitoring using sketches of stationary sensor readings. In *Proceedings of Europrean Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML-PKDD 2013*. IEEE, 2013.

M. Kamp, M. Boley and T. Gärtner. Predicting the fundamental value of financial assets by ridge regression. In *25th European Conference on Operational Research (EURO 2012)*, 2012.

M. Kamp, M. Boley and T. Gärtner. Beating human analysts in nowcasting corporate earnings by using publicly available stock price and correlation features. In *13th International Conference on Data Mining (ICDM 2013)*. IEEE, submitted.

D. Keren, I. Sharfman, A. Schuster, and A. Livne. Shape sensitive geometric monitoring. *TKDE*, 2011.

D. Keren, G. Sagy, A. Abboud, D. Ben-David, A. Schuster, I. Sharfman, and A. Deligiannakis. Geometric monitoring of heterogeneous streams. *TKDE*, submitted.

R. R. King, V. L. Smith, A. W. Williams, and M. Van Boening. The robustness of bubbles and crashes in experimental stock markets. *Nonlinear dynamics and evolutionary economics*, pages 183–200, 1993.

C. Kopp, M. Mock, and M. May. Privacy-preserving distributed monitoring of visit quantities. In *ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 438–441, 2012.

C. Kopp, M. Mock, O. Papapetrou, and M. May. Largescale online mobility monitoring with exponential histograms. In *First International Workshop on Big Dynamic Distributed Data (BD^3), to appear*, 2013.

C. Körner. *Modeling Visit Potential of Geographic Locations Based on Mobility Data*. PhD thesis, University of Bonn, Germany, 2012.

C. Körner, D. Hecker, M. May, and S. Wrobel. Visit potential: A common vocabulary for the analysis of entity-location interactions in mobility applications. In M. Painho, M. Y. Santos, and H. Pundt, editors, *Geospatial Thinking*, Lecture Notes in Geoinformation and Cartography, pages 79–95. Springer, 2010.

L. Kruschwitz and A. Löffler. *Discounted cash flow: a theory of the valuation of firms*, volume 330. John Wiley & Sons, 2006.

T. Liebig, Z. Xu, M. May, and S. Wrobel. Pedestrian quantity estimation with trajectory patterns. In *Proceedings of the ECML/PKDD*, 2012.

O. S. Maps. www.osm.org.

A. Monreale. *Privacy by Design in Data Mining*. PhD thesis, University of Pisa, 2011.

A. Monreale, G. Andrienko, N. Andrienko, F. Giannotti, D. Pedreschi, S. Rinzivillo, and S. Wrobel. Movement data anonymity through generalization. *Transactions on Data Privacy*, 3(2):91–121, 2010.

A. Monreale, W. Wang, F. Pratesi, S. Rinzivillo, D. Pedreschi, G. Andrienko, and N. Andrienko. Privacy-preserving distributed movement data aggregation. In *The 16th AGILE International Conference on Geographic Information Science (*AGILE*) 2013*, 2013a.

A. Monreale, W. Wang, F. Pratesi, S. Rinzivillo, D. Pedreschi, G. Andrienko, and N. Andrienko. Differential privacy in distributed mobility analytics. *Technical Report TR-13-03 Computer Science Dept. Univ. of Pisa*, 2013b.

A. Monreale, W. Wang, F. Pratesi, S. Rinzivillo, D. Pedreschi, G. Andrienko, and N. Andrienko. Differential privacy in distributed mobility analytics. *Proceedings of the VLDB Endowment (PVLDB)*, submitteda.

A. Monreale, M. Nanni, V. Grossi, R. Trasarti, and D. Pedreschi. Privacy in distributed monitoring. *International Conference on Data Engineering (ICDE 2014)*, submittedb.

M. Nanni, R. Trasarti, A. Monreale, and D. Pedreschi. Distributed monitoring of cluster assignment quality. In *IEEE International Conference on Data Engineering (ICDE 2014)*, submitted.

M. E. Nergiz, M. Atzori, Y. Saygin, and B. Güç. Towards trajectory anonymization: a generalization-based approach. *Transactions on Data Privacy*, 2(1):47–75, 2009.

Octotelematics. http://www.octotelematics.com/.

O. Papapetrou and M. Garofalakis. "Continuous Fragmented Skylines over Distributed Streams". In *30th IEEE Intl. Conference on Data Engineering (ICDE'2014)*, submitted.

O. Papapetrou, M. Garofalakis, and A. Deligiannakis. "Sketch-based Querying of Distributed Sliding-Window Data Streams". In *Proc. of the 38th Intl. Conference on Very Large Data Bases*, pages 992–1003, Istanbul, Turkey, August 2012.

F. Pratesi, A. Monreale, W. Wang, S. Rinzivillo, D. Pedreschi, G. Andrienko, and N. Andrienko. Privacy-preserving distributed movement data aggregation. In *(*SEBD*) 2013*, 2013.

F. Pratesi. Privacy by design in distributed mobility data. *Master Thesis in Computer Science, University of Pisa*, February, 2013.

R. K. Sahoo, A. J. Oliner, I. Rish, M. Gupta, J. E. Moreira, S. Ma, R. Vilalta, and A. Sivasubramaniam. Critical event prediction for proactive management in large-scale computer clusters. In *Proc. SIGKDD*, pages 426–435, 2003.

I. Sharfman, A. Schuster, and D. Keren. A geometric approach to monitoring threshold functions over distributed data streams. *ACM Transactions on Database Systems (TODS)*, 32(4):23, 2007.

Appendix A

Publications

M. Nanni, R. Trasarti, A. Monreale and D. Pedreschi. Distributed monitoring of cluster assignment quality. Submitted to 30th IEEE Intl. Conference on Data Engineering (ICDE'2014).

A. Monreale, M. Nanni, V. Grossi and D. Pedreschi. Privacy in Distributed Monitoring. Submitted to 30th IEEE Intl. Conference on Data Engineering (ICDE'2014).

M. Kamp, C. Kopp, M. Mock, M. Boley and M. May. Privacy-Preserving Mobility Monitoring using Sketches of Stationary Sensor Readings. ECML/PKDD'13, 2013 (to appear).

C. Kopp, M. Mock, O. Papaetrou and M. May Largescale Online Mobility Monitoring with Exponential Histograms First International Workshop on Big Dynamic Distributed Data (BD^3), 2013 (to appear).

F. Pratesi, A. Monreale, W. H. Wang, S. Rinzivillo, D. Pedreschi, G. Andrienko and N. Andrienko. Privacy-Aware Distributed Mobility Data Analytics. SEBD 2013.

N. Giatrakos, A. Deligiannakis, M. Garofalakis, I. Sharfman and A. Schuster. Distributed Geometric Query Monitoring using Prediction Models. Submitted to ACM Trans. on Database Systems. (Extended version of (Giatrakos et al., 2012))

M. Garofalakis, D. Keren and V. Samoladas. Sketch-based Geometric Monitoring of Distributed Stream Queries. In Proc. of the 39th Intl. Conference on Very Large Databases (VLDB'2013), Trento, Italy, August 2013.

O. Papapetrou and M. Garofalakis. Continuous Fragmented Skylines over Distributed Streams. Submitted to 30th IEEE Intl. Conference on Data Engineering (ICDE'2014).

D. Keren, G. Sagy, A. Abboud, D. Ben-David, A. Schuster, I. Sharfman and A. Deligiannakis. Geometric Monitoring of Heterogeneous Streams. Submitted to TKDE.

M. Gabel, D. Keren and A. Schuster. Communication-efficient Outlier Detection for Scale-out Systems. First International Workshop on Big Dynamic Distributed Data (BD^3), 2013 (to appear).

M. Kamp, M. Boley and T. Gärtner. Predicting the Fundamental Value of Financial Assets by Ridge Regression. In 25th European Conference on Operational Research (EURO 2012), 2012.

M. Kamp, M. Boley and T. Gärtner. Beating Human Analysts in Nowcasting Corporate Earnings by using Publicly Available Stock Price and Correlation Features. Submitted to ICDM 2013.

Distributed monitoring of cluster assignment quality

Mirco Nanni[1], Roberto Trasarti[1], Anna Monreale[1,2], Valerio Grossi[2], Dino Pedreschi[2]
ISTI-CNR, Pisa, Italy[1], University of Pisa, Pisa, Italy [2]
{nanni,trasarti}@isti.cnr.it[1], {annam,pedre,vgrossi}@di.unipi.it[2]

***Abstract*—Distributed streams of time-evolving data pose significant issues in several data analysis tasks that were initially studied having in mind static and centralized contexts. In particular, communicating the full streams of data updates to a single, centralized location might be unsustainable for various reasons, such as information traffic congestion. In this paper we tackle one task of this kind, namely the monitoring of cluster assignments, having the objective of continuously testing the quality of a given assignment of cluster labels (for instance obtained through a clustering or community discovery algorithm, or through a set of user-defined rules). We investigate the context where each data collection node represents a single object to be labelled, and propose a solution based on the testing of local conditions performed by each node. The solutions proposed, then, are tested over synthetic data and on real data. For the latter, in particular, a full case study is provided, consisting of a customer relationship management (CRM) application in the context of car insurances, based on the monitoring of mobility behaviors of customers. For this case study, we validate the framework on a large database of real mobility data, coming from GPS devices of private cars.**

I. Introduction and motivations

The current spread of data collection technologies in business, scientific research and private lives is leading towards the so called era of big data. Big data have several peculiarities that make them difficult to deal with, such as their size, their distributed and streaming nature, that make them difficult to store in – or even to send to – a centralized location. These features can pose significant issues in several data analysis tasks that were initially studied having in mind static and centralized contexts. The objective of this paper is to study a specific problem in such kind of context, namely the monitoring of cluster assignment quality.

Cluster assignments emerge naturally in several application domains. For instance Customer Segmentation, a basic task in business intelligence, is focused on assigning customers to homogeneous categories – for instance through recency-frequency-monetary segmentation rules (RFM) – to be separately dealt with by the CRM expert; Community Discovery is a very popular (social) network analysis problem that basically yields a cluster assignment; finally, several traditional clustering algorithms have been studied in the literature to provide cluster assignments for a wide range of specific objectives.

The stream and distributed nature of big data requires that the cluster assignment process is modified in order to produce reasonable assignments without sifting all the data updates. Also, such process should push as much as possible the computation towards the nodes of the distributed network of data sources. A basic way to do that consists in collecting all the data only for an initial time slot, compute the cluster assignment we need, and then test in the successive time instants whether the initial assignment is still coherent with the data updates received from each data source. For instance, if all updates involve just a negligible change of our variables, it is clearly useless to send data around and recompute the assignments – a kind of test that each node (or data source) can perform individually, i.e. locally. On the opposite, a single large variable update could change the assignments, and therefore some kind of global synchronization is needed. Following these ideas, we address the problem of continuous monitoring of the quality of the cluster assignments by adapting and extending the *safe-zones* approach [19], a method that supports the distributed monitoring of global functions (especially non-linear ones) while limiting as far as possible the need for communication between the distributed agents and a central server.

The notion of cluster assignment quality we adopt in this paper concerns the compactness of clusters, for instance requiring that the group of customers labelled with the same cluster do not become too dispersed. We remark that this does not directly involve other possible notions of cluster quality, such as cluster separation, cluster density or the optimization of some objective function. That means, for instance, that two compact clusters that – step by step – get closer to each other till they overlap, might still satisfy our compactness requirements, and therefore be acceptable. This apparently weak requirements look actually reasonable in several contexts, where the focus is on the significance of the cluster, and not on its diversity from other ones. For instance, customer segmentation has the practical objective of identifying and understanding customers, and a stable assignment is usually preferable even if the different segments are not neatly separated.

The challenge here is to properly exploit the looseness of the requirements we are discussing, in order to limit the need for communicating information and recomputing assignments. Indeed, as we will remark when comparing against an alternative approach in literature, focused on guaranteeing the (much more restrictive) quasi-optimality of the clustering, such a looseness creates a large margin of communication reduction.

The main novel contributions of our work can be summarized as follows:

1) we formulate the cluster assignment monitoring as a problem that can fit the *safe zone* general framework;
2) we develop a three-layer logical architecture of the monitoring system, aimed to limit the need of global communications;

3) we introduce an improvement of the *safe zone* approach based on history-based predictive models, particularly well suited for data streams showing significant periodic behaviours (very frequent with human-generated data);
4) we perform a wide testing of our solution both on synthetic and real data;
5) we develop a full case study on customer relationship management (CRM) in the context of car insurances, with a deep exploration of performances and results.

II. Problem definition

In this section we formalize the problem of continuously monitoring the quality of a cluster assignment in a distributed setting. We start by introducing the quality measure adopted in this work, and later use it to define two variants of the corresponding monitoring problem. Here the discussion will assume to work in a centralized context where it is possible to store and analyze all the mobility data streaming in. In Section IV we will move to a more realistic setting, where the communication issues (inevitable on a large scale application like the case study discussed in Section VI) are considered and dealt with.

Quality measure. A simple and very popular method for measuring the overall quality of a clustering is the so-called *Sum of Squared Error* (SSE in short), defined as follows:

$$SSE = \sum_{i=1}^{k} \sum_{p \in C_i} ||p - c_i||_2^2 \quad (1)$$

where C_i represents the i-th cluster, and c_i is its center (average vector). This measure evaluates the dispersion of each cluster around its centroid, and therefore gives emphasis to the compactness of clusters.

The SSE is also at the core of several clustering methods, especially the K-means algorithm. Indeed, it is easy to prove that K-means tries to reduce the SSE at each step of its iterative process, stopping when a local optimum is reached.

Monitoring formulation of the problem. Our approach to deal with dynamic data consists in continuously checking whether the current cluster assignment is still good enough, raising an alarm in the negative case. This requirement can be easily translated in terms of SSE by asking that the dispersion of the objects within the clusters did not grow, or at least not significantly. That means computing the SSE at each time stamp t, which we will denote with SSE_t, and test that it stays below some threshold. We refer to this continuous testing with the term *monitoring*. Finally, such a threshold should take into account the dispersion obtained at the very moment the cluster labels were assigned, which we denote with SSE_0 (i.e. time counting starts from the moment the most recent assignment was performed), suggesting to adopt a relative threshold. That is summarized in the following problem definition:

Definition 1 (Cluster Monitoring Problem): Given a clustering $C = \{C_1, \ldots, C_k\}$ having initial SSE equal to SSE_0, and given a tolerance $\alpha \in \mathcal{R}^+$, we require to ensure that at each time instant t the following holds for the SSE of the (dynamic) dataset D_t:

$$SSE_t \leq (1 + \alpha) SSE_0 \quad (2)$$

When that does not happen, an alarm is raised.

We should note that SSE describes all the clusters together, aggregating the dispersions of the single clusters. That means that in principle having a good SSE does not guarantee that each single cluster is compact, since some slightly over-dispersed cluster might be balanced in the sum by some virtuous one that adds very little to the SSE. From this perspective, it might happen that the end user of our application wants to ask for stronger requirements in the monitoring problem. Therefore, we will consider also the following variant of the problem, where the constraints are imposed over each single cluster:

Definition 2 (Strict Cluster Monitoring): Given a clustering $C = \{C_1, \ldots, C_k\}$ having initial SSE equal to SSE_0, and given a tolerance $\alpha \in \mathcal{R}^+$, we require to ensure that at each time instant t the following holds:

$$\forall_{i=1}^{k}. SSE_t^{(i)} \leq SSE_0^{(i)} + \theta^{(i)} \quad (3)$$

where $SSE_t^{(i)}$ is the contribution of cluster i to the SSE at time t, i.e. $SSE_t = \sum_{i=1}^{k} SSE_t^{(i)}$, and the $\theta^{(i)} \in \mathcal{R}^+$ are fixed thresholds such that $\sum_{i=1}^{k} SSE_0^{(i)} + \theta^{(i)} = (1+\alpha) SSE_0$. When condition (3) is violated, an alarm is raised.

III. Related Work

We briefly summarize some works in the literature on fields that are tightly connected to our proposal.

Distributed Clustering can be classified into two main groups. The first one includes completely decentralized methods requiring a significant amount of communications among nodes [5], [7], [3]. Most of these works propose algorithms for k-means clustering over a P2P network. The second group requires that the nodes build local clustering models and send them asynchronously to a central station, that forms a combined global model [18], [17], [13]. The basic assumption of these algorithms is that each node contains more than one point. Some of these works propose algorithms for hierarchical clustering [17], others extend the density-based clustering to this setting [13] and others present a k-means version suitable for wireless sensor networks with a hierarchical structure [18]. As opposed to the methods mentioned above, in our context, in a given timestamp each node has a single point that is transmitted to the central station for the clustering. Moreover, our setting does not allow communications among nodes.

Clustering on Data Streams is addressed in [10]: given a sequence of points, the approach maintains a good clustering of the sequences observed, using a small amount of memory and time. Some other works in the data stream scenario may be found in [1], [2], [6]. [9] proposes an algorithm for clustering distributed data streams. Given a network of nodes, where each of them receives its share of a distributed data stream, the goal is to obtain a common clustering. Here,the nodes cannot share

single points of their datasets, but only aggregate information. This setting differs from ours because they image to have more than one point in each node.

The single work that is most related to ours is [21] where the authors study the problem of continuous k-means computation at a server that monitors a set of moving objects. In particular, the paper proposes an approach, named TKM, for avoiding the re-evaluation of k-means for each object update, that typically imposes large communication and computation costs due to the fact that the updates can be frequent. The solution proposed provides a guaranteed bound on the quality of the solution obtained with respect to the naive approach that recomputes the clusters at each iteration. The framework assigns each moving object a specific threshold (i.e., range) such that the object sends a location update only when it crosses the range boundary. The key feature of this work is that it aims at guaranteeing the optimality of the clustering in terms of quality of the centroids. Clearly, this requirement is significantly stricter than those involved in our monitoring of cluster assignments. Indeed, our problem does not directly constrain the location of the clusters' centers, implicitly tolerating the collective movement of a whole cluster as far as it does not cause the dispersion of the cluster itself. However, exploiting this additional degree of freedom in the monitoring problem is the major challenge in this work. While our problem can, to some extent, be translated to a continuous k-means computation, this reduction leads to inefficient solutions, as will be proved in our experimental sections.

Monitoring. Besides the methods addressing the monitoring of arbitrary threshold functions over distributed data streams described in [19], [8] and already discussed in previous sections, in the literature some works treat the problem of clustering monitoring by considering different aspects and settings. For instance, [20] studies the monitoring of the L2 norm, useful for monitoring the accuracy of a k-means clustering. Others consider the clustering monitoring in a P2P network by proposing a hierarchical structure of nodes [12].

IV. Distributed monitoring

Our context implies that our data sources are both streaming and distributed. For instance, in the application described in Section VI several vehicles are involved, each playing the role of a node that continuously generates updates for its indicators of driving behaviour, and all these data need somehow to be collected by a single unit to elaborate them. This creates a simple network with several *nodes* that communicate exclusively with a single special node, that we call *controller*, where nodes just communicate everything and the controller performs all the computation. However, such a purely centralized solution is applicable only on the small scale, since the communication stream generated by millions of nodes would be hard to sustain. In this section we propose a solution based on the distributed monitoring paradigm, that aims to save a significant amount of communications by deferring a small part of the computation to the nodes of the network. More specifically, our objective is to continuously verify that the cluster assignment satisfies the quality constraint (2) or (3), and recompute the clustering only when the constraint is violated. Therefore, a simple way to distribute the computation consists in providing to the nodes some *local* conditions to test such that, if each local test is successful, they guarantee also the satisfaction of the *global* quality constraint. The basic idea is that the nodes themselves can recognize the data changes that do not significantly impact on the quality measure to monitor, thus avoiding to update the controller with useless information.

In the following subsections we describe the setting of distributed monitoring of functions that we are going to use, showing how our problem can be translated is such terms. Then, we describe the full framework that implements the monitoring and integrates several forms of predictive models that improve its communication reduction capabilities.

A. Distributed monitoring of functions

The work in [19] addresses the general problem of monitoring the value of a function computed over data that are distributed in a network. More specifically, the framework considers a two-tiered setting, with n geographically dispersed sites (also called *nodes*) and a central coordinator (also called *controller*) that is capable of communicating with every site, while pairwise site communication is only allowed via the coordinating source. Each site receives a stream of data updates and maintains a d-dimensional local measurements vector $v_i(t)$. The task of the coordinator is to ensure that at each time instant t the following holds:

$$f(v(t)) \leq T \tag{4}$$

where f is a given function, $f : \mathcal{R}^d \rightarrow \mathcal{R}$, $T \in \mathcal{R}$ is a threshold and $v(t)$ is the weighted average of the $v_i(t)$ of all sites, i.e. $v(t) = (\sum_i w_i v_i(t))/\sum w_i$ for some weights $w_i \geq 0$. While the latter condition is apparently a strong limitation to the applicability of the framework, it has been shown that several interesting problems can be reformulated in this way. A basic means to do this, which will also be used later in this paper, is a *vector augmentation trick*, consisting in adding to vectors $v_i(t)$ (sent by the sites to the coordinator) one or more extra components. The basic example is the task of monitoring the variance of all $v_i(t)$, assuming $d = 1$, i.e. ensure that $var_i(v_i(t)) \leq T$. While not directly fitting the form in (4), we can exploit the well known property $var(X) = avg(X^2) - [avg(X)]^2$ to rewrite our problem as $f(v^*(t)) = v^*(t)_1 - [v^*(t)_2]^2$, assuming that each site now communicates a 2-dimensional vector $v_i^*(t) = \big(v_i(t), v_i(t)^2\big)$.

The algorithmic solution proposed in [19] to perform the monitoring of (4) follows a so called *geometric approach*. All the points in $\mathcal{R}^d$ where (4) is satisfied form the *admissible region* G, and our objective is simply to ensure that $v(t) \in G$. The method stems from the following property: the convex hull of a set $\{x_i\}_i \subset \mathcal{R}^d$ of points is entirely contained in $\bigcup_i B(x_i, e)$, where e is any point in $\mathcal{R}^d$ and $B(x_i, e)$ is the ball having the segment $\overline{x_i e}$ as diameter. In turn, it is straightforward to see that our $v(t)$ is contained in

the convex hull of the set $\{v_i(t)\}_i$. Therefore, if every ball $B(v_i(t), e)$ is contained in the admissible region (namely, it is *monochromatic*), then also $v(t)$ will be, and therefore (4) will be satisfied. Once the controller has communicated to all sites the point e, each site will be able to test whether its ball $B(v_i(t), e)$ is monochromatic. As long as no site detects a failure, we are guaranteed to satisfy (4), without any need of communicating information to the controller. When a site fails, it notifies the controller, who, in the basic approach, will ask to every site to send their new vector values, and test condition (4). Notice that the test performed on each site might cause false alarms (its ball exits the admissible region, yet the overall $v(t)$ is still inside) but not false negatives, i.e. when condition (4) is violated the system will always discover that. While in principle the point e can be chosen freely, it is convenient in practise to compute it as $e = v(t')$, where t' is the time of the most recent *synchronization*, i.e. the phase where every site communicates its new values to the controller. Beside at the start-up of the system, synchronizations usually occur when a site rises an alarm. With this choice, it results useful to slightly change the method by asking each site to check the ball $B(u_i(t), e)$ instead of $B(v_i(t), e)$, where $u_i(t) = e + (v_i(t) - v_i(t'))$ is called the *drift vector*. It is trivial to prove that the average $avg_i(u_i(t))$ is equal to $v(t)$, thus not changing the monitoring task. Yet, now, immediately after any synchronization we have that $\forall i.u_i(t') = e$, therefore the vectors of all sites start from the same point e, and the balls $B(u_i(t), e)$ have actually the size of a single point, clearly reducing the chance that a false alarm will rise in the near future.

B. Distributed monitoring of cluster quality

The cluster monitor problem (Definitions 1 and 2) can be fitted to the geometric approach described above, by properly rewriting it and exploiting the vector augmentation trick seen for the variance monitoring (Section IV-A). Indeed, the formulation of SSE is very similar to a variance, though on d dimensions. To find the precise relation between the two, we observe that each cluster C_i, having centroid c_i, contributes to the SSE by the following value:

$$\begin{aligned} SSE^{(i)} &= \textstyle\sum_{p \in C_i} ||p - c_i||_2^2 \\ &= \textstyle\sum_{j=1}^{d} \sum_{p \in C_i} (p^j - avg_{q \in C_i}(q^j)) \\ &= |C_i| \textstyle\sum_{j=1}^{d} var_{p \in C_i}(p^j) \\ &= |C_i| \textstyle\sum_{j=1}^{d} \left[avg_{p \in C_i}((p^j)^2) - \left(avg_{p \in C_i}(p^j)\right)^2 \right] \end{aligned} \tag{5}$$

where p^j represents the j-th component of the d-dimensional vector p. This means that by augmenting the vector $v_i(t)$ of each node with the additional d features $v_{i,1}(t)^2, \ldots, v_{i,d}(t)^2$, we can compute the variance for each component, as requested in (5). Actually, we can do slightly better, by further aggregating the terms in the last line:

$$SSE^{(i)} = |C_i| \cdot \left[avg_{p \in C_i} \left(||p||_2^2 \right) - ||avg_{p \in C_i}(p)||_2^2 \right] \tag{6}$$

which means that only one additional component is needed, corresponding to $||p||_2^2$ of the node (p represents our $v_i(t)$).

C. Monitoring framework

The relation (6) states that the geometric approach discussed in Section IV-A can be applied to monitor a single cluster, provided that we have defined a threshold value for it. This provides a direct solution to the strict version of our monitoring problem (Definition 2), since it already introduces an individual threshold for each cluster, and therefore implicitly partitions the problem into K separate sub-problems. The basic problem (Definition 1), instead, does not impose such a partitioning, and allows some interplay between the different clusters, e.g., clusters with a high $SSE^{(i)}$ might be *balanced* by others having a low value.

Problem partitioning. In our solution we choose to divide the monitoring into K separate sub-problems in both the basic and the strict problem definitions. That means, in practice, we will introduce thresholds for each single $SSE^{(i)}$ also in the basic case, thus actually making it more restrictive. Additional mechanisms to exploit the interplay between clusters allowed by the basic problem will be introduced later in this section. Then, after each synchronization, where all nodes send their vectors to the controller (including the initial start-up of the system), the controller will send to the node of each cluster both its centroid and the corresponding SSE threshold. With this information, each node will be able to locally test sufficient conditions that ensure the cluster not to exceed the SSE threshold assigned to it, based on the geometric solution described in Section IV-A.

In order to perform the above mentioned threshold partitioning, we essentially need to define the values (or rather, a method to compute them) of the constants $\theta^{(i)}$ mentioned in Definition 2, i.e. the SSE increments allowed to each cluster. Two natural choices emerge: using the same value for all clusters; or making $\theta^{(i)}$ proportional to $SSE^{(i)}$. The first solution will be called *uniform SSE distribution*, and implies that we set $\forall i.\theta^{(i)} = \alpha SSE_0/K$. The second one will be called *proportional SSE distribution*, and requires to set $\forall i.\theta^{(i)} = \alpha SSE_0^{(i)}$. In order to allow a flexible choice between the two alternatives, we will adopt the following schema, parametric in $\beta \in [0, 1]$:

$$\forall i.\theta^{(i)} = \beta \left(\alpha SSE_0^{(i)} \right) + (1 - \beta) \left(\alpha \frac{SSE_0}{K} \right) \tag{7}$$

Thus, $\beta = 0$ will correspond to the uniform distribution, $\beta = 1$ to the proportional one, and any value in the middle to some trade-off between them.

Multi-level monitoring. The previous discussions suggest that the cluster monitoring can be organized at three levels, as depicted in Fig.1. At the bottom layer, each node checks its local constraints, based on the information it received during the last synchronization. Whenever a local violation occurs at some node, the controller reacts by verifying whether the alarm was true (the $SSE^{(i)}$ of the corresponding cluster is too large) or not, which constitutes the second layer. The basic action taken by the controller is to simply query all nodes of the cluster to compute the real $SSE^{(i)}$, yet smarter

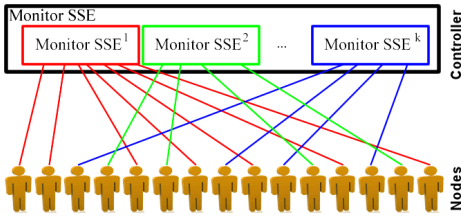


Fig. 1. Logic organization of the framework.

solutions are possible, including the node balancing strategy that will be discussed later in this section. When the alarm is true, the strict problem definition raises an alarm, which typically (depending on the specific application) calls for a recomputation of the clustering, and thus the controller also needs to start a full synchronization. In the basic problem definition a smarter approach is possible (also discussed later in this section) which checks whether other, more *virtuous* clusters have a $SSE^{(i)}$ that can balance our large one.

Algorithm 1 summarizes the whole monitoring process for a more complex case, i.e. the basic problem, where some balancing is possible between single nodes or between clusters.

Algorithm 1: Overall monitoring workflow

```
   Input: number of clusters K; tolerance α; dynamic customers data set D
   // Synchronization and setup
1  Initialize t := 0;
2  All nodes in D send vectors v_i(0);
3  Compute clustering;
4  foreach cluster C_i do
5      Compute threshold SSE^(i)_thr;
6      Send c_i and SSE^(i)_thr to all nodes in C_i;
   // Monitoring
7  repeat
8      t := t + 1;
9      foreach node S_i ∈ D do
10         S_i tests its local constraints;
11         if Violation then
12             Send alarm to Controller;
13     foreach cluster C_i do
14         if Controller receives at least one alarm from C_i then
15             Try node balancing on C_i;
16             if balancing failed then
17                 Try cluster balancing;
18                 if balancing failed then
19                     Exit loop and request synchronization;
20     if Requested synchronization then
           // Recompute clusters
21         Compute clustering;
22         foreach cluster C_i do
23             Compute threshold SSE^(i)_thr;
24             Send c_i and SSE^(i)_thr to all nodes in C_i;
25 until Forever ;
```

First, in an initialization phase (steps 1–6) the controller receives all local vectors $v_i(0)$ from the nodes and computes a first clustering (e.g., running a K-means algorithm, a community discovery method or applying some user-defined rules), sending back to the nodes all the information they need to start the distributed monitoring. The monitoring phase (steps 7–25) contains three main actions: first, each node tests its local constraints (steps 9–12) using the methods described in Sections IV-A and IV-B. If no violation occurs, the monitoring cycle simply goes on without any communication and any action from the controller's side. If, instead, there is some local violation, the controller communicates with the nodes of the corresponding cluster to check if the violation can be balanced (steps 13–15). If that is not possible, it means the cluster SSE is too large, and the controller tries to see if other clusters can help to balance it (steps 16–17). If everything fails, it means the whole clustering is actually violating the monitoring condition (2), and therefore an initialization of the system is required (steps 18–19). The latter event triggers the execution of the last phase (steps 20–24), essentially requiring the same operations of the initialization phase. We remark that the strict version of the monitoring problem does not allow any interplay between clusters, therefore, in that case the cluster balancing (step 17) should simply be omitted. Also, as we will see later, the cluster balancing is executed in such a way that when it fails all nodes will already have sent their up-to-date vectors to the controller, and therefore it is not required any further communication from the nodes side. When the cluster balancing is omitted, as for the basic monitoring problem, such communications are needed, and step 2 should be replicated before the clustering takes place (i.e., between steps 20 and 21).

D. Node-level improvements

The proposed solution can be improved in several ways, by exploiting recent developments of the general theory summarized in Section IV-A (which will be discussed in the next paragraphs) or by extending it with application-specific improvements (topic mainly covered in the next section).

Safe Zones for convex inadmissible regions. The geometric method discussed in Section IV-A provides a means to decide locally to the node which vector values guarantee that the overall function satisfies the global constraint to monitor. The set of such values is also called *safe zone* and, since it is only based on the global function and on the reference point e, all nodes have the same safe zone.

In [14] it is shown that the safe zones built by the geometric method can also be computed as the intersection of an infinite set of hyperplanes. Yet, it is also shown that part of them are unnecessary, which makes the safe zones smaller (and thus less effective) than what strictly needed. A particular case is that where the inadmissible region (the set of values that violate the global constraint) is convex. In this situation we can easily find an optimal safe zone in two steps: first, find the point p of the inadmissible region which is closest to the reference point e; second, draw the hyperplane that passes through p and is orthogonal to the segment $\overline{ep}$, and then, of the two half-spaces determined by the hyperplane take as safe zone the one that contains e.

This is very relevant for our problem, since the $SSE^{(i)}$ of a cluster has a quadratic form with positive coefficients, and therefore our inadmissible region is the part of space that stands above an "upward" parabola, which is a convex set. For

this reason, in our application we will adopt the smarter safe zones described above.

Balancing. As already mentioned earlier, when a node violates its local constraints (i.e., it exits its safe zone) there might be other nodes that can compensate it. A simple method to do that consists in applying a straightforward property: if we move the local vectors $v_i(t)$ and $v_j(t)$ of two nodes, respectively by δ and $-\delta$, the overall average $v(t)$ will not be affected. That means that if $v_i(t)$ trespassed the border (an hyperplane, in our case) of the safe zone by an amount δ, generating an alarm, and we realize that $v_j(t) - \delta$ still remains in the safe zone, we can remove the violation by translating both vectors as described above, without any further effect on the monitoring process. Actually, since the border of the safe zone is a hyperplane, we only need to consider δ that are orthogonal to it, therefore we can code the translation by means of a single scalar, representing the distance between $v_i(t)$ and the hyperplane.

The balancing of nodes, then, is realized along the following strategy: when a node of cluster C_i raises a violation, it also communicates its amount δ (which will be negative); then, the controller asks to 2 other nodes of the cluster, chosen randomly, their distance δ_1 and δ_2 from the hyperplane. If $\delta + \delta_1 + \delta_2 \geq 0$, the balancing is successful, otherwise the controller tries with 4 more nodes, etc., at each iteration doubling the number of nodes involved, till either $\delta + \sum_i \delta_i \geq 0$, or all nodes have been contacted without success. In the latter case, the balancing fails.

When the balancing is successful, we have two alternatives, that we call respectively *balancing with memory* and *memoryless balancing*. In the first one, the node that rose the violation and the other ones needed to balance it, permanently translate their vectors; in the second one, the translation is only virtual, and all nodes keep their vector. The latter is expected to work better in situations where the violations are due to temporary noisy updates of the vectors, which should disappear at the next timestamp.

The balancing process described above, when successful, assigns a final distance from the hyperplane that is zero for the node who rose the violation and all those used to balance it, while all other nodes (those queried by the controller but not needed to reach the amount $|\delta|$) are left unchanged. A possible alternative consists in building a *buffer* between all the n nodes and the hyperplane, by redistributing the total amount of δ_i of the nodes not yet involved. That can be realized by setting the distance from the hyperplane of each node involved to $(\delta + \sum_i^n \delta_1) / (n+1)$.

Finally, the same kind of reasoning followed so far can be applied at the level of clusters, in the case of the basic problem definition: if a cluster exceeds its assigned threshold $\theta^{(i)}$, the controller can try to balance it with other clusters. In order to do that, all the nodes of each cluster C_j involved will send their up-to-date vectors, in order to compute the real $SSE^{(j)}$, and see if it can be used to balance $SSE^{(i)}$. Similarly as above, we can also introduce a buffer, in order to provide some extra-tolerance to the clusters for the next iterations of the monitoring. Since the number of clusters is usually relatively low, the exponential schema applied at the level of nodes can be replace by a linear one, i.e. the controller involves new clusters in the balancing one-by-one, instead of doubling their number at each step.

In Section VI we will experiment all these variants over a specific application domain, in order to verify which variant seems to fit better the application.

E. Introducing predictive models

The recent work in [8] extends the geometric approach by introducing predictive models. The basic idea is that the reference point e used to define the safe zones needs not to be static. As far as all nodes always share the same value, it can change in time. The basic idea consists in defining models that provide some predictions for each $v_i(t)$, which in turn are combined together by the controller to provide a model that predicts $v(t)$. Such model is distributed to all nodes, which will then use its predicted values as (time-varying) reference point. The final effect is that, if the predictive models work well, the drift vectors of the nodes (see Section IV-A) will all be very close to the real value $v(t)$, thus minimizing the chance of false alarms.

We distinguish two types of predictive models to use in our framework: one based on the detection of trends in the recent data (introduced in [8]); and one based on the detection periodicities of the data, obtained by analyzing historical information (a novel contribution of this work).

Trend-based models. From this family of models, introduced in [8], we will consider two alternatives: the *linear growth* model, which assumes that the vector of a node will evolve in time following the law $v_i(t) = \frac{t}{t_s} v_i(t_s)$, where t_s is the time of the last synchronization; and the *velocity/acceleration* model, which assumes the law followed has the form $v_i(t) = v_i(t_s) + (t - t_s) vel_i + (t - t_s)^2 acc_i$, where vel_i and acc_i are estimates of the velocity and acceleration (as vectors) of $v_i(t)$ estimated at time t_s by the node itself, by looking to a piece of its recent history. Both models favour simplicity and ease of computation, only requiring to send very little extra information on the network.

History-based models. This models extend the approach of [8] by learning regularities in the behaviour of a vector $v_i(t)$ from a segment of history of its node, in a way similar to [16]. We consider two variants. The first one is called *history-based constant* model, and simply computes the average value of $v_i(t)$ on the history segment, which then represents the "default" vector for the node. The second one is called *history-based variable*, and performs a similar computation, yet providing different values for different time slots within a fixed period. For instance, in the case study of Section VI a period of 24h is used. This way, we will obtain a "default" vector which is dependent on the hour of the day (or other time unit adopted). Since they are based on longer-term history, as compared to the trend-based methods, these solutions require

to send the local models of the nodes to the controller (and the aggregate global model backward) only at the beginning of the monitoring, thus not affecting communications in any significant way.

Adapting the *Choosing Among Alternative* approach shown in [8], we combine together all models listed above in the following way: at each synchronization only one model type is chosen, based on how well each model performed since the previous synchronization. Therefore, each node always applies all models, in order to test whether it would cause an alarm or not, even if only one of them is actually used to decide whether a violation is occurring. During the synchronization, for each model the coordinator counts the number of nodes where the model was successful (i.e., did not signal a false alarm), and selects the best one.

V. Evaluation on synthetic data

In order to provide a first study of the performances of the methods proposed in this paper, we developed a synthetic data generator that allows to create dynamic data following some predefined behaviours.

A. Synthetic data generator

The generator starts by partitioning N nodes into K different clusters. Each cluster is associated to a virtual center, and the data vectors of all nodes belonging to that cluster are initially generated by adding a Gaussian noise to the cluster center. The simulation runs for several iterations, at each step making each node move in a random direction at a (Gaussian) random distance from the previous position. The dynamicity of the data, i.e. how fast the data vectors move, is controlled by the dynamicity parameter D, which represents the standard deviation σ_D of the random length of each single displacement. D, in particular, is expressed as the ratio between σ_D and the minimum distance between the initial cluster centers. Finally, each time a direction and length of displacement is chosen, it is kept for I iterations. I represents an inertia factor, and trend-based predictive models (see Section IV-D) are expected to exploit that and provide larger performance improvements for larger values of I.

While the process can be easily generalized to produce vectors of any dimensionality, at the present the generator has been implemented and tested over two dimensions. That allowed to perform a close inspection of the generated data, through static plots and animations. The data generator and the datasets used in this experiments (including plots and animated videos of the corresponding data) will be made publicly available[1].

B. Comparison with TKM

In order to compare our approach against the TKM method introduced in [21], we implemented a Java version of its whole monitoring process.

[1]http://kdd.isti.cnr.it/node/493

Parameter	Values (default in bold)
dinamicity (D)	0.1, 0.2, **0.3**, 0.4, 0.5
inertia (I)	5, **10**, 25
n. of clusters (K)	5, **10**, 15, 20, 25, 50, 75
n. of nodes (N)	100, 200, **500**, 1000, 2000, 5000

TABLE I
Range and default value of the simulation parameters.

During the initial step the server computes the standard k-means obtaining the reference clustering M^{REF}, then computes the centroids and sends to each node p_i a threshold θ_i. Such thresholds are derived through an optimization problem that starts from a parameter Δ provided by the user. In particular, the threshold assignment ensures that the clustering obtained by TKM has at each instant a cost (equivalent to the SSE measure we adopt in this paper) which is at most larger than the optimum by $n\Delta^2$, where n is the number of nodes. The schema requires a node p_i to send the update to the server only if its current location deviates from the previous one by at least θ_i.

When the server receives updates from a subset of the total nodes, it obtains the new k-means using HC^*, that is an optimized version of *hill climbing* (HC) algorithm able to recompute the clustering using the new updates and the last recorded locations of the other objects which are still within their assigned thresholds. Then, the server computes the new threshold values for the nodes involved in the last iteration.

In order to use TKM for our cluster assignment monitoring problem, we basically need to fix a threshold Δ that can guarantee the property stated in Equation (2). That can be done by equating the tolerance αSSE_0 required in (2) with the upper bound $n\Delta^2$ guaranteed by TKM, thus obtaining $\Delta = \sqrt{SSE_0 \times \frac{\alpha}{n}}$.

In the TKM process, Δ is actually used to derive other thresholds, denoted as Δ_1 and Δ_2, for which we used the default relations $\Delta_1 = 0.7 \times \Delta$ and $\Delta_2 = 0.3 \times \Delta$ suggested in in [21].

C. Evaluation

The generator mentioned above has been used to create several datasets, by varying the values of the four key parameters discussed: dynamicity (D), inertia (I), number of clusters (K) and number of nodes (N). The range of values considered in the simulations, together with the default value for each parameter, are summarized in Table I. Each simulation lasts exactly 1000 iterations.

The monitoring method that involves history-based models requires to use a portion of the data to create a predictive model. In our experiments, the first 200 iterations of the simulated data were used for this training phase, while the remaining 800 iterations were used to measure the performances of the system. The framework we proposed was run for the basic cluster monitoring problem using a default setting where $\alpha = 1.0$ (see Definition 1) and $\beta = 0.5$ (see Equation 7). The clusters were computed running a K-means algorithm with

the correct number of clusters (i.e., the parameter K of K-means was set to the number of clusters created by the data generator). This step is performed at the initialization phase and repeated whenever a global resetting of the system is required, i.e. when the global SSE value exceeds the actual threshold.

Figure 2 concisely reports the results of the experiments, expressed in terms of fraction of communications. Values close to 1 mean that most of the data generated by each node at each time step are sent to the central controller, thus yielding a large communication traffic, while a value close to 0 means that only a very small portion of such communications took place. For each parameter setting, three different datasets have been generated, and the average percentage of communications is reported in the plots.

The results largely confirm the expected behaviour of our framework. When the dinamicity of the data increases (top left plot), it becomes easier for the nodes to violate their local conditions (i.e., exiting their safe zones) and therefore the need for communicating the nodes' vector becomes more frequent. The effect of increasing the inertia parameter I (top right plot) goes in the expected direction, i.e. communications decrease, yet the impact on performances look rather limited: increasing the minimum inertia by a factor of 5 (from 5 to 25) just decreases communications from 42% to 40%. Increasing the numbers of clusters has in principle two positive effects on performances: each cluster becomes smaller, reducing the probability that one of its nodes rises an alarm, and the local alarms can be more easily balanced (if it was a false alarm) by peeking at other clusters. The plots (bottom left of Figure 2) reflect this phenomena, and show that larger values of K consistently yield smaller percentages of communications. Finally, the framework appears to scale relatively well with larger numbers of nodes (bottom right plot), since the percentage of communications show some small fluctuations but generally follows a stationary trend.

The method proposed in [21] was tested on the same set of datasets described above. In our experiments we observed that in all cases the communications exceed 95%, quite often getting even close to 100%. That empirically proves that reducing our monitoring problem to the method in [21] introduces too tight constraints, and makes it inefficient in practice.

VI. A CASE STUDY ON MOBILITY BEHAVIOURS

A key task in modern customer relationship management (CRM) is to understand the needs of each customer and to devise policies to harmonize them at the best with the company objectives. In most cases that translates into identify a portfolio of *customer profiles*, each representing the needs and requirements of a reasonable number of customers. Then, each profile can be treated separately in order to devise the market strategies and business models that best fit its customers' peculiarities. In the business intelligence field, this process is best known as *customer segmentation*, and is traditionally implemented by applying pre-defined customer classification rules (for instance based on RFM indices: Recency of last contact with customer, Frequency of transactions, Monetary volume involved in the relation with the customer) or, in more recent times, by using clustering algorithms. This kind of process can be applied to any kind of business focused on services towards several customers, including the classical domain of retail selling as well as e-commerce and many others. In particular, in this section we describe an application of customer segmentation in the very actual and rather uncommon ground of car insurance business, where the main features that characterize a customer are related to how he/she drives, and therefore on his/her mobility. Using the solution introduced in the previous sections, we instantiate the general problem to the new context by adopting a data mining-oriented approach, working from a realistic *big data* perspective, where the application can benefit from a continuous flow of fresh information.

The first point leads us to model the characteristics of the customers through features extracted from his/her mobility. In particular, we analyze the trajectories that describe such mobility based on standard GPS traces, as those recently collected by specialized companies for the car insurance sector.

A. Background on trajectory data analysis

The history of a user is represented by the set of points in space and time recorded by his/her mobility device defined as $H = \langle p_1 \dots p_n \rangle$ where $p_i = (x, y, t)$ and x, y are spatial coordinates and t is an absolute time-point. Starting from this sequence we are interested in extracting the user's trajectories, where a trajectory is a subsequence of points representing the movement between two places where the user stops for an activity. In the literature there are complex techniques for stop computation, but in our experiences a simple cut when using a minimum period of time γ in considering a spatial buffer with radius r is a good trade-off between the computational efficiency and quality of result obtained. A common setup of the parameters in case of private cars is $\gamma = 2hrs.$ and $r = 50m$. In the following, we will also use the concepts of "the most frequent location $L1$" and "the second most frequent location $L2$". These two areas represent for the user the most attractive places and in the literature, usually are interpreted respectively as $home$ and $workplace$.

B. Application definition

For the description of the *user driving behavior* in a time window we identified four categories of measures: (i) *basic*, (ii) *space-time distribution*, (iii) *context-aware*, and (iv) *behavioral*. The first one contains measures describing the *basic* features of the trajectories in the time window such as:

Length: distance travelled by the user.
Duration: time spent travelling by the user.
Count: number of different user's trips.
MaxAcceleration: maximum user's acceleration.
MaxDeceleration: maximum user's deceleration.

This information is directly computable from the raw GPS traces without any complex process. However, they are useful to understand the behavior of the car usage. The second

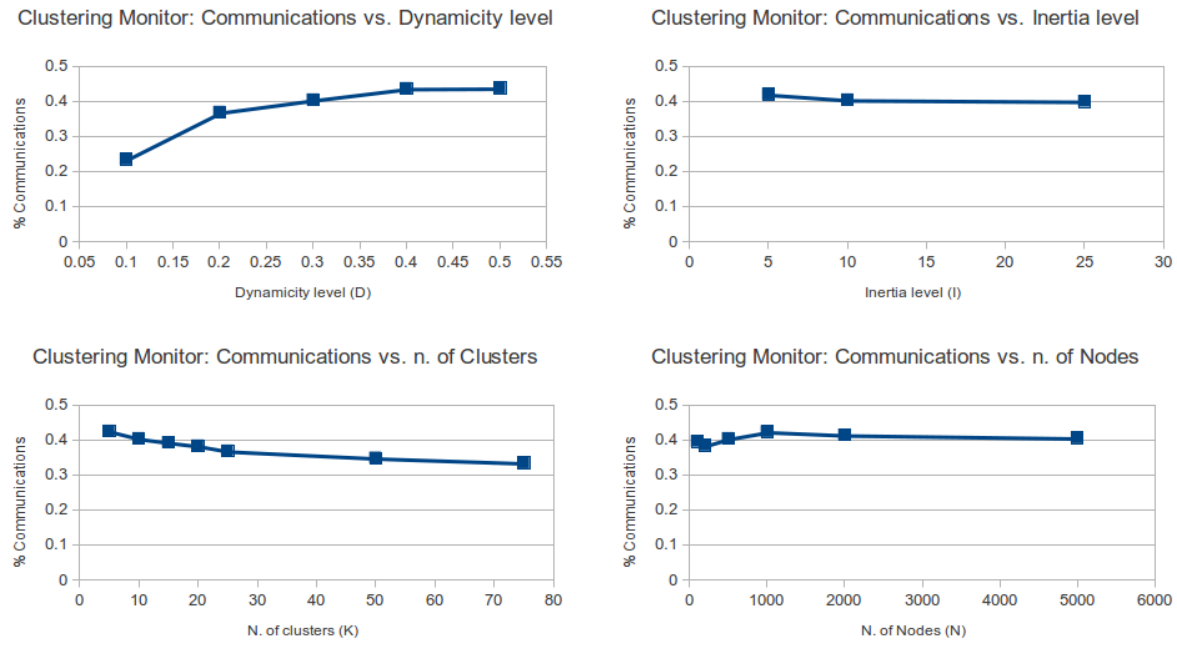


Fig. 2. Communications performed during clustering monitor over synthetic datasets obtained by varying the following simulation parameters: dinamicity (D), inertia (I), number of clusters (K) and number of nodes (N).

category comprehends more complex measures that capture how the territory is used, both spatially and temporally:

Avg_Dist_L1: average distance of the user from his most frequent location L1.

Radius_g: radius of gyration of the user (i.e. the standard deviation from the center of mass of his movements).

Radius_g_L1: radius of gyration w.r.t. to the user's L1.

TimeL1L2: time spent by the user in L1 or L2.

EntropyLocation: entropy of the location frequencies where the user stops. Users that visit only a few locations will yield a low entropy, while those that are more dispersed across several different locations will yield a high entropy.

EntropyTime: entropy of user's travel time frequencies. Same as for EntropyLocation, but this time we consider the hour of the day of the visits rather than the locations visited. This set of measures describes the spatial and temporal *user distribution movements*. The third category is composed of the *context-aware* features, where information about the user's movement is related to the spatial and temporal context in which he moves:

EntropyArc: entropy of road segment frequencies traversed by the user. Similar to EntropyLocation, but focused on the routes followed to move between locations.

Phighway: distance travelled on highways by the user.

Pcity: distance travelled inside urban areas by the user.

Length_arc_crowded: distance travelled on top 20% most crowded road segments. This provides a measure of how much the user tends to follow or avoid high traffic roads.

Pnight: distance travelled during night time (i.e. between 10 p.m. and 5 a.m.) by the user.

The last category focuses on capturing some specific behaviors of the user:

PAccelerationDeceleration: percentage of rapid accelerations/decelerations of the user during his/her movements.

Pover: how much the user drives over the speed limits, expressed as percentage of kilometers.

Profile: how much the user follows his profiles, i.e. trips that he performs frequently.

1) Clustering-based customer segmentation: . The indicators built at the previous step provide a summary of all factors deemed relevant to characterize the driving of an individual. On this base, the customer segmentation can be defined essentially in two ways: by means of *user-defined rules* to assign each customer to one out of a set of predefined segments; or by applying a *data-driven approach* that looks for the existence of meaningful groups of customers, each group containing individuals with similar feature values. The first solution mimics quite closely the traditional approach to CRM, where a set of golden rules, learned through years of experience or through prestigious studies, is assumed to hold perpetually, virtually unaffected by external factors. The second solution follows a data-mining perspective, and essentially trades the clear understanding of static, well-established golden rules for the capability of capturing the potentially dynamic group structure of customers directly suggested by their mobility indicators.

Quite obviously, the user-defined rules look advantageous in all contexts where such a set of such rules are known and the domain is characterized by a large inertia. In all other cases (no applicable rules are known or the domain is inherently

dynamic, thus quickly turning such rules obsolete), the data-driven approach may come to help. Since the characteristics of the domain considered in this paper better fit the latter situation, in our work we will devise a data-driven solution to the customer segmentation problem.

Following well-known precedents in the business intelligence literature [4], we choose to build segments through a clustering algorithm. In selecting the most appropriate clustering schema, we should consider the following requirements of our application: first, each segment should contain customers that are significantly similar to each other, and therefore clusters should be basically compact; second, some customers might not fit well any segment, and therefore the clustering procedure should account outliers, to some extent; third, as the pool of customers might be very large, for instance in the order of several millions, the algorithms need to have a low computational complexity.

The first and last requirements can be met very simply by the standard K-means algorithm, which seeks globular clusters and has an almost linear complexity. In order to account outliers and yet keep the overall procedure efficient, we choose to first apply K-means on the input data, and afterwards apply a postprocessing to remove objects too far from the cluster center. In particular, the distance threshold adopted in the experiments we will show later in this section was computed separately for each cluster as $2 \cdot median_{x \in C_i}\{d(x, c_i)\}$, where C_i represents the cluster and c_i its centroid.

The K-means algorithm starts from a set of random cluster centers (centroids), and then iteratively assigns each data object to the closest centroid and then recomputes the centroid of each cluster as the average of its data vectors; the process is repeated until convergence is reached, i.e. the cluster centroids are stable. As we can see, the only parameter of the method is the number of clusters K. If no (correct) guesses for K are provided by the domain knowledge, a reasonable value can be found by applying standard methods (see, e.g. [15]) that run the clustering with several values for K, and select the largest one such that a further increase in K generates no significant improvement in the clusters compactness (the latter being measured through a quantity called SSE, defined in the next sections for other purposes).

2) Customer segmentation on dynamic data: The mobility of individuals is a phenomenon that, in principle, can highly change with time. For instance, some routines might be affected by sudden and temporary variations of the environment (special events or road works forcing people to adapt their daily routes), or they might be influenced by seasonal factors (actually, the whole lifestyle might change from winter to summer). Finally, a change in the mobility might be simply an effect of the natural evolution of individual lives, possibly involving changing working conditions, family status and taste on how to enjoy the spare time – cases in which the variation of the individual mobility is more likely to last relatively long.

For these reasons, in our context it is advisable to have mechanisms that ensure a good fit between the actual segmentation (the one that the company is using to shape its business) and the real mobility behavior of the customers.

Assuming to have access to a stream of continuously updated indicators for all our customers, the first question to tackle is the following: how and when the changes detected on single users should translate into changes in the customer segmentation? Indeed, some individual changes might be small enough to have negligible effects on the corresponding segments, or at least not to impact on the overall structure of segments. In these cases, it is advisable to simply keep the known segmentation, avoiding the practical troubles at the management level that would result from an excessively frequent redefinition of the segments.

In order to implement the general principles mentioned above, we follow the monitoring approach introduced in this paper, where the last computed segmentation is kept as long as the SSE-based quality requirements are satisfied. Then, whenever such requirements are violated, the existing segmentation is discarded, and a new one is computed from the most recent data.

C. Dataset and data preprocessing

The application introduced above has been implemented on top of our monitoring framework ad tested against a dataset of real GPS traces. The dataset is provided by an Italian company called *OctoTelematics* collecting data for insurance purposes. This dataset is composed by GPS observations of 11,470 private cars active in Tuscany in a period of 35 days between June and July 2011. Due to some pre-processing (i.e. aggregation and filtering) performed by the device on board, the sampling rate is reduced to a observation every 3 minutes and it is not regulated by any policy of synchronization.

We divided the dataset temporally in order to create a training and a test set, respectively using the first week and the remaining 4 weeks. The two datasets have been processed to extract the measures presented in Sec.VI-B using a time window of 3 days with a time granularity of 15 minutes[2]. Unfortunately, due to the specific data format adopted by the data provider and to the low sampling rate, some of the measures we discussed cannot be extracted. In particular, all the acceleration-based measures have been excluded from our experiments for this reason.

Once all the measures were collected, we studied their values and since some variables follow a skewed distribution, they were transformed to a log scale. Finally, all variables have been normalized through z-score, thus making all variables have zero average and variance equal to 1. Since the idea is to build the *customer profiles* using a clustering method, we must consider also the correlation between the measures, hence we used the training set to produce a schema of correlations revealing that several strong correlations held. Therefore we selected a subset of measures to avoid strong biases in successive analyses and, as side effect, reduce the dimensionality of the dataset. The attributes remained after this selection are

[2]The processed dataset containing the driving indicators is available at *kdd.isti.cnr.it/node/493*

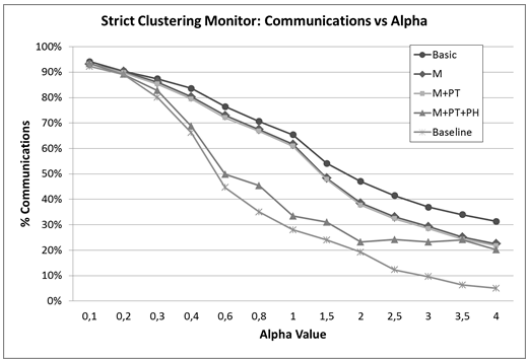


Fig. 3. Communications during the *strict* clustering monitor by varying the α parameter.

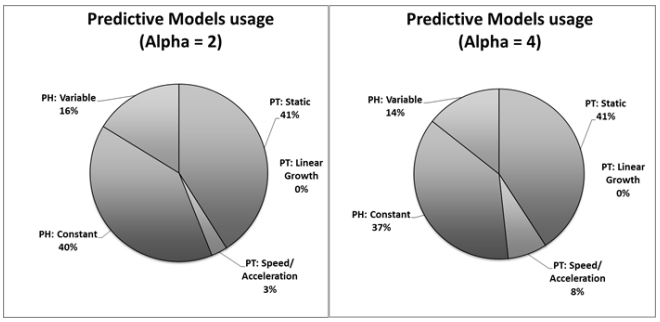


Fig. 4. The percentage predictive model usage.

the following: *Duration*, *Radius_g_L1*, *TimeL1L2*, *EntropyArc*, *Pcity*, *Phighway*, *Pnight*, *Pover* and *Profile*.

D. Performance evaluation

As we did in Section V, we evaluate the performances of the proposed method by measuring the communications exchanged between the nodes and the controller. However, in this case study we provide a much deeper level of detail of the experimental results: first, we separate the messages originated by the nodes and directed toward the controller from those following the opposite direction; second, we consider both the strict and basic monitoring problem definitions; third, we measure the impact of the predictive models we introduced as improvement of the framework; finally, we explore the actual clusters obtained in order to verify whether the overall application we are modelling produces reasonable results.

The nodes-to-controller communications are always of the same size (a vector with d dimensions plus other parameters for predictive models) and the channel is a *point-to-point* link between the node and the controller; the controller-to-node communications, instead, are composed by messages of variable sizes that can use *broadcasting* capabilities of the network to reach all the nodes at once. A worst case analysis of the communications from the controller revealed that they are dominated by the requests for the balancing process, and they have an upper bound of 4%, which in an empirically evaluation drops to values between 1.23% and 2.34%.

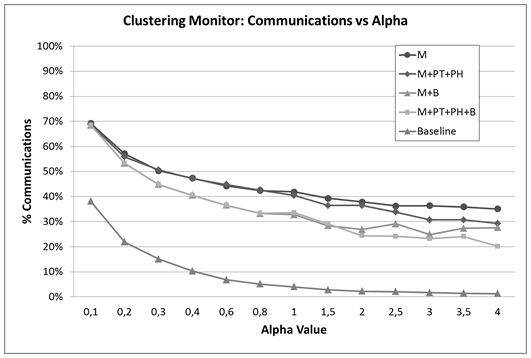


Fig. 5. Communications during the clustering monitor by varying the α parameter.

For sake of readability we first present the results of the strict clustering monitor. In Fig.3 the method is evaluated varying the α parameter. The algorithm is applied with different variants presented in previous sections: the method with only node balancing without memory (**Basic**); with memory during the balancing (**M**); using the trend predictive models (**PT**); using the history predictive models (**PH**); and considering a buffer in the balancing (**B**). Moreover in the figure we report also the baseline representing a lower bound of the communications needed to monitor, computed as the communications generated by synchronizations. It is interesting to see the effect of the introduction of the memory during the node balancing (in the strict problem the clusters are not balanced) which gives a first boost to the performances reducing the communications. The usage of the trend-based predictive models contributes only marginally to reduce communications, while the history-based ones have a rather large impact, reducing the communications close to the baseline. The impact of the history-based predictive models it is also confirmed by the Fig.4 where the usage of each predictive model is analyzed in terms of the amount of time in which it is used during the monitoring. It is important to notice how, excluding the static one, the constant history-based model is the most used outperforming the others. The second most used is the variable version, which is probably more affected by noisy values than the constant version. The least used one, is the linear growth model, which is never used and, as expected, proves to be not suitable to describe the trend of our data. Considering the basic version of the clustering monitor problem, it is clear from the baseline in Fig.5 that the communications needed are less. Our method obtains good performances and, as in the strict version, the predictive models improve the performances. Experiments showed that the buffer feature presented in Sec.IV-D, applied at the node level, does not improve the performances significantly, hence in the experiments we present it only in the case of the balancing applied at the cluster level, where we can see that it reduces significantly the communications, both with and without the predictive model.

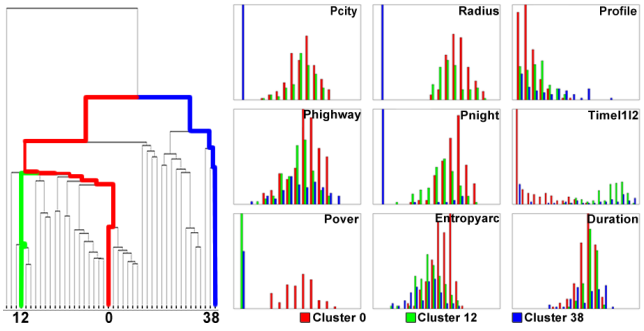


Fig. 6. Hierarchical clustering on a set of centroids (left) and feature distributions (right).

E. Cluster evaluation

Given the result of a clustering operation, our end user needs to interpret the solution by defining and labeling the obtained clusters. Interpreting clusters usually involves examining their centroids, which represent the average behavior of each group of objects. This analysis is important because sheds light on whether the segments identified by the clusters are conceptually distinguishable. As a consequence we have decided to examine the values of centroids and tried to identify explanatory features to profile the clusters. The first step of the interpretation process was to understand the relation among the cluster centroids in order to understand their similarity (or dissimilarity). Therefore, we have applied on the set of centroids a hierarchical clustering algorithm with "single-linkage" method for the distance computation. The result simply highlights groups of centroids (clusters), which are similar and clusters that are particularly different. In this way, it is easy to understand which clusters represent similar customer profiles. In Fig.6 (left) we depict the result of the hierarchical clustering on a set of centroids. We observe that by analyzing the dendogram an analyst could easily understand that the customer profile represented by Cluster 12 (green) and that of Cluster 0 (red) should be more similar to each other than the customer profile of Cluster 38 (blue). This is confirmed by the behaviors of some of the feature distributions in each one of these three clusters. Indeed, in Fig.6(right), we observe that the feature distributions are more similar in Clusters 0 and 12; especially, for almost all the *context-aware* features.

Analyzing the feature distributions in Fig.6(right) we can characterize each one of the three clusters by labeling them with a particular customer profile. We classified Cluster 0 as the group of drivers that we call *explorers*. The people in this cluster travel a lot without following any particular systematic behavior and the entropy of their movements is high. Moreover, this category of drivers tends not to respect the speed limits. Cluster 12, as explained above, is very similar to Cluster 0 but the people here have a more systematic behavior (Profile), spend more time in their frequent locations – typically representing home and working locations –, which probably determine most of their mobility. We identify these people as *long-range commuters*. Finally, Cluster 38 represents drivers called *Sunday drivers*; indeed, they travel rarely, typically use the highways and do not travel within the city. Also, they tend to respect the speed limits.

Another important point in evaluating the application is how much time passes between consecutive reclusterings – i.e., how long does a set of profiles typically remain valid before needing a recomputation. As expected, we verified that it strongly depends on the α tolerance parameter. As an example, for $\alpha = 4$ we perform the re-clustering on average once every 20 hours, and an average of 11 hours for $\alpha = 0.8$. A deeper analysis reveals that this average is strongly affected by a few particularly unstable periods of time where many clusterings are executed consecutively. However, once good profiles are detected the relative clustering persists much longer. These time durations are also affected by the time window adopted, that in our experiments was limited to 3 days due to the relatively short period covered by our raw dataset. Enlarging the window, for instance to 2 weeks or 1 month, we expect to reduce the significant random fluctuations, and therefore largely extend the lifespan of the profiles.

VII. Conclusions

In this paper we tackled the problem of monitoring cluster assignments in a dynamic context, where several nodes continuously receive updates on their variables (or vectors), and communicating them systematically to a central controller could create traffic congestion issues. The proposed framework adapts the existing *safe zones* technologies to the monitoring of a cluster compactness measure, introducing a three-layer logical organization of the monitoring process and improving the general approach with specific extensions, such as a novel kind of history-based predictive models.

Beside testing the approach on synthetic and real data, we also used it as basis for defining and instance of the customer segmentation problem in the context of car insurances, characterized by a strong human mobility component and the large scale, distributed and streaming nature of its data sources. We believe that the distributed monitoring of cluster assignments proposed here can be applied to an extensive array of clustering problems, which we plan to explore and study in detail.

References

[1] C. Aggarwal, J. Han, J. Wang, P. Yu. A framework for clustering evolving data streams. VLDB 2003, pp. 81-92.

[2] B. Babcock, M. Datar, R. Motwani, L. O'Callaghan. Maintaining variance and k-medians over data stream windows. ACM PODS 2003, pp. 234-243.

[3] S. Bandyopadhyay, C. Giannella, U. Maulik, H. Kargupta, K. Liu, S. Datta. Clustering distributed data streams in peer-to-peer environments. Inf.Sci.176(14):1952-1985,2006.

[4] M. J. Berry, and G. S. Linoff. Data mining techniques: for marketing, sales, and customer relationship management. Wiley Computer Publishing, 2004.

[5] S. Datta, C. Giannella, H. Kargupta. Approximate Distributed K-Means Clustering over a Peer-to-Peer Network. IEEE TKDE 21(10): 1372-1388, 2009.

[6] P. Domingos, L. Spencer, G. Hulten. Mining time-changing data streams. ACM KDD 2001, pp. 97-106.
[7] P. A. Forero, A. Cano, G. B. Giannakis. Distributed Clustering Using Wireless Sensor Networks. J. Sel. Topics Signal Processing 5(4): 707-724, 2011.
[8] N. Giatrakos, A. Deligiannakis, M.N. Garofalakis, I. Sharfman, A. Schuster. Prediction-based geometric monitoring over distributed data streams. SIGMOD 2012, pp. 265-276.
[9] A. Guerrieri, A. Montresor: DS-Means: Distributed Data Stream Clustering. Euro-Par 2012, pp. 260-271.
[10] S. Guha, N. Mishra, R. Motwani, L. O'Callaghan. Clustering data streams. FOCS 2000, pp. 359-366.
[11] X.Hong; Q. Gangyi. Data Mining in Market Segmentation and Tariff Policy Design: A Telecommunication Case. Information Processing (APCIP) 2009, pp. 328-331.
[12] M. Hua, M. Ki Lau, J. Pei, K. Wu. Continuous K-Means Monitoring with Low Reporting Cost in Sensor Networks. IEEE TKDE 21(12): 1679-1691, 2009.
[13] E. Januzaj, H.-P. Kriegel, M. Pfeifle. DBDC: density based distributed clustering. EDBT 2004, pp. 88-105.
[14] D. Keren, I. Sharfman, A. Schuster, A. Livne. Shape Sensitive Geometric Monitoring. In IEEE TKDE 24(8): 1520-1535, 2012.
[15] P.-N. Tan, M. Steinbach, V. Kumar. Introduction to Data Mining. Addison-Wesley 2005.
[16] M.Nanni, R. Trasarti, G. Rossetti and D. Pedreschi. Efficient distributed computation of human mobility aggregates through User Mobility Profiles. In UrbComp'12: ACM SIGKDD International Workshop.
[17] N. Samatova, G. Ostrouchov, A. Geist, A. Melechko. RACHET: An efficient cover-based merging of clustering hierarchies from distributed datasets. Distrib. Parallel Databases 11 (2): 157-180, 2002.
[18] P. Sasikumar, S. Khara. K-Means Clustering in Wireless Sensor Networks. Conference on Computational Intelligence and Communication Networks, 2012.
[19] I. Sharfman, A. Schuster, and D. Keren. A Geometric Approach to Monitoring Threshold Functions over Distributed Data Streams. ACM Trans. Database Systems, 32(4), 2007.
[20] R. Wolff, K. Bhaduri, and H. Kargupta. Local L2 Thresholding Based Data Mining in Peer-to-Peer Systems. SDM 2006.
[21] Zhenjie Zhang, Yin Yang, Anthony K. H. Tung, Dimitris Papadias. Continuous k-Means Monitoring over Moving Objects. IEEE Trans. Knowl. Data Eng. 20(9): 1205-1216, 2008.

Privacy in Distributed Monitoring

Anna Monreale [#1], Mirco Nanni [*2], Valerio Grossi [#1], Roberto Trasarti [*2], Dino Pedreschi [#1]

[#1] *Department of Computer Science, University of Pisa*
Largo Pontecorvo, 3, Pisa - Italy
[1] {annam,vgrossi,pedre}@di.unipi.it

[*2] *ISTI-CNR, Pisa*
Via Moruzzi, 1, Pisa - Italy
[2] name.surname@isti.cnr.it

***Abstract*—In many emerging applications, such as real-time traffic monitoring, financial analysis, sensor network monitoring an important task is the continuos monitoring of stream data. In these contexts where large amount of data arrive continually the data processing requires to access often valuable personal information. As a consequence, the entire monitoring process could put at risk the privacy of people represented in the stream data. In this paper, we study the privacy issues in distributed systems during the monitoring of thresholds functions, where several nodes contribute with their data to the monitoring of a specific event. We provide a privacy-preserving framework suitable to find an acceptable trade-off among privacy protection, data quality and system performance. Using real-life data from GPS devices of private cars, we demonstrate the effectiveness of our approach in a case study consisting of the monitoring of customers mobility behaviors; in other words, we show how techniques for efficient communication can be used while preserving the individual privacy of the actors who are participating to the collective analysis.**

I. Introduction

In the last years, many new emerging applications require sophisticated, real-time processing and monitoring of high-volume data streams. These stream-based applications include real-time financial analysis, network and infrastructure monitoring, fraud detection, mobility traffic monitoring and command and control in military environments. All these applications require an important task: the *continuos monitoring of stream data*. In the last years, many study in the literature have been addressed the problem of monitoring queries in distributed systems [25], [26], [7], [20], [30], where massive amount of data arrive continually and the data processing requires to access often valuable personal information. As a consequence the entire monitoring process could put at risk the privacy of people represented in the stream data. Especially, in cases in which the data streams describe some individual activity, revealing some behavior and habit.

In this paper, we study the privacy issues in distributed systems during the monitoring of thresholds functions. We refer to the monitoring model presented in [30] and we provide a privacy-preserving approach suitable for this kind of systems. In particular, we assume a framework where some there are geographically dispersed sites called *nodes* and a central station called *coordinator*. The communication is only allowed between every site and the coordinator.

Assuring privacy protection in this kind of systems is challenging. Many privacy models proposed in literature are inapplicable due to the absence of communications among the nodes and because the communication is always *point-to-point* with the coordinator that is supposed to be untrusted. Moreover, addressing privacy issues also means finding an acceptable trade-off between privacy protection and data quality; in this specific context the goal is harder because we need to consider an additional requirement the *system performance*; in other words, in distributed monitoring systems it is important to preserve efficiency as well as privacy.

In this paper we propose a solution based on the well-know *additive randomization* [2], [6] that is suitable to guarantee privacy *at collection time* without requiring any trusted entity for the data collection. We exploit some results in the literature [18], [19] to bound the possible reconstruction of the perturbed data by an adversary. We test the proposed privacy-preserving framework in a real-world application for the monitoring of customers mobility behaviors in the context of car insurances. In our experiments on real world data coming from GPS devices of private cars, we show that our privacy-preserving framework provides acceptable results in terms of amounts of communications, privacy protection and quality of the global function to be monitored.

The proposed solution is perfectly compatible with the change of perspective towards a user-centric model for personal data management highlighted by the reform of the data protection rules proposed on January 25, 2012 by the EC and the last report of the World Economic Forum [14]. One possible way to achieve a user-centric paradigm is to enable individuals to have the full control for the user on the lifecycle of his personal data (e.g., collection, storage, processing, sharing). Thus, the user has to have an active role into a righteous and fruitful ecosystem based on personal data. Moreover, the user has to have the right to dispose or distribute his data for receiving the desired service with the desired privacy level that reflects the his privacy expectation. This is exactly the basic idea behind our framework.

The remaining of the paper is organized as follows. Section II introduces some background information. Section III provides the details of the distributed monitoring of threshold functions we refer to. Section IV describes the privacy model

and states the problem that we want to address. In Section V we present the details of or privacy-preserving solution. Section VI discusses the correctness of the privacy-preserving monitoring process. Section VIII describes the details of the monitoring of the clustering quality where we want to apply our privacy-preserving method. In Section IX we introduce the application scenario where we test our method and we show our empirical results. In Section X we discuss some work proposed in the literature. Finally, Section XI concludes the report.

II. Preliminaries

In this section we introduce some notions that are important for better understanding the proposed privacy-preserving scheme.

A. Additive Randomization

Randomization methods are used to modify data with the aim of preserving the privacy of sensitive information. They were traditionally used for statistical disclosure control [1] and later have been extended to the privacy-preserving data mining problem [5]. Randomization based approaches use a noise quantity in order to perturb data. The algorithms belonging to this group of techniques first of all modify the data by using randomization techniques. Then, from the perturbed data it is still possible to extract patterns and models. The most famous random perturbation technique is called *additive random perturbation* This method can be described as follows. Denote by $U = \{u_1 \dots u_m\}$ with m records and n attributes, the original dataset. The new distorted dataset, denoted by $\tilde{U} = \{\tilde{u_1} \dots \tilde{u_m}\}$, is obtained drawing independently from a certain probability distribution a noise quantity z_i and adding it to each record $u_i \in U$. The set of noise components is denoted by $Z = \{z_1, \dots, z_m\}$. Most commonly used distributions are the uniform distribution over an interval $[-\alpha, \alpha]$ and Gaussian distribution with mean $\mu = 0$ and standard deviation σ. The original record values cannot be easily guessed from the distorted data as the variance of the noise is assumed enough large. Instead, the distribution of the dataset can be easily recovered. Indeed, if U is the random variable representing the data distribution for the original dataset, Z is the random variable denoting the noise distribution, and $\tilde{U}$ is the random variable describing the perturbed dataset, we have:

$$\tilde{U} = U + Z$$
$$U = \tilde{U} - Z.$$

Notice that, both m instantiations of the probability distribution $\tilde{U}$ and the distribution Z are known. In particular, the distribution Z is known publicly. Therefore, by using one of the methods discussed in [5], [3], we can compute a good approximation of the distribution $\tilde{U}$, by using a large enough number of values of m. Then, by subtracting Z from the approximated distribution of $\tilde{U}$, we can compute an approximation of U. At the end of this process individual records are not available, while obtain a distribution only along individual dimensions describing the behavior of the original dataset U.

B. Spectral Filtering Attack

Kargupta et al. in [24] addressed the problem to extract the real data U from the perturbed data $\tilde{U}$ by knowing only the noise distribution applied to perturb the data. The authors in this paper present an attack capable to separate the data from the noise by using a spectral filtering technique. This attack is based on the observation that the distribution of eigenvalues of random matrices presents some well-known characteristics that can be exploited for the data reconstruction.

In the following we briefly describe the spectral filtering approach. Consider a noise matrix Z with same dimensions as U. The random value perturbation techniques generate a perturbed data matrix $\tilde{U} = U + Z$. The objective of the spectral filtering based approach is to derive the estimation $\hat{U}$ of U from the perturbed data $\tilde{U}$ based on random matrix theory. An explicit filtering procedure is shown below:

Step 1: Calculate the covariance matrix of $\tilde{U}$ by $\tilde{A} = \tilde{U}^T\tilde{U}$.

Step 2: Since the covariance matrix is symmetric and positive semi-definite, we apply spectral decomposition on $\tilde{A}$ to get $\tilde{A} = \tilde{Q}\Lambda\tilde{Q}^T$, where $\tilde{Q}$ is orthogonal matrix whose column vectors are eigenvectors of $\tilde{A}$, and Λ is the diagonal matrix with the corresponding eigenvalues on its diagonal.

Step 3: Derive information of the eigenvalues from the covariance matrix of the noise Z.

Step 4: Extract the first k components of $\tilde{A}$ as the principal components by comparing $\tilde{\lambda_i}$ with eigenvalues of the noise. $\tilde{\lambda_1} \geq \tilde{\lambda_2} \geq \ldots \geq \tilde{\lambda_k}$ are the first k largest eigenvalue of $\tilde{A}$ and $\tilde{e_1}, \tilde{e_2}, \ldots, \tilde{e_k}$ are the corresponding eigenvectors. These eigenvectors form an orthonormal basis of a subspace $\tilde{\mathcal{X}}$. Let $\tilde{X} = [\tilde{e_1}, \tilde{e_2}, \ldots, \tilde{e_k}]$. The orthogonal projection onto $\tilde{\mathcal{X}}$ is $P_{\tilde{\mathcal{X}}} = X\tilde{X}^T$.

Step 5: Obtain the estimated data set using $\hat{U} = \tilde{U}P_{\tilde{\mathcal{X}}}$.

Based on the random matrix theory, we can derive the theoretical bounds of the eigenvalues corresponding to the noise matrix Z as $\lambda_{Z_{min}} = \sigma^2(1 - 1/\sqrt{Q})^2$ and $\lambda_{Z_{max}} = \sigma^2(1 + 1/\sqrt{Q})^2$, where Q is linear to the ratio between the number of records and the number of attributes. As in most data mining applications, the number of records far exceeds that of attributes (hence Q is large), we can see $\lambda_{Z_{min}} \approx \lambda_{Z_{max}} \approx \sigma^2 = \lambda_Z/(m-1)$.

C. Error Lower Bound for spectral filtering based reconstruction methods

Guo et al. in [18], [19] theoretically explore the problem which originates from the usage of additive noise for privacy preservation and give a bound for the reconstruction error and the perturbations in terms of matrix norm. This bound can help data owners to decide how much noise should be added

to satisfy a given threshold of tolerated privacy breach. In other words, they provide an approach for generating the noise matrix Z with the suitable σ^2 value in such a way that a significant differences between $\hat{U}$ and U is introduced and so, a desirable privacy level is guaranteed.

The approach for discovering the lower bound is based on the notion of relative error that is defines as

$$re(\hat{U}, U) = \|\hat{U} - U\|_F / \|U\|_F \tag{1}$$

where $\|.\|_F$ denotes the Frobenius norm.

In [18], [19] authors derive a lower bound for the Singular Value Decomposition (SVD) based reconstruction and show that this bound can be considered valid also for the spectral filtering method. The SVD based reconstruction method estimates U as

$$\hat{U} = \tilde{U}_k = \tilde{L}_k \tilde{D}_k \tilde{R}_k = \sum_{i=1}^{k} \tilde{d}_i \tilde{l}_i \tilde{r}_i{}^T$$

where $\tilde{D}_k$ is the diagonal matrix the diagonal matrix with k principal singular values of $\tilde{U}$ and $\tilde{L}_k$ and $\tilde{R}_k$ contain the corresponding left and right singular vectors. Based on this reconstruction of the original data U the reconstruction error between $\hat{U}$ and U is

$$\|\hat{U} - U\|_F \geq \|U_k - U\|_F.$$

In order to preserve privacy, data owners need to make sure that the relative error is greater than the privacy threshold τ specified before the perturbation. Therefore, we need to have

$$\tau \|U\|_F \leq \|U_k - U\|_F = d_{k+1}^2 + \ldots + d_k^2 \tag{2}$$

Here, k which might be chosen by attackers can be determined by

$$k = max\{i | \tau \leq (d_{i+1}^2 + \ + d_n^2)/\|U\|_F\} \tag{3}$$

For i.i.d. noise based on equation (3) we have that $d_i \geq d_Z$, and the data owner should generate Z such that the eigenvalue of $(Z^T Z)$ satisfies $d_k < d_Z \leq d_{k+1}$. Since d_Z is the eigenvalue of $Z^T Z$, the variance of the noise can be derived $\sigma^2 = d_Z/(m-1)$ where m is the number of rows in Z.

III. Distributed monitoring of threshold functions

In this paper we consider the monitoring problem described in [30] where the specific scope is to solve the problem of monitoring the value of a function computed over data that are distributed in a network. We assume a framework with m geographically dispersed sites (*nodes*) and a central coordinator (*coordinator*) that can communicate with every site, while pairwise site communication is only allowed via the coordinating source. Each site receives a stream of data updates and maintains a s-dimensional local statistics vector $v_i(t)$. The coordinator has to assure that at each time instant t the following condition holds:

$$f(v(t)) \leq T \tag{4}$$

where f is a given function, $f : \mathcal{R}^s \rightarrow \mathcal{R}$, $T \in \mathcal{R}$ is a threshold and $v(t)$ is the weighted average of the $v_i(t)$ of all sites, i.e. $v(t) = (\sum_i w_i v_i(t))/\sum w_i$ for some weights $w_i \geq 0$. While the latter condition is apparently a strong limitation to the applicability of the framework, it has been shown that several interesting problems can be reformulated in this way. A way to do this, which will also be used later in this paper, is a *vector augmentation trick*, consisting in adding to vectors $v_i(t)$ (sent by the sites to the coordinator) one or more extra components. As an example consider the monitoring of the variance of all $v_i(t)$, assuming $s = 1$, i.e. ensure that $var_i(v_i(t)) \leq T$. Clearly, it does not directly fit the form in (4), but we can exploit the well known property $var(X) = avg(X^2) - [avg(X)]^2$ to rewrite our problem as $f(v(t)) = v(t)_1 - [v(t)_2]^2$. This only requires that each site has to communicates a 2-dimensional vector $\left(v_i(t), v_i(t)^2\right)$.

In [30] authors propose the so called *geometric approach* to perform the monitoring of (4). In the following, we provide some details on that method. All the points in $\mathcal{R}^s$ where (4) is satisfied form the *admissible region* G, and our objective is simply to ensure that $v(t) \in G$. The method is based on the following property: the convex hull of a set $\{x_i\}_i \subset \mathcal{R}^s$ of points is entirely contained in $\bigcup_i B(x_i, e)$, where e is any point in $\mathcal{R}^s$ and $B(x_i, e)$ is the ball having the segment $\overline{x_i e}$ as diameter. It is straightforward to see that our $v(t)$ is contained in the convex hull of the set $\{v_i(t)\}_i$. Therefore, if every ball $B(v_i(t), e)$ is contained in the admissible region (namely, it is *monochromatic*), then also $v(t)$ will be, and therefore (4) will be satisfied. Once the coordinator has communicated to all sites the point e, each site will be able to test whether its ball $B(v_i(t), e)$ is monochromatic. As long as no site detects a failure, we are guaranteed to satisfy (4), without any need of communicating information to the coordinator. When a site fails, it notifies the coordinator, who will ask to every site to send their new vector values, and test condition (4). Notice that the test performed on each site might cause false alarms (its ball intersects the inadmissible region, while the overall $v(t)$ is completely inside the admissible one) but not false negatives; in other words, when condition (4) is violated the system will always capture that. In principle the point e can be chosen freely, however it is convenient to compute it as $e = v(t')$, where t' is the time of the most recent *synchronization*, i.e. the phase where every site communicates its new values to the coordinator. Note that Synchronizations usually occur during the set-up phase but also any time there is a site rising an alarm.

A. Safe Zones for convex inadmissible regions

The geometric method discussed above provides a means to decide locally to the node which vector values guarantee that the overall function satisfies the global constraint to monitor. The set of such values is also called *safe zone* and, since it is only based on the global function and on the reference point e, all nodes have the same safe zone.

In [26] it is shown that the safe zones built by the geometric method can also be computed as the intersection of an infinite

set of hyperplanes. Yet, it is also shown that part of them are unnecessary, which makes the safe zones smaller (and thus less effective) than what strictly needed. A particular case is that where the inadmissible region (the set of values that violate the global constraint) is convex. In this situation we can easily find an optimal safe zone in two steps: first, find the point p of the inadmissible region which is closest to the reference point e; second, draw the hyperplane that passes through p and is orthogonal to the segment $\overline{ep}$, and then, of the two half-spaces determined by the hyperplane take as safe zone the one that contains e.

IV. Privacy Model

In Section III we described the computational model that we refer to and we explained that each node observes local update streams and verifies that the local constraint on its stream has not been violated. In case of violation the node has to communicate its value to the coordinator. In this case we can have serious privacy issues especially when each node observes information about a single individual, thus the transmitted vector may contains sensitive information. As an example, in a scenario where the coordinator is responsible for monitoring functions on mobility data the local vector could describe the mobility behavior of a person. An attacker accessing the user vector could learn information such as the typical speed or typical trips.

Moreover, the non-communication from a specific node can reveal sensitive information about the state of the node. Finally, when the node has to communicate to the coordinator means that it is violating a local constraint, and this information itself could be sensitive. *How can we protect this sensitive information?* We think that a suitable method that we can apply in this setting is the *additive randomization* for perturbing the data to be sent. Clearly, the data randomization affects also the safe zone inserting also there an uncertainty.

In our setting, we assume that each node in our system is secure; in other words, we do not consider attacks at the node level. We also assume that the coordinator is untrusted. Therefore, we focus on designing privacy-preserving techniques to defend against an untrusted coordinator. In particular, our goal in this paper is to inscribe privacy protection in the monitoring system (Section III) enabling the distributed monitoring of global functions while preserving the privacy of each node.

In the following we formally define the problem that we want to address.

Definition 1: Let $\{n_1, n_2, \ldots, n_m\}$ be the m nodes of the system. We want to find a privacy-preserving technique such that the following requirements are satisfied:

- individual privacy is guaranteed;
- the system performance, in terms of number of communications, is reasonable;
- the correctness and the quality of the global function f to be monitored is not compromised.

In order to address this problem we propose a method based on the additive randomization of each local vector before sending it to the coordinator.

V. Preserving Privacy in Distributed Monitoring

In this section, we present the algorithm for privacy-preserving distributed monitoring. The basic idea of our approach is to add to the original vector a noise vector where the components are drawn from a Gaussian distribution with mean 0 and standard deviation σ. Then, during the whole process for the geometric-based monitoring, described in Section III, in the system has to be considered the noisy version of each vector. In particular, each node uses the noisy version of the local statistics vector for checking the local constraint and if there is a violation the node transmits it to the coordinator. The coordinator averages all these noisy vectors, and checks whether the function of the global average has crossed the threshold T.

A. Setup Phase.

Our proposal considers an initial phase where each node adds to its initial local statistics vector $v_i(0)$ a noise vector $z_i(0)$ obtaining $\tilde{v_i}(0)$ and sends it to the coordinator, that checks that the global vector computed by using the noisy vectors $\tilde{v_i}(t)$ is within the admissible region; otherwise a global violation is raised. The coordinator defines the initial vector e and communicates it to all sites. At this point each site is able to construct its ball $B(\tilde{v_i}(t), e)$ with radius $\tilde{r_i} = \frac{\|\tilde{v_i}(t)-e\|}{2}$ and center is $\tilde{c_i} = \frac{\tilde{v_i}(t)+e}{2}$. It is immediate to understand that the addition of the noise vector affects the radius and the center of the ball and as a consequence the construction of the safe zone; in other words also the safe zone is randomized.

B. Local Monitoring Phase.

After constructing its ball a node monitors the local statistics vector against that safe zone; in other words, for each time t the node n_i adds a noise vector z_i to the current statistics vector $v_i(t)$ and tests its local constraints, i.e., checks if the perturbed vector $\tilde{v_i}(t)$ is contained in the admissible region (i.e., if the ball $B(\tilde{v_i}(t), e)$ is *monochromatic*). If no violation occurs, the monitoring cycles simply goes on without any communication and any action from the controller's side. If, instead, there is some local violation, the controller has to check whether there is a global violation. In particular, to verify whether the global threshold T was crossed the coordinator requires a *synchronization*, i.e., all the nodes have to transmit their perturbed statistics vectors and then evaluates whether the average of this vector is within the admissible region. In the case that a global breach is detected the coordinator computes a new estimate vector e according to the updated statistics vectors sent by the nodes.

VI. Correctness of the Monitoring

As already stated above, the randomization of each local statistics vector $\tilde{v_i}(t)$ implies the randomization of each ball $B(\tilde{v_i}(t), e)$. In particular, when we add a noise vector z_i (with

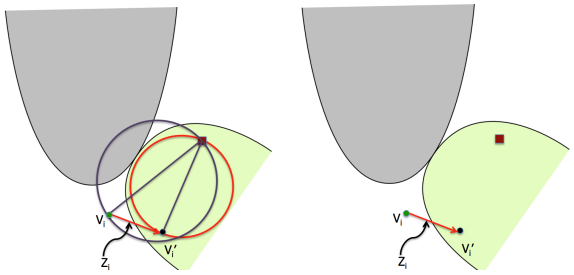


Fig. 1. Missing Alarms caused by the randomization

components drawn by a Gaussian distribution with mean 0 and standard deviation σ) to $v_i(t)$ the diameter of the original ball could increase or decrease and the ball could be also change it position. All these changes can lead to a fake or missing alarms. The first case is due to the fact the a non-monochromatic ball after the randomization could become monochromatic and generate fake violations. In other words, we could increase the false positive alarms and so, the number of communications with respect to the communications required by the monitoring without any privacy protection. The second case represents the opposite situation: a monochromatic ball becomes non-monochromatic with the randomization. This means that the node might not communicate when really happens a violation of the original constraint. In other words, the correctness of the system could be compromised because of missing alarms. This case is represented in Figure 1 where grey area represents the inadmissible zone, the red ball represents the randomized ball while the other ball is the original one. We can observe that the construction of the red ball given the perturbed vector leads to a missing alarm. The same figure in the right side shows what happens in the system in terms of safe zones. We can note that the original vector lies out of the safe zone while the adding of noise moves the vector within the safe zone generating the missing alarm.

In the following we give the correctness guarantees of the privacy-preserving monitoring. In particular, we provide a probabilistic guarantee about *missing alarms*.

Given a vector $\tilde{v}_i(t)$, we know that it is the result of adding noise to each original component drawn by a Gaussian distribution with mean 0 and standard deviation σ. Fixed a probability $1-\delta$, we want to find the minimum radius such that the original vector $v_i(t)$ is one of the points in the area covered by the sphere (in s dimensions) with center $\tilde{v}_i(t)$ and a specific radius r_l; in other words, $||z_i|| = ||v_i(t) - \tilde{v}_i(t)|| \leq r_l$ with probability at least $1-\delta$. In order to do that we can observe that $||z_i||^2$ follows a χ^2_s distribution, and in particular the distribution is $\sigma^2\chi^2_s$.

Given the ball $B(\tilde{v}_i(t), e)$ of the node n_i with center $\tilde{c}_i$, we denote by $dist(\tilde{c}_i, b)$ the distance between $\tilde{c}_i$ and the boundary of the non-admissible region. We are ready to formulate the theorem that states the correctness of the monitoring.

Theorem 1: Given a perturbed local statistics vector if its ball $B(\tilde{v}_i(t), e)$ is monochromatic and $dist(\tilde{c}_i, \tilde{v}_i(t)) + r_l < dist(\tilde{c}_i, b)$ then the probability to have a missing alarm is at most δ.

Proof: As explained above with probability at least $1-\delta$ we have $||v_i(t) - \tilde{v}_i(t)|| \leq r_l$. So, we observe that $dist(\tilde{c}_i, \tilde{v}_i(t)) + r$ represents the radius of the original ball $B(v(t), e)$ with probability at least $1-\delta$. In fact, we have that $dist(\tilde{c}_i, \tilde{v}_i(t)) = \frac{||\tilde{v}_i(t)-e||}{2}$, i.e., it is the radius of the ball $B(\tilde{v}_i(t), e)$ while $\frac{||\tilde{v}_i(t)-e||}{2} + r_l \geq \frac{||\tilde{v}_i(t)-e||}{2} + ||v_i(t) - \tilde{v}_i(t)|| = \frac{||v_i(t)-e||}{2}$, i.e., the original ball will have at most this radius. Since, $dist(\tilde{c}_i, \tilde{v}_i(t)) + r_l < dist(\tilde{c}_i, b)$ we can infer that with probability at least $1-\delta$ the original ball $B(v(t), e)$ is monochromatic and as a consequence the probability of missing alarms (non-monochromatic) ball is at most δ. ■

The above theorem is related to the missing alarms at node level, i.e., missing alarms that each single node can generate with the construction of its ball after the perturbation. Another form of missing alarms are those that we call *global missing alarms*. We have a missing alarm of this kind when the coordinator receives one or more alarms from the nodes, computes the average vector $\tilde{v}(t)$ and it is within the admissible region while the original $v(t)$ would not be within that region. Before providing the theorem that states the probability of global missing in the monitoring process, we note that if each node vector is perturbed by a noise vector with components drawn by a Gaussian distribution $\mathcal{N}(0, \sigma)$, then the average vector is affected by a noise from a Gaussian distribution with standard deviation $\frac{\sigma}{\sqrt{m}}$, where m is the number of nodes in the system. By following the same reasoning as in the case of local missing alarms we have that, given the perturbed average vector $\tilde{v}(t)$, with probability at least $1-\delta$ its original version is within the area covered by the sphere (in s dimensions) with center $\tilde{v}(t)$ and radius r_g. Therefore, we have that $||v(t) - \tilde{v}_i|| \leq r_g$ with probability at least $1-\delta$ and the noise $||v(t) - \tilde{v}_i||^2$ follows the distribution $\frac{\sigma}{\sqrt{m}}^2\chi^2_s$.

In the following we denote by $dist(\tilde{v}(t), b)$ the distance between the global vector $\tilde{v}(t)$ and the boundary of the non-admissible region.

Theorem 2: Given the perturbed global vector $\tilde{v}(t)$, if $r_g < dist(\tilde{v}(t), b)$ then the probability to have a missing alarm is at most δ.

Proof: The proof is straightforward and derives from the observation that with probability at least $1-\delta$ we have $||v(t) - \tilde{v}_i(t)|| \leq r_g$. ■

VII. Protection against Spectral Filtering Attack

In Section II-B we discuss the weakness of the additive randomization. In our setting we assume that an attacker can access the coordinator site, obtain the matrix $\tilde{U}$ where each row is a perturbed node vector $\tilde{v}(t)$. From $\tilde{U}$ the attacker applying the spectral filtering reconstruction obtains $\hat{U}$. The distance between U and $\hat{U}$ represents the privacy protection that we measure by the relative error $re(U, \hat{U})$: greater relative error means more privacy protection. The relative error increases when we increase the magnitude of the noise to be added to the original data; in other word, a Gaussian

distribution with a greater standard deviation σ guarantees more privacy protection. The goal is to find the suitable σ for the noise distribution so that a minimum level of privacy is guaranteed.

In Section II-C we presented the result obtained in [18], [19] related to a lower bound for the relative error after a spectral filtering attack. We also discussed as this bound can be exploited for identifying the standard deviation of the noise distribution to guarantee a minimum level of privacy τ. This methodology is perfect in a centralized system where the data owner has the original matrix U and can find the best k such that with U_k the condition (2) is satisfied and, as a consequence, can identify the best σ of the noise distribution to be used for the perturbation. In a distributed setting, we do not have a global vision of the original vectors and so of the matrix U, thus finding the best σ for the perturbation is not possible. To solve this problem we propose to learn the standard deviation observing the historical data of the nodes N. The idea is to analyze for a long time the data about the nodes in the system and by observing the typical behavior of the data we can learn the standard deviation σ suitable to have the minimum privacy level τ. The learnt values of σ will be used during the monitoring phase. The basic assumption here is that user's behaviors present some typical regularities and we want to exploit them for finding the suitable standard deviation of the noise distribution. In the following we describe the details of the procedure for the learning phase.

For each monitor iteration tp, we consider the matrix $U(tp)$ composed of all the node vectors in the historical data $v_i(tp)$. For each possible value of $k = 1, \ldots, p$ we computes the eigenvalues, namely d_k, d_{k+1} and then d_Z defined as the average of the two eigenvalues, respecting the properties that $d_k < d_Z \leq d_{k+1}$. Finally the value of $\sigma(tp) = \sqrt{d_Z/(m-1)}$ is computed; in the following we denote by $\sigma_k(tp)$ the standard deviation at the iteration tp computed with the value k. Then, we compute the corresponding relative error corresponding to the privacy level guaranteed by the computed σ value; in other words, we compute $re(U(tp), U_k(tp)) = \tau_k(tp)$.

The learnt information, composed of a set of pairs $\langle \sigma_k(tp), \tau_k(tp) \rangle$, can be used by each node during the monitoring phase after setting the global privacy level that we desired to be guaranteed in the system. In particular, given a monitoring iteration tp and the global privacy level to be guaranteed τ the node will draw the noise from the Gaussian distribution with standard deviation $\sigma_k(tp)$ corresponding to minimum the $\tau_k(tp)$ such that $\tau_k(tp) \geq \tau$. Clearly, the learnt information could be used in a different way. As an example, after the learning we could decide to always use the maximum standard deviation found in the historical data. This could bring to use in some steps too much noise that corresponds to a better privacy but also a worst impact on the correctness of the monitoring function.

VIII. Distributed Monitoring of Clustering Quality

In order to evaluate our privacy-preserving method we need to verify in real applications the empirical privacy guarantees, the impact of the privacy approach on the number of communications and on the correctness of the monitoring function. To this aim we decide to apply our privacy-preserving mechanism in the application presented in [28], where the goal is a distributed monitoring of clustering quality.

The measure used to evaluate the quality of a clustering is the simple and very popular *Sum of Squared Error* (SSE in short), defined as follows:

$$SSE = \sum_{i=1}^{h} \sum_{p \in C_i} ||p - c_i||_2^2 \tag{5}$$

where C_i represents the i-th cluster, and c_i is its center (average vector).

The monitoring problem in this setting deals with dynamic data consists in continuously checking whether the last clustering computed is still good enough, recomputing the clusters only in the negative case. This requirement can be easily translated in terms of SSE by asking that the dispersion of the objects within the clusters did not grow, or at least not significantly. That means computing the SSE at each time stamp t, denoted by SSE_t, and test that it stays below some threshold. We refer to this continuous testing with the term monitoring. Finally, such a threshold should take into account the dispersion obtained at the very moment the clusters were created, which we denote with SSE_0 (i.e. time counting starts from the moment the most recent clustering was performed), suggesting to adopt a relative threshold. That is summarized in the following problem definition:

Definition 2 (Cluster Monitoring Problem): Given a clustering $C = \{C_1, \ldots, C_h\}$ having initial SSE equal to SSE_0, and given a tolerance $\alpha \in \mathcal{R}^+$, we require to ensure that at each time instant t the following holds for the SSE of the (dynamic) dataset D_t:

$$SSE_t \leq (1 + \alpha) SSE_0 \tag{6}$$

When that does not happen, a re-computation/update of cluster assignments should be performed.

In [28] a strict version of this problem is also considered with the motivation that SSE describes all the clusters together, aggregating the dispersions of each single clusters, but this does not guarantee that a good SSE implies that each single cluster is compact, since some slightly over-dispersed cluster might be balanced in the sum by some virtuous one that adds very little to the SSE.

In this paper we use the strict variant of the clustering monitoring problem, where the constraints are imposed over each single cluster:

Definition 3 (Strict Cluster Monitoring): Given a clustering $C = \{C_1, \ldots, C_h\}$ having initial SSE equal to SSE_0,

and given a tolerance $\alpha \in \mathcal{R}^+$, we require to ensure that at each time instant t the following holds:

$$\forall_{i=1}^{h}.SSE_t^{(i)} \leq SSE_0^{(i)} + \theta^{(i)} \tag{7}$$

where $SSE_t^{(i)}$ is the contribution of cluster i to the SSE at time t, i.e. $SSE_t = \sum_{i=1}^{h} SSE_t^{(i)}$, and the $\theta^{(i)} \in \mathcal{R}^+$ are fixed thresholds such that $\sum_{i=1}^{h} SSE_0^{(i)} + \theta^{(i)} = (1+\alpha)SSE_0$. When condition (7) is violated, a recomputation/update of cluster assignments should be performed.

The clustering monitor problem (Definitions 2 and 3) can be fitted to the geometric approach described above, by properly rewriting it as a variance monitoring. We observe that the formulation of SSE is very similar to a variance, though on s dimensions. Each cluster C_i, having centroid c_i, contributes to the SSE by the following value:

$$\begin{aligned} SSE^{(i)} &= \textstyle\sum_{p \in C_i} ||p - c_i||_2^2 \\ &= \textstyle\sum_{j=1}^{s} \sum_{p \in C_i} (p^j - avg_{q \in C_i}(q^j)) \\ &= |C_i| \textstyle\sum_{j=1}^{s} var_{p \in C_i}(p^j) \\ &= |C_i| \textstyle\sum_{j=1}^{s} \left[avg_{p \in C_i}((p^j)^2) - \left(avg_{p \in C_i}(p^j)\right)^2 \right] \end{aligned} \tag{8}$$

where p^j represents the j-th component of the s-dimensional vector p. This means that by augmenting the vector $v_i(t)$ of each node with the additional s features $v_{i,1}(t)^2, \ldots, v_{i,s}(t)^2$, we can compute the variance for each component. Actually, we can do slightly better, by aggregating the terms in the last line:

$$SSE^{(i)} = |C_i| \cdot \left[avg_{p \in C_i} \left(||p||_2^2 \right) - ||avg_{p \in C_i}(p)||_2^2 \right] \tag{9}$$

which means that only one additional component is needed, corresponding to $||p||_2^2$ of the node (p represents our $v_i(t)$).

The relation (9) states that the geometric approach can be applied for the monitoring of a single cluster, provided that we have defined a threshold value for it. This represents a solution to the strict version of our monitoring problem (Definition 3). We have an implicit partition of the problem into K separate subproblems, that means having a threshold for each single $SSE^{(i)}$. In [28] propose the fallowing partition of the global SSE; in other words they provide the values for the constants $\theta^{(i)}$ in Definition 3:

$$\forall i.\theta^{(i)} = \beta \left(\alpha SSE_0^{(i)} \right) + (1 - \beta) \left(\alpha \frac{SSE_0}{K} \right) \tag{10}$$

where the parameter $\beta \in [0, 1]$.

For evaluating our privacy-preserving approach we make simpler the task and assume that an initial set of profiles are provided to the coordinator, and in the set-up phase it assigns each node to the profile more similar to it. This assignment generates the initial clustering. The monitoring consists in continuously checking whether this clustering is still good enough. In the negative case a new assignment is computed. In particular, we observe that in our process when a node violates its constraint, communicates the new statistics vector to the coordinator indicating its cluster C_j. The coordinator requires to each node belonging to C_j a synchronization and checks the new value of the $SSE_t^{(j)}$, In case of violation at cluster level the coordinator raises a global violation that requires the re-assignment of the statistics vectors to the initial profiles and so all nodes have to communicate their new vectors. Note that, in this version of the problem we only change the way to compute the clustering because here the initial centroids (profiles) are already provided.

A. Privacy in Distributed Monitoring of Clustering Quality

As explained in Section V, each node before sending its local statistics vector adds a noise vector. In this application each node has to perturb the vector and the additional component thus it has to send the pair $\langle \tilde{v_j(t)}, \widetilde{||v_j(t)||_2^2} \rangle$. The coordinator will use this information to compute each $SSE^{(i)}$; in particular, the formula will be:

$$SSE_t^{(i)} = |C_i| \cdot \left[avg_{v_j(t) \in C_i} \left(\widetilde{||v_j(t)||_2^2} \right) - ||avg_{\tilde{v_j}(t) \in C_i}(\tilde{v_j}(t))||_2^2 \right].$$

We noted that the additional component, i.e.,$|||v_j(t)||_2^2$ is useful for the clustering monitoring but could be used by an attacker for reducing the uncertainty in the vector reconstruction. So, in this particular application we propose to add a semi-trusted entity that has the goal of: 1) receiving all the perturbed additional components from the nodes; 2) computing for each cluster C_i the right component of the $SSE^{(i)}$, i.e, $||avg_{\tilde{v_j}(t) \in C_i}(\tilde{v_j}(t))||_2^2$; and 3) sending this value to the coordinator that in this way can compute the $SSE^{(i)}$ without any information about the single value of each additional component. Note that the additional component alone cannot reveal any private information about the single node.

IX. Application Scenario

The monitoring of clustering quality can be used in different contexts. Here, we consider the particular case of continuous monitoring of the quality of the profiles of similar drivers by using the safe zones approach. This scenario is the same used in [28] where authors identified four categories of measures for the description of the *user driving behavior* in a time window: (i) *basic*, (ii) *space-time distribution*, (iii) *context-aware*, and (iv) *behavioral*.

The first one contains measures, directly computable from the raw GPS traces, describing the *basic* features of the trajectories in the time window. These measures allow understanding the behavior of the car usage:

- **Length**: distance travelled by the user.
- **Duration**: time spent traveling by the user.
- **Count**: number of different user's trips.
- **MaxAcceleration**: maximum user's acceleration.
- **MaxDeceleration**: maximum user's deceleration.

The second category involves more complex measures that capture how the drivers use territory in space and time and in

some way describe the spatial and temporal *user distribution movements*:

- **Avg_Dist_L1**: average distance of the user from his most frequent location L1.
- **Radius_g**: radius of gyration of the user (i.e. the standard deviation from the center of mass of his movements).
- **Radius_g_L1**: radius of gyration w.r.t. to the user's L1.
- **TimeL1L2**: time spent by the user in L1 or L2.
- **EntropyLocation**: entropy of the location frequencies where the user stops.
- **EntropyTime**: entropy of user's travel time frequencies.

The third category is composed of the *context-aware* features, where information about the user's movement is related to the spatial and temporal context in which he moves:

- **EntropyArc**: entropy of road segment frequencies traversed by the user.
- **Phighway**: distance travelled on highways by the user.
- **Pcity**: distance travelled inside urban areas by the user.
- **Length_arc_crowded**: distance travelled on top 20% most crowded road segments.
- **Pnight**: distance travelled during night time (i.e. between 10 p.m. and 5 a.m.) by the user.

The last category focuses on capturing some specific mobility behaviors:

- **PAccelerationDeceleration**: percentage of rapid accelerations/decelerations of the user during his movements.
- **Pover**: how much the user drives over the speed limits.
- **Profile**: how much the user follows his profiles, i.e. trips that he performs frequently.

A. Dataset and data preprocessing

We performed our experiments on measures extracted from both real-world data and synthetic data.

The real-world data are provided by an Italian company called *OctoTelematics* collecting data for insurance purposes. This dataset is composed by GPS observations of 11,470 [1] private cars active in Tuscany in a period of 35 days between June and July 2011. Due some pre-processing (i.e. aggregation and filtering) performed by the device on board the sampling rate is reduced to a observation every 3 minutes and it is not regulated by any policy of synchronization. Moreover, we divided the dataset temporally in order to create a training and a test set: the first week for the training set and the remaining 4 weeks for the test set.

We used the training set for two tasks: (1) extracting the measures presented in the previous section using a time window of 3 days with a time granularity of 15 minutes; and (2) learning the regularity of the drivers to extract the suitable standard deviation to use in the Gaussian distribution for drawing the random noise and assuring a minimum level of privacy.

Concerning the first task we computed the measures and applied a pre-processing on them for making them suitable to the specific use. First of all, due the low sampling rate in the data, some of the measures, described in the above section, cannot be extracted. These measures involve all the acceleration based measurements and thus we have excluded them from our experiments. Once all the measures are computed on the first week, we transformed them into a log scale because some variables follow a skewed distribution. We also normalized the variables through z-score to have zero average and variance equal to 1. Finally, since the idea is to build the *customer profiles* using a clustering method, we have also studied the correlation between the measures, discovering that several strong correlations held. Therefore, we selected a subset of measures to avoid strong biases in successive analyses and, as side effect, this reduced the dimensionality of the dataset. After the above pre-processing the remaining attributes are the following: *Duration*, *Radius_g_L1*, *TimeL1L2*, *EntropyArc*, *Pcity*, *Phighway*, *Pnight*, *Pover* and *Profile*.

Concerning the second task we performed the algorithm for learning suitable standard deviation values for the noise distribution (Section VII). In other words, given the data in the first week we learnt a set of pairs $\langle \sigma_k(tp), \tau_k(tp) \rangle$, can be used by each node during the monitoring phase after setting the global privacy level that we desired to be guaranteed in the system. In our experiments we set the global privacy level at $\tau = 0.25, 0.5, 0.8$.

B. Experimental Evaluation

Now we empirically analyze the effects of the privacy transformation on the number of communications and on the quality of clustering and global function. Moreover, we also measure the privacy guarantees. In our experiments we set the probability of missing alarm to $\delta = 0.01$, this means that we capture possible local and global missing alarms with a probability at least equal to 99.99% and we consider a number of profiles equal to 10.

1) Communication Evaluation: To evaluate the performances of the proposed privacy-preserving approach we consider the amount of communications exchanged between the nodes and the controller and between the nodes and the semi-trusted entity for the communication of the additional component. The communications of the first type are always a vector with s dimensions while the messages of the second type are vectors of 1 dimension. In both cases the channel is a *point-to-point* link between the node and the controller/third

[1] The dataset is available at *kdd.isti.cnr.it/node/493*

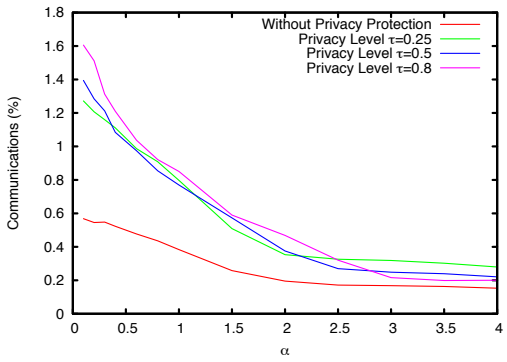


Fig. 2. Communications evaluation by varying α and for different levels of privacy protection.

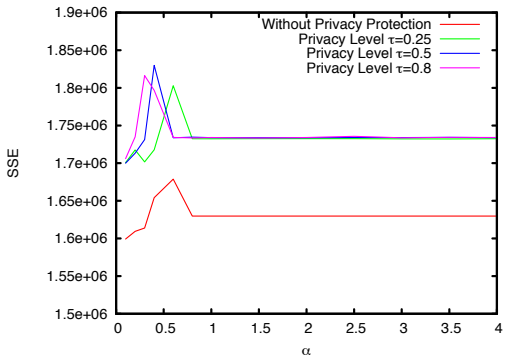


Fig. 3. SSE by varying α and for different levels of privacy protection.

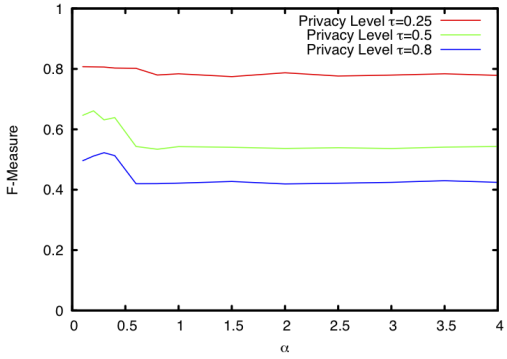


Fig. 4. F-Measure behavior by varying α and for different levels of privacy protection.

party. Here, we do not consider the communications from the controller to the nodes; these communications can be of different sizes and they can use *broadcasting* capabilities of the networks to reach all the nodes at once. The number of communications of this kind are negligible as showed in [28] thus, we decided to not include them in the analysis.

We compare the amount of communications required by the monitoring process without any privacy guarantee and the one required in the system when we use our privacy-preserving method with different levels of privacy. In the privacy-preserving monitoring the number of communications also includes the communications between the nodes and the semi-trusted entity.

The Figure 2 shows the behavior of the communications increasing the α parameter. As expected the number of communications increases with the privacy protection: more privacy requires more communications. This is due to two reasons: 1) in the privacy-preserving approach any time the node has to transmit the vector it has also to transmit the additional component with another transmission, so we have double communications; 2) the randomization can increase the number of false negative alarms. However, we can see that with a reasonable $\alpha = 1.5$ the privacy-preserving approach adds about 30% of communications to the original ones. This is also effect of the double communications due to the third party; indeed without this additional messages we would have a number of communications very similar. We can also note that after the α value of about 2 we have that increasing the level of privacy leads to a decreasing of the communications. This is probably due to the bad effect of a too big value of α in computing Equation (6).

2) Quality of the Clustering Monitoring: We also analyzed the impact of the randomization on the monitored global SSE and on the quality of clusters. Figure 3 shows the behavior of the SSE measure by varying α and with different levels of privacy. We observe that the SSE value increases when the level of privacy in the data is greater; however, the effect of the privacy is reasonable because we have an increasing of about 7% of the original value at worst case.

For evaluating the quality of the obtained clusters we also measured the F-measure, that is the harmonic mean of precision and recall. The recall measures how the cohesion of a cluster is preserved; it is 1 if the whole original cluster is mapped into a single randomized cluster, it tends to zero if the original elements are scattered among several randomized clusters. The precision measures how the singularity of a cluster is mapped into the private version: if the private cluster contains only elements corresponding to the original cluster its value is 1, otherwise the value tends to zero if there are other elements corresponding to other clusters. As expected we have that the increasing of privacy protection reduces the quality of the clusters. This result is depicted in Figure 4 where we vary α and plot for each value the average of F-measure obtained in each monitoring iteration. Finally, we also analyzed how changes the number of re-clustering with the application of the privacy transformation. Figure 5 shows that again the effect of the privacy level is the increasing of the average of re-clustering for each value of α; however, we can note that the increment is negligible.

3) Privacy Guarantee: Finally, we also evaluated the level of privacy guaranteed at coordinator site. We simulated an attack by spectral filtering technique. Our assumption in this experiment is that the attacker has access to the list of learnt values of standard deviation that the nodes used for the noise

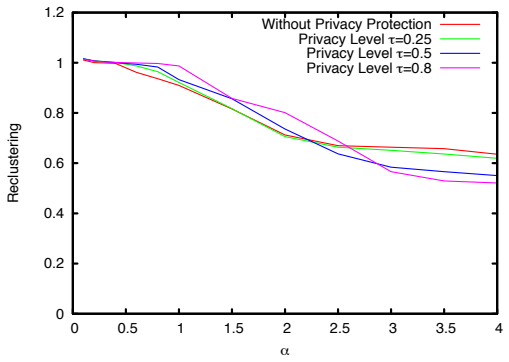


Fig. 5. Number of re-clustering operation required by varying α and for different levels of privacy protection.

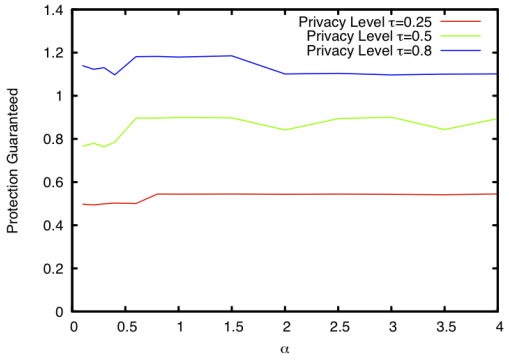
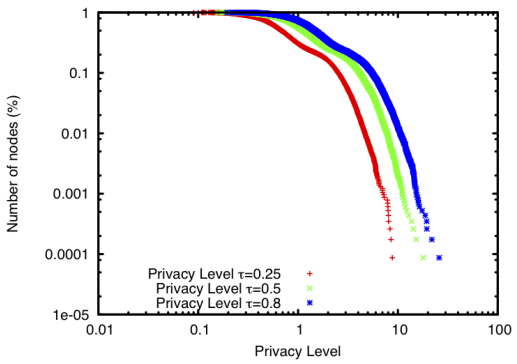


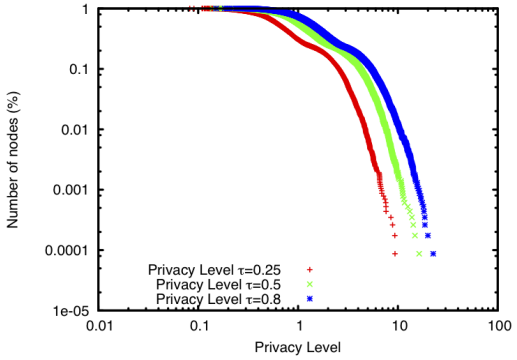
Fig. 6. Global privacy level guaranteed by varying α

distribution. We computed the relative error (Equation 1) any time that the nodes have sent their vectors to the coordinator. Figure 6 depicts the obtained results where we plot for each α the average of the global privacy level guaranteed during the whole process of monitoring. We can observe that the level of privacy provided is much higher w.r.t. the theoretical level of privacy set as a parameter. This is a good result considering that the performance of the system and the correctness of the global function can be considered acceptable even with this high data distortion. In particular, we can observe that setting the global privacy level at 0.25 we have a data protection corresponding to almost 0.5.

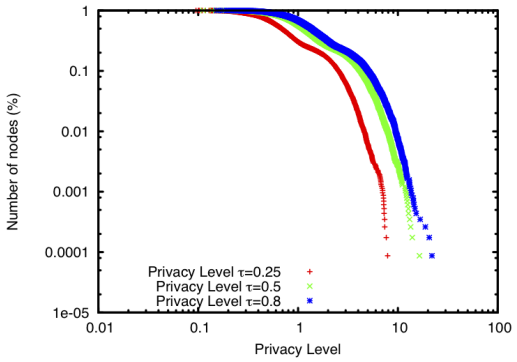
Although the methodology described in Section VII aims at learning the suitable standard deviation for the noise distribution to have a specific global privacy level, we also analyzed the the individual privacy level. In other words, we measured the relative error at record granularity with the following formula: $re_l(\hat{X}, X) = \|\hat{X} - X\|_2 / \|X\|_2$. The three plots in Figure 7 depict the cumulative distribution of the individual privacy level for α values equal to 0.6, 1.5 and 3. We can observe that we obtain a reasonable privacy protection also at individual level; in particular all the plots show a similar result. For example we have that for global privacy $\tau = 0.5$ the 80% of vectors have an individual protection of at least 0.7.



(a) $\alpha = 0.6$



(b) $\alpha = 1.5$



(c) $\alpha = 3$

Fig. 7. Privacy protection a individual level by varying α and τ.

X. Related Work

Various solutions have been proposed to preserve privacy in distributed systems. Some of these solutions propose to share information using the trusted third party services [22]. But in real system sometime it is hard to have a trusted by all entities. In case where the architecture does not consider trusted third party the privacy problem is usually formulated as a variation of the secure multiparty computation (SMC) problem, which has been extensively studied in the literature [16]. However, even if in [17], [34] it has been proved that a general solution to SMC problems exists it has a high computational overhead and thus cannot be efficiently used in practice. By making a

tradeoff between generality and efficiency, different solutions have been proposed to solve various information sharing issues such us intersection and equijoin [21], [4], [15], association rule mining [23], [32], classification [10], [35], top-k queries [33], and statistical analysis [9].

A recent model proposed in the literature is the differential privacy model [12] that provides privacy guarantees against adversaries with arbitrary background information. There are two popular mechanisms to achieve differential privacy, *Laplace* mechanism that supports queries whose outputs are numerical [12] and *exponential mechanism* that works for any queries whose output spaces are discrete [27]. Dwork et al. in [13] recently propose the notion of pan privacy, i.e., how to achieve differential privacy when the adversary is allowed access to the mechanism's internal states. The authors use the pan privacy in the continual counter mechanism [13], [11], and show how to make their counter mechanism resilient against a single unannounced intrusion. A similar problem is addressed in [8]. Rastogi et al. [29] and Chan et al. [31] consider the problem of privately aggregating sums over multiple time periods. Both of them consider untrusted coordinator, in particular, malicious coordinator, and use both encryption and differential privacy for the design of privacy-preserving data aggregation methods.

However, all these works does not consider monitoring systems of thresholding function, where the main goal is to monitor the value of a function and in the same time maintain under control the communications between the nodes and the monitor allowing the communication only when it is necessary.

XI. Conclusion

In this paper, we have proposed a method for inscribing privacy in a distributed monitoring process. Our approach is based on the well-know additive randomization and exploits some results in the literature to bound the possible reconstruction of the perturbed vectors.

We have applied our privacy preserving technique for the monitoring of the clustering quality in a real-world application and we have evaluated the system performance in terms of number of communications, privacy guarantees and quality of the global function to be monitored. The results show that the quality of the monitoring is reasonable while preserving good levels of privacy.

Further investigations will be directed to test our privacy preserving approach in other real-world applications such as the quality monitoring of customer segmentation with respect to their shopping habits in distributed market basket data.

References

[1] Nabil R. Adam and John C. Wortmann. Security-control methods for statistical databases: A comparative study. *ACM Computing Surveys*, 21(4):515–556, 1989.

[2] Charu C. Aggarwal and Philip S. Yu. A survey of randomization methods for privacy-preserving data mining. In Charu C. Aggarwal and Philip S. Yu, editors, *Privacy-Preserving Data Mining*, volume 34 of *Advances in Database Systems*, pages 137–156. Springer, 2008.

[3] Dakshi Agrawal and Charu C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the Twentieth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. ACM, 2001.

[4] Rakesh Agrawal, Alexandre V. Evfimievski, and Ramakrishnan Srikant. Information sharing across private databases. In Alon Y. Halevy, Zachary G. Ives, and AnHai Doan, editors, *SIGMOD Conference*, pages 86–97. ACM, 2003.

[5] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 439–450, 2000.

[6] Shipra Agrawal, Jayant R. Haritsa, and B. Aditya Prakash. Frapp: a framework for high-accuracy privacy-preserving mining. *Data Min. Knowl. Discov.*, 18(1):101–139, 2009.

[7] Chrisil Arackaparambil, Joshua Brody, and Amit Chakrabarti. Functional monitoring without monotonicity. In *ICALP (1)*, pages 95–106, 2009.

[8] T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. *ACM Trans. Inf. Syst. Secur.*, 14(3):26, 2011.

[9] Wenliang Du, Yunghsiang S. Han, and Shigang Chen. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In Michael W. Berry, Umeshwar Dayal, Chandrika Kamath, and David B. Skillicorn, editors, *SDM*. SIAM, 2004.

[10] Wenliang Du and Zhijun Zhan. Building decision tree classifier on private data. In *n Proceedings of the IEEE International Conference on Privacy, Security and Data Mining*, 2002.

[11] Cynthia Dwork. Differential privacy in new settings. In Moses Charikar, editor, *SODA*, pages 174–183. SIAM, 2010.

[12] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2006.

[13] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. Differential privacy under continual observation. In Leonard J. Schulman, editor, *STOC*, pages 715–724. ACM, 2010.

[14] World Economic Forum. Report: Unlocking the value of personal data: From collection to usage. February 2013.

[15] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2004.

[16] Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.

[17] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In Alfred V. Aho, editor, *STOC*, pages 218–229. ACM, 1987.

[18] Songtao Guo, Xintao Wu, and Yingjiu Li. On the lower bound of reconstruction error for spectral filtering based privacy preserving data mining. In *Proceedings of the 10th European conference on Principle and Practice of Knowledge Discovery in Databases*, PKDD'06, pages 520–527, 2006.

[19] Songtao Guo, Xintao Wu, and Yingjiu Li. Determining error bounds for spectral filtering based reconstruction methods in privacy preserving data mining. *Knowl. Inf. Syst.*, 17(2):217–240, November 2008.

[20] Ling Huang, XuanLong Nguyen, Minos N. Garofalakis, Joseph M. Hellerstein, Michael I. Jordan, Anthony D. Joseph, and Nina Taft. Communication-efficient online detection of network-wide anomalies. In *INFOCOM*, pages 134–142, 2007.

[21] Bernardo A Huberman, Matt Franklin, and Tad Hogg. Enhancing privacy and trust in electronic communities. In *Proceedings of the 1st ACM conference on Electronic commerce*, pages 78–86. ACM, 1999.

[22] Nigel Jefferies, Chris J. Mitchell, and Michael Walker. A proposed architecture for trusted third party services. In Ed Dawson and Jovan Dj. Golic, editors, *Cryptography: Policy and Algorithms*, volume 1029 of *Lecture Notes in Computer Science*, pages 98–104. Springer, 1995.

[23] Murat Kantarcioglu and Chris Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE Trans. Knowl. Data Eng.*, 16(9):1026–1037, 2004.

[24] Hillol Kargupta, Souptik Datta, Qi Wang, and Krishnamoorthy Sivakumar. Random-data perturbation techniques and privacy-preserving data mining. *Knowl. Inf. Syst.*, 7(4):387–414, 2005.

[25] Ram Keralapura, Graham Cormode, and Jeyashankher Ramamirtham. Communication-efficient distributed monitoring of thresholded counts. In *SIGMOD Conference*, pages 289–300, 2006.

[26] Daniel Keren, Izchak Sharfman, Assaf Schuster, and Avishay Livne. Shape sensitive geometric monitoring. *IEEE Trans. Knowl. Data Eng.*, 24(8):1520–1535, 2012.

[27] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *FOCS*, pages 94–103. IEEE Computer Society, 2007.

[28] Mirco Nanni, Monreale Anna Trasarti, Roberto, Valerio Grossi, and Dino Pedreschi. Distributed monitoring of cluster quality for car insurance customer segmentation. *Technical Report: TR-13-11, Department of Computer Science, University of Pisa*, 2013.

[29] Vibhor Rastogi and Suman Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In Ahmed K. Elmagarmid and Divyakant Agrawal, editors, *SIGMOD Conference*, pages 735–746. ACM, 2010.

[30] Izchak Sharfman, Assaf Schuster, and Daniel Keren. A geometric approach to monitoring threshold functions over distributed data streams. *ACM Trans. Database Syst.*, 32(4), 2007.

[31] Elaine Shi, T.-H. Hubert Chan, Eleanor G. Rieffel, Richard Chow, and Dawn Song. Privacy-preserving aggregation of time-series data. In *NDSS*. The Internet Society, 2011.

[32] Jaideep Vaidya and Chris Clifton. Privacy preserving association rule mining in vertically partitioned data. In *KDD*, pages 639–644. ACM, 2002.

[33] Jaideep Vaidya and Chris Clifton. Privacy-preserving top-k queries. In Karl Aberer, Michael J. Franklin, and Shojiro Nishio, editors, *ICDE*, pages 545–546. IEEE Computer Society, 2005.

[34] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167. IEEE Computer Society, 1986.

[35] Justin Zhijun Zhan, Stan Matwin, and LiWu Chang. Privacy-preserving naive bayesian classification over horizontally partitioned data. In Tsau Young Lin, Ying Xie, Anita Wasilewska, and Churn-Jung Liau, editors, *Data Mining: Foundations and Practice*, volume 118 of *Studies in Computational Intelligence*, pages 529–538. Springer, 2008.

Privacy-Preserving Mobility Monitoring using Sketches of Stationary Sensor Readings

Michael Kamp, Christine Kopp, Michael Mock, Mario Boley, and Michael May
{michael.kamp, christine.kopp, michael.mock, mario.boley, michael.may}@iais.fraunhofer.de

Fraunhofer IAIS, Schloss Birlinghoven
St. Augustin, Germany

Abstract. Two fundamental tasks of mobility modeling are (1) to track the number of distinct persons that are present at a location of interest and (2) to reconstruct flows of persons between two or more different locations. Stationary sensors, such as Bluetooth scanners, have been applied to both tasks with remarkable success. However, this approach has privacy problems. For instance, Bluetooth scanners store the MAC address of a device that can in principle be linked to a single person. Unique hashing of the address only partially solves the problem because such a pseudonym is still vulnerable to various linking attacks. In this paper we propose a solution to both tasks using an extension of linear counting sketches. The idea is to map several individuals to the same position in a sketch, while at the same time the inaccuracies introduced by this overloading are compensated by using several independent sketches. This idea provides, for the first time, a general set of primitives for privacy preserving mobility modeling from Bluetooth and similar address-based devices.

1 Introduction

Advanced sensor technology and spread of mobile devices allows for increasingly accurate mobility modeling and monitoring. Two specific tasks are crowd monitoring, i.e., counting the number of mobile entities in an area, and flow monitoring between locations, i.e., counting the number of entities moving from one place to another within a given time interval. [1] Both have several applications in event surveillance and marketing [10, 16]. Moreover, matrices containing the flow between every pair of locations (origin-destination, or OD-matrices) are an important tool in many GIS applications, notably traffic planning and management [3].

Today's sensor technologies such as GPS, RFID, GSM, and Bluetooth have revolutionized data collection in this area, although significant problems remain to be solved. One of those problems are privacy concerns. They mandate that, while the count of groups of people can be inferred, inference on an individual

[1] In this paper, we use the term 'flow' always as a short-hand for 'flow between two or more locations'.

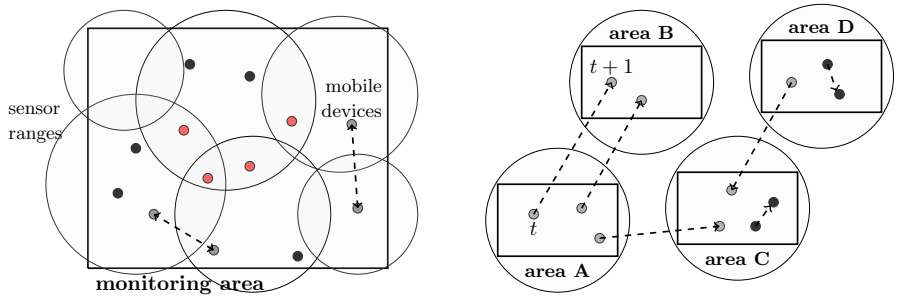


Fig. 1: The two mobility modeling tasks addressed in this paper: (a) *crowd monitoring* and (b) *flow monitoring*.

person remains infeasible. Directly tracing IDs through the sensors violates this privacy constraint, because the amount of information stored allows for linking attacks [12]. In such an attack, sensor information is linked to additional knowledge in order to identify a person and infer upon her movement behavior. Hence, application designers have to design and use new, privacy preserving methods.

The contribution of this paper is to provide a general set of primitives for privacy-preserving mobility monitoring and modeling using stationary sensor devices. Following the *privacy-by-design* paradigm [17, 21], we present a method that stores just enough information to perform the desired inference task and discards the rest. Thereby, privacy constraints are much easier to enforce.

Technically, the method we propose is based on Linear Counting sketches [26], a data structure that allows to probabilistically count the distinct amount of unique items in the presence of duplicates. Linear Counting not only obfuscates the individual entities by hashing, but furthermore provides a probabilistic form of k-anonymity. This form of anonymity guarantees that, by having access to all stored information, an attacker can not be certain on a single individual but can at most infer upon k individuals as a group. Furthermore, Linear Counting is an efficient and easy to implement method that outperforms other approaches in terms of accuracy and privacy on the cost of higher memory usage [19].

The rest of the paper is structured as follows. After discussing related work in section 2, we describe the application scenarios in section 3. In section 4 we present our extension to the linear counting method and give a theoretical analysis of the error. Subsequently, we analyze the privacy of our method in section 5. In section 6 we conduct extensive experiments on the accuracy of Linear Counting and flow estimation under different privacy requirements to test our approach. These experiments have been carried out on a a real-world simulation. Section 7 concludes with a discussion of the results and pointers to future directions.

2 Related Work

The basic tool we are using in this paper are sketches (see Cormode et al. [9] for a good general introduction). Sketches are summaries of possibly huge data

collections that, on the one hand, discard some information for the sake of space efficiency or privacy, but that, on the other hand, still contain enough information to restore the current value of certain variables of interest. Sketches are a very universal tool and have been successfully applied for inferring heavy hitters, moments of a distribution, distinct elements, and more. The general idea of using sketches for privacy is described in Aggarwal and Yu [1] and Mir et al. [20]. The first relates the privacy of a sketch to the variance of the estimator. The latter discusses privacy paradigms that go beyond differential privacy [11] and address security as well. They use various techniques for achieving privacy, notably adding noise. In our approach, we do not employ noise as a source of privacy. However, due to the probabilistic nature of our method, noise has no excessive impact on the accuracy. Hence, adding noise to improve privacy can be combined with our method.

The crucial task in this paper is to count the number of distinct objects at a location (see Gibbons [14] for a general overview on approaches for this problem). The method discussed in our paper, Linear Counting, is first described in [26]. This method can be seen as a special case of Bloom filters for counting [6]. So far, it has received relatively little attention, because it is not as space efficient as log-space approaches such as FM sketches, and traditionally sketches have mainly been used to provide short summaries of huge data collections. However, in the context of privacy preservation in the mobility modeling scenarios of this paper, space is not so much an issue as is *accuracy*. To this end, recently very positive results are reported for comparisons of Linear Counting with FM sketches and other methods [19, 15]. Especially for smaller set sizes that appear in mobility mining, Linear Counting has often an advantage in terms of accuracy. Hence, for our scenario, it is a promising choice.

The idea of using Linear Counting for stationary sensors, specifically Bluetooth measurements, recently has also been described by Gonçalves et al. [15] and similar work for mobile devices is reported in Bergamini et al. [4]. However, here, for the first time, we describe extensions of the basic Linear Count sketches that can be used to monitor flows between locations as well as to compensate for the precision loss incurred for raising privacy.

Our approach for tracking flows is based on the ability to compute the intersection of sketches. A general method for computing general set expressions from sketches is described in Ganguly et al. [13]. Though highly general, the approach is ruled out for our application because the scheme requires to store information at the coordinator which could be used for identifying persons and thus is not privacy aware—it should be noted that it was not build for that purpose.

3 Application Scenarios

In the following, the two application scenarios in mobility monitoring covered by this paper are described. The general setup in both applications is using stationary sensor devices centralizing their sensor readings at a coordinator.

In general, a well-studied approach to monitoring people in an area is to use sensors that count the number of mobile devices in their sensor radius [14,

15], such as Bluetooth, or GSM scanners. From the amount of mobile devices, the actual amount of people can be accurately estimated by assuming a stable average fraction of people carrying such a device [18].

A stationary sensor S scans periodically for devices in its sensor range. Each device is identifiable by its address a from a **global address space** $\mathcal{A}$. The stream of **sensor readings** of a sensor S is defined by $\mathcal{R}_S \subseteq \mathcal{A} \times \mathbb{R}_+$ where $(a,t) \in \mathcal{R}_S$ means that S has read a device address a at time stamp t. For a measurement interval $T = [b,e] \subseteq \mathbb{R}_+$, the readings of sensor S in this time interval is denoted by $\mathcal{R}_S^T = \{a \in \mathcal{A} | \exists t \in T : (a,t) \in \mathcal{R}_S\}$. In both application scenarios, the sensor readings are evaluated to solve the application-specific problem.

3.1 Crowd Monitoring

In major public events, such as concerts, sport events or demonstrations, continuously monitoring the amount of people in certain areas is a key tool for maintaining security. It is vital for the prevention of overcrowding as well as for allocating security and service personnel to the places they are needed most.

To monitor an area using stationary sensor devices, a set of sensors $\mathcal{S} = \{S_1, ..., S_k\}$ is distributed over the area such that the union of their sensor ranges covers the complete area (see fig. 1(a)). For a single sensor S, the **count distinct** of unique entities that have been present in its range during a time interval T is $\left|\mathcal{R}_S^T\right|$. The task is then to continuously monitor at a central site the count distinct of the **union** of all sensor ranges. That is, we aim to monitor $|\mathcal{R}_{S_1}^{T_i} \cup ... \cup \mathcal{R}_{S_k}^{T_i}|$ for consecutive measurement intervals $T_0, T_1, T_2...$ of a fixed time resolution.

Note that the problem of duplicates in the sensor readings cannot be avoided in practice because of several reasons: Covering an area with circular sensor ranges requires overlap and the radius of each sensor range cannot be accurately estimated beforehand. Furthermore, entities can move between sensor ranges during a measurement interval. Thus, independent of the specific application design, summing the distinct counts of individual sensor readings usually overestimate the true number of devices. Without privacy restrictions, this problem can be solved by centralizing the read device addresses or a unique hash of them to eliminate duplicates. However, when addresses or unique identifiers are centralized, devices can be tracked and linked to real persons, thereby violating common privacy constraints.

To solve this problem, we use **Linear Counting** that has been introduced as an accurate and privacy preserving method for estimating the number of distinct items in the presence of duplicates. Linear counting is a sketching technique, i.e., the vector of sensor readings is compressed to a lower dimensional vector, the so called sketch, in such a way that a desired quantity can still be extracted sufficiently accurate from the sketch. The linear count sketch maintains privacy by deliberately compressing the sensor readings with a certain amount of collisions, such that a deterministic inference from a sketch to an address is impossible. A detailed analysis of the privacy aspect is provided in section 5. The distinct amount of people in an area covered by several overlapping sensors is estimated by combining the individual sketches to a sketch of the union of sensor ranges.

The method is described in section 4.1. For that, only the sketches have to be stored and send to the coordinator so that privacy is preserved at a very basic level. This provides two general primitives for monitoring the amount of entities in an area: linear count sketches for privacy-preserving estimation of the **count distinct** of mobile entities in a sensor radius and estimation of the distinct count of entities in an area defined as the **union** of sensor ranges.

3.2 Flow Monitoring

The flow of mobile entities, such as pedestrians, cars or trains, is a key quantity in traffic planning. Streets, rails and public transportation networks are optimized for handling these flows. For a given set of locations, all flows between the locations can be combined in the form of an origin-destination matrix. These matrices are an important tool in traffic management [2].

Flows of mobile entities between a set of locations can be estimated using stationary sensors [3]. For that, given a set of locations of interest, a sensor is placed at each location (e.g., see fig. 1(b)). For a given time period, the flow between two locations denotes the amount of mobile entities that have been present at one location at the beginning of that time period and present at the other location at the end of the period. An entity is present at a location, if it is staying within the range of the sensor placed at that location. Thus, given a time interval at the beginning of the period, T_b, and one at the end, T_e, the **flow** between two sensors S and S' is defined as $v\left(S, S'\right) = |\{a \in \mathcal{A} \mid a \in \mathcal{R}_S^{T_b} \wedge a \in \mathcal{R}_{S'}^{T_e}\}|$. For convenience, we assume that sensor ranges do not overlap. In the case of overlap, this notion of flow has to be extended so that the number of mobile devices that stayed in the intersection is handled separately.

The existing approaches to flow monitoring with stationary sensors rely on tracing unique identifiers through the sensor network. Hence, identifying and tracking a specific device, i.e., a specific person, is possible in monitoring systems as soon as the identifier can be linked to a person. Again, this violates common privacy restrictions. In order to monitor flows in a privacy-preserving manner using linear count sketches, the definition of flow is modified to be able to express the flow as the intersection of sensor readings of different time intervals. Therefore, let T_b and T_e be disjoint time intervals as in the aforementioned definition of flow, then the flow can be expressed as $v\left(S, S'\right) = |\mathcal{R}_S^{T_b} \cap \mathcal{R}_{S'}^{T_e}|$. In section 4.2, we show how local linear count sketches can be combined in a privacy-preserving manner to estimate the size of an intersection.

The extension to paths of arbitrary length is straightforward. Given three consecutive, disjoint time intervals T_1, T_2, T_3, the flow on a path $S_1 \to S_2 \to S_3$ can be represented as $|\mathcal{R}_{S_1}^{T_1} \cap \mathcal{R}_{S_2}^{T_2} \cap \mathcal{R}_{S_3}^{T_3}|$. However, the accuracy of the flow estimation is highly dependent on the size of the intersection compared to the original sets. Moreover, the accuracy drops drastically in the number of intersections, thereby limiting the length of paths that can be monitored. To boost the accuracy and thereby ensure applicability, we present an improved estimator that uses a set of intermediate estimators and their mean value in section 4.3.

An OD-matrix L for a set of locations $\{l_1, ..., l_k\}$ is defined as the flow between each pair of locations, i.e., $L \in \mathbb{R}^{k \times k}$ with $L_{ij} = v\,(l_i, l_j)$. By placing a sensor at each location, that is, sensor S_i is placed at location l_i, an OD-matrix L can be estimated by $L_{ij} = v\,(S_i, S_j)$. This provides another mobility mining primitive: the estimation of flows, paths and OD-matrices based on the **intersection** of sensor readings.

4 Extending Linear Counting

In this section, we present the technical solution to the application scenarios introduced in section 3. We start by recapitulating Linear Counting sketches, which serve as fundamental tool. In particular, for the flow-monitoring it is necessary to extend this sketching technique to monitoring the size of an intersection of two or more sensor readings. For both application scenarios, privacy preservation demands that we use basic sketches at the sensors with relatively high variance in their estimates. This variance even increases when estimating intersections. In order to increase the accuracy again on the output layer, we present an improved estimator that reduces the variance by combining several independent sketches.

4.1 From Sensors to Sketches

Given the sensor readings $\mathcal{R}_S^T = \{a \in \mathcal{A} \mid \exists t \in T : (a,t) \in \mathcal{R}_S\}$ of a sensor S during a time interval T, the goal is to represent the number of distinct devices observed without explicitly storing their addresses. A problem of this kind is referred to as **count distinct problem**, which can be tackled by Linear Counting sketches [26]. They have originally been introduced to estimate the number of unique elements within a table of a relational database.

In our scenario, this means that, instead of storing all readings within a measurement interval T, a sensor just maintains a binary **sketch** $\mathrm{sk}\big(\mathcal{R}_S^T\big) \in \{0,1\}^m$ of some fixed length m. The sketch is determined by a random **hash map** $h : \mathcal{A} \to \{0, \ldots, m-1\}$ such that the following **uniformity** property holds: for all $a \in \mathcal{A}$ and all $k \in \{0, \ldots, m-1\}$ it holds that $\mathbb{P}(h(a) = k) = 1/m$. For practical purposes this can be approximately achieved by choosing, e.g., $h(a) = ((va + w) \mod p) \mod m$, with uniform random numbers $v, w \in \mathbb{N}$ and a fixed large prime number p. Other choices of hash functions are possible (see e.g., Preneel [22]).

A sensor maintains its sketch as follows. At the beginning of the measurement interval it starts with an empty sketch $(0, \ldots, 0)$, and on every address $a \in \mathcal{A}$ read, until the end of the interval, the sketch is updated by setting the $h(a)$-th position to 1. For the whole measurement interval this results in a sketch

$$\mathrm{sk}\big(\mathcal{R}_S^T\big)\,[k] = \begin{cases} 1 & , \text{ if } \exists a \in \mathcal{R}_S^T,\, h(a) = i \\ 0 & , \text{ otherwise} \end{cases} .$$

In our application scenarios, a **global population** of mobile entities $\mathcal{P} \subseteq \mathcal{A}$ of size $|\mathcal{P}| = n$ is monitored with a set of sensors. The size of the global population

n is an upper bound to the number of mobile entities in a sensor reading. In each sensor reading, a subset of the global population is captured, i.e., $\mathcal{R}_S^T \subseteq \mathcal{P}$, thus $|\mathcal{R}_S^T| \leq |\mathcal{P}|$, or $n_S \leq n$ (from now on we denote $\left|\mathcal{R}_S^T\right|$ as n_S).

The number of distinct addresses within a sensor reading can then be estimated based on the sketch as follows. Assume a sensor reading $\mathcal{R}_S^T$ with $\left|\mathcal{R}_S^T\right| = n_S$ and the respective sketch $\mathrm{sk}\left(\mathcal{R}_S^T\right)$. Let $\mathbf{u}_S$ denote the number of zeros in the sketch and $\mathbf{v}_S = \mathbf{u}_S/m$ the relative zero count. Now, the maximum likelihood **count estimator** for the number of distinct items n_S is $\widehat{n}_S = -m \ln \mathbf{v}_S$. Whang et al. [26] shows that the expected value, and the variance of this estimator are asymptotically well-behaved. Here, asymptotically refers to the **limit** for increasing n while the **loadfactor** $t = n/m$ and the **relative size** $c_S = n_S/n$ of S are kept constant. With this notion of limit, that we simply denote by lim for the remainder of this paper, the expected value and the variance can be expressed as

$$\lim \mathbb{E}[\widehat{n}_S] = n_S + (e^{n_S/m} - n_S/m - 1)/2 = n_S \tag{1}$$

$$\lim \mathbb{V}(\widehat{n}_S) = m\left(e^{n_S/m} - n_S/m - 1\right) \quad , \tag{2}$$

respectively. Hence, asymptotically, the estimator is unbiased and has a bounded variance. Standard concentration inequalities can be used to convert this result into probabilistic error guarantees.

In the crowd monitoring scenario, the size of the global population n can be estimated as the size of the union of all individual sensor readings. By construction of the sketches, it is possible to build a sketch of the union of sensor readings $\mathcal{R}_S^T$ and $\mathcal{R}_{S'}^T$ by combining the individual sketches with the point-wise binary OR operation (i.e., $\mathrm{sk}\left(\mathcal{R}_S^T\right)[k] \vee \mathrm{sk}\left(\mathcal{R}_{S'}^T\right)[k]$ is equal to 1 if and only if $\mathrm{sk}\left(\mathcal{R}_S^T\right)[k] = 1$ or $\mathrm{sk}\left(\mathcal{R}_{S'}^T\right)[k] = 1$). The following statement holds:

Proposition 1 (Whang et al. [26]). *Let $\mathcal{R}_{S_1}, ..., \mathcal{R}_{S_k}$ be readings of a set of sensors $S = S_1, ..., S_k$. The sketch constructed from the union of these readings can be obtained by calculating the binary or of the individual sketches. That is,*

$$sk\left(\bigcup_{i=1}^{k} \mathcal{R}_{S_i}\right) = \bigvee_{i=1}^{k} sk(\mathcal{R}_{S_i}) \quad .$$

This is already sufficient to continuously track the total number of distinct addresses in the crowd monitoring scenario: for a pre-determined time resolution, the sensor nodes construct sketches of their readings, send them to a monitoring coordinator, and start over with new sketches. As required by the application, the coordinator can then compute the estimate of distinct counts of mobile entities based on the OR-combination of all local sketches. However, for the flow-monitoring scenario we have to be able to compute the number of distinct addresses in the intersection of sensor readings. Therefore, we have to extend the Linear Counting approach.

4.2 Intersection Estimation

In the following, a method is presented for estimating the intersection of two sets using linear count sketches. The sketch of the intersection cannot be constructed from the individual sketches (note that using the binary 'and' operation on the two sketches does in general not result in the correct sketch of the intersection). Therefore, this method is based on the inclusion-exclusion formula for sets. That is, we can express the size of the intersection of two sets A, B as $|A \cap B| = |A| + |B| - |A \cup B|$. The estimator for the intersection of two sensor ranges is defined in a similar way, using the estimators for each sensor and their union. Let $\widehat{n}_S, \widehat{n}_{S'}$ denote the estimator for $\left|\mathcal{R}_S^T\right|$, respectively $\left|\mathcal{R}_{S'}^T\right|$. Let furthermore $\widehat{n}_{S\cup S'}$ denote the estimator based on the sketch of the union of the sensor readings $\mathcal{R}_S^T$ and $\mathcal{R}_{S'}^T$ as defined in proposition 1. Then the **intersection estimator** is defined as

$$\widetilde{n}_{S,S'} = \widehat{n}_S + \widehat{n}_{S'} - \widehat{n}_{S\cup S'} \tag{3}$$

It turns out that also this estimator asymptotically is unbiased and has a bounded variance. The first follows directly from the linearity of the expected value. Thus, we can note:

Proposition 2. *For a constant loadfactor t and constant fractions c_S, c'_S, the estimator $\widetilde{n}_{S,S'}$ is asymptotically unbiased, i.e.,* $\lim \mathbb{E}[\widetilde{n}_{S,S'}]/|S \cap S'| = 1$.

Furthermore, we can bound the variance of our estimator by the variance of the union estimator. This implies that resulting probabilistic error guarantees become tighter the closer the ratio $|S \cap S'|/|S \cup S'|$ is to one.

Proposition 3. *Asymptotically, the variance of the intersection estimator $\widetilde{n}_{S,S'}$ is bounded by the variance of the count estimator for the union, i.e.,* $\lim \mathbb{V}(\widetilde{n}_{S,S'}) \leq \lim \mathbb{V}(\widehat{n}_{S\cup S'})$.

Proof (sketch). For some subset $A \subseteq \mathcal{P}$ and a fixed sketch position $k \in \{0, \ldots, m-1\}$ let us denote by $p_A = \mathbb{P}(sk(A)[k] = 0)$ the probability that the sketch of A has entry 0 at position k. Due to the uniformity of the hash function h it holds that $p_A = (1 - 1/m)^{n_A}$. The limit of this probability $p_A^* = \lim p_A$ is equal to $\lim (1 - t/n)^{nc_A} = e^{-t \cdot c_A}$. The variance of the intersection estimator can be re-expressed in terms of the covariances σ of the individual count estimators:

$$\begin{aligned}\mathbb{V}(\widetilde{n}_{S,S'}) &= \mathbb{V}(\widehat{n}_S + \widehat{n}_{S'} - \widehat{n}_{S\cup S'}) \\ &= \mathbb{V}(\widehat{n}_S) + \mathbb{V}(\widehat{n}_{S'}) + \mathbb{V}(\widehat{n}_{S\cup S'}) + 2\sigma(\widehat{n}_S, \widehat{n}_{S'}) \\ &\quad - 2\sigma(\widehat{n}_S, \widehat{n}_{S\cup S'}) - 2\sigma(\widehat{n}_{S'}, \widehat{n}_{S\cup S'}) \ .\end{aligned} \tag{4}$$

In order to determine the limit of the covariances for the count estimators $\hat{n}_A$ and $\hat{n}_B$ for some arbitrary subsets $A, B \subseteq \mathcal{P}$ we can use Whang et al. [26, Eq. (8)] and the bi-linearity of the covariance

$$\begin{aligned}\lim \sigma(\hat{n}_A, \hat{n}_B) &= \sigma(m\,(tc_A - \mathbf{v}_A/p_A^* - 1), m\,(tc_B - \mathbf{v}_B/p_B^* - 1)) \\ &= m^2\sigma(\mathbf{v}_A, \mathbf{v}_B)/(p_A^* p_B^*) = m^2\sigma(\mathbf{u}_A/m, \mathbf{u}_B/m)/(p_A^* p_B^*) \\ &= \sigma(\mathbf{u}_A, \mathbf{u}_B)/(p_A^* p_B^*) \ .\end{aligned} \tag{5}$$

Let $\overline{\mathrm{sk}}(A)\,[k]$ denote the binary negation of sketch position k. The co-variances of the absolute number of zero entries $\mathbf{u}_A$ and $\mathbf{u}_B$ is

$$\begin{aligned}\sigma(\mathbf{u}_A, \mathbf{u}_B) &= \sum_{k=1}^{m}\sum_{l=1}^{m} \sigma\big(\overline{\mathrm{sk}}(A)\,[k], \overline{\mathrm{sk}}(B)\,[l]\big) \\ &= \sum_{k=1}^{m} \sigma\big(\overline{\mathrm{sk}}(A)\,[k], \overline{\mathrm{sk}}(B)\,[k]\big) + \sum_{\substack{k,l=1 \\ k\neq l}}^{m} \sigma\big(\overline{\mathrm{sk}}(A)\,[k], \overline{\mathrm{sk}}(B)\,[l]\big)\ .\end{aligned} \tag{6}$$

Let $U = A \cup B$ and $I = A \cap B$. We state without proof that the co-variances of the individual sketch positions are given by

$$\begin{aligned}\sigma\big(\overline{\mathrm{sk}}(A)\,[k], \overline{\mathrm{sk}}(B)\,[k]\big) &= p_U - p_A p_B \\ \sigma\big(\overline{\mathrm{sk}}(A)\,[k], \overline{\mathrm{sk}}(B)\,[l]\big) &= p_{A\setminus B}\, p_{A\setminus B}(1-2/m)^{|I|} - p_A p_B\end{aligned}$$

for the cases $k = l$ and $k \neq l$, respectively. From this and Eq. (6) it follows that

$$\begin{aligned}\lim \sigma(\mathbf{u}_A, \mathbf{u}_B) &= m\,(p_U^* - p_A^* p_B^*) + m(m-1)\left(p_{A\setminus B}^* p_{B\setminus A}^* (1-2/m)^{|I|} - p_A^* p_B^*\right) \\ &= m\left(e^{-t(c_A+c_B-c_I)} - e^{-t(c_A+c_B)} - c_I t e^{-t(c_A+c_B)}\right)\ ,\end{aligned}$$

where the second equality follows from several steps of elementary calculus that we omit here. By Eq. (5) we can then conclude

$$\lim \sigma(\hat{n}_A, \hat{n}_B) = m\left(e^{tc_I} - tc_I - 1\right) = \lim \mathbb{V}(\hat{n}_I)$$

Inserting this result in Eq. (4), and noting that for fixed $c_A \leq c_B$ it holds that $\lim \mathbb{V}(\hat{n}_A) \leq \lim \mathbb{V}(\hat{n}_B)$ (see eq. (2)), in particular $\mathrm{Var}[\widehat{n}_{S\cap S'}] \leq \mathrm{Var}[\widehat{n}_S]$ as well as $\mathrm{Var}[\widehat{n}_{S\cap S'}] \leq \mathrm{Var}[\widehat{n}_{S'}]$, yields

$$\begin{aligned}\lim \mathbb{V}(\widetilde{n}_{S,S'}) &= \lim(\mathbb{V}(\widehat{n}_{S\cup S'}) + 2\mathbb{V}(\widehat{n}_{S\cap S'}) - \mathbb{V}(\widehat{n}_S) - \mathbb{V}(\widehat{n}_{S'})) \\ &\leq \lim(\mathbb{V}(\widehat{n}_{S\cup S'}) + 2\mathbb{V}(\widehat{n}_{S\cap S'}) - \mathbb{V}(\widehat{n}_{S\cap S'}) - \mathbb{V}(\widehat{n}_{S\cap S'})) \\ &= \lim \mathbb{V}(\widehat{n}_{S\cup S'})\end{aligned}$$

□

When estimating the flow, the intersection of the readings of sensor S in a time interval T_b are intersected with the readings of sensor S' in consecutive time interval T_e. Let ΔT denote the time period between those two intervals. Then we denote the estimator for the flow between S and S' for this time period as $\widetilde{n}_{S,S'}^{\Delta T}$. This method can straight-forwardly be extended to paths. The flow on a path $S_1 \to S_2 \to S_3$ can be represented as $|\mathcal{R}_{S_1}^{T_1} \cap \mathcal{R}_{S_2}^{T_3} \cap \mathcal{R}_{S_3}^{T_3}|$. This quantity can again be estimated using the inclusion-exclusion formula.

$$\widetilde{n}_{S_1 S_2, S_3}^{\Delta T} = \widehat{n}_{S_1} + \widehat{n}_{S_2} + \widehat{n}_{S_3} - \widehat{n}_{S_1\cup S_2} - \widehat{n}_{S_1\cup S_3} - \widehat{n}_{S_2\cup S_3} + \widehat{n}_{S_1\cup S_2\cup S_3}$$

The drawback of estimating the flow on paths is that the accuracy decreases drastically in the number of nodes on the path. In conclusion, we now have two major sources of high variance. A high loadfactor that is necessary to comply

with high privacy requirements and a large number of intersections, required to monitor long paths. In the following a method for reducing the variance of the estimators is presented that improves the estimation of count distinct at a single sensor, as well as union and intersection estimation. Through this improvement, a higher loadfactor can be chosen to increase privacy while maintaining the same estimation accuracy. Furthermore, this improvement allows for monitoring the flow on longer paths with sufficient accuracy.

4.3 Improved Estimator

The improved estimator is based on the idea that the average of independent estimations of the same quantity is again an equally biased estimator with lower variance [7]. Hence, at each sensor, not one sketch is constructed, but r different sketches using r different and independent hash functions. This yields r different intermediate estimates, $\widehat{n}^1, ..., \widehat{n}^r$. The improved estimator is then defined as the mean of these intermediate estimates, i.e., $\widehat{\eta} = \frac{1}{r}\sum_{i=1}^{r}\widehat{n}^i$. The $\widehat{n}^1, ..., \widehat{n}^r$ are maximum likelihood estimators for count distinct and as such they are normally distributed and independent with common mean and variance [24], i.e., for all $i \in \{1, ..., r\}$ it holds that $\widehat{n}^i \sim \mathcal{N}\left(\mathbb{E}[\widehat{n}], \mathbb{V}(\widehat{n})\right)$. Thus, the improved estimator is normally distributed with $\widehat{\eta} \sim \mathcal{N}\left(\mathbb{E}[\widehat{n}], \frac{1}{r}\mathbb{V}(\widehat{n})\right)$. The improved estimator has the same expected value as the intermediate estimates, that is, it is asymptotically unbiased, whereas the variance of the improved estimator is reduced by a factor of $1/r$. Furthermore, because the intermediate estimators are normally distributed and asymptotically unbiased, the improved estimator based on their mean is not only again a maximum likelihood estimator for the count distinct, it is also the uniformly minimum variance unbiased estimator and the minimum risk invariant estimator [23].

However, in the pathological event that a sketch becomes full, i.e., $\mathbf{u}_n = 0$, the estimate for the count distinct based on this sketch is infinity. If only one of the r sketches runs full, the estimator fails. This drawback can be circumvented by using the median of the intermediate results instead of their mean. The median is very robust to outliers but has also weaker error guarantees, i.e., to guarantee an error not larger than ϵ with probability $1 - \delta$, the mean estimator requires $r \geq z_{1-\delta}\sqrt{\mathbb{V}(\widehat{n})}/\epsilon$, the median method requires $r \geq \log(1/\delta)/\epsilon^2$ [8] intermediate estimators. Consequently, for $\epsilon < 1$, the mean estimator requires less intermediate estimates to be as accurate as the median method.

5 Privacy Analysis

The main threat to privacy in the presented application scenarios is the so called linking attack, i.e., an attacker infiltrates or takes over the monitoring system and links this knowledge to background information in order to draw novel conclusions. For example, in a standard monitoring system that distributes the sensor readings, i.e., the device addresses, an attacker that knows the device address of a certain person as background knowledge, and furthermore infiltrates the monitoring system, is able to track this person throughout the monitored area.

Sketching prevents these linking attacks in two ways, obfuscation and k-anonymity. Obfuscation is accomplished by hashing the device address to sketch positions. Hence, before an attacker is able to re-identify a device, she has to infer the employed hash function. However, this very basic obfuscation technique can be vanquished using statistical analysis on sensor readings. The second anonymization technique is accomplished by the natural property of sketches to compress the address space, implicating collisions of addresses when mapped to sketch positions. Whereas these collisions entail a loss in accuracy, they create a form of anonymity, because an attacker can only infer upon a set of devices whose addresses are all mapped to that very same sketch position.

Formally, a monitoring system guarantees k-anonymity (see Sweeney [25]), if every monitored entity is indistinguishable from at least k other entities. Using linear count sketches with a loadfactor t results in t collisions per bucket on expectation, as implied by the uniformity property of the hash function, i.e.,

$$\forall i \in \{0, ..., m-1\} : \mathbb{E}\left[\left|\{a \in \mathcal{R}_S^T : h(a) = i\}\right|\right] = t \ .$$

Hence, the expected level of anonymity is t. We denote this form of anonymity **expected k-anonymity**, because the number of collisions is not deterministically guaranteed as required by regular k-anonymity. For a mathematical derivation of a similar probabilistic guarantee in the context of Bloom filters the reader is referred to Bianchi et al. [5].

The union of sensor readings is estimated by the binary or of the individual sketches. The binary or of a set of sketches has a loadfactor at least as high as the individual sketches themselves. Therefore, the level of expected k-anonymity is at least as high.

The intersection of sensor readings can contain far less device addresses than the individual readings. A sketch that is created on the readings of the intersection has thus a lower loadfactor. Even so, the intersection estimator presented in this paper is based on the estimators of the individual sketches and their union; the sketch of the intersection is not constructed at all. Therefore, the level of k-anonymity of this estimator for the intersection is again at least as high as the anonymity of the individual sketches.

6 Experiments

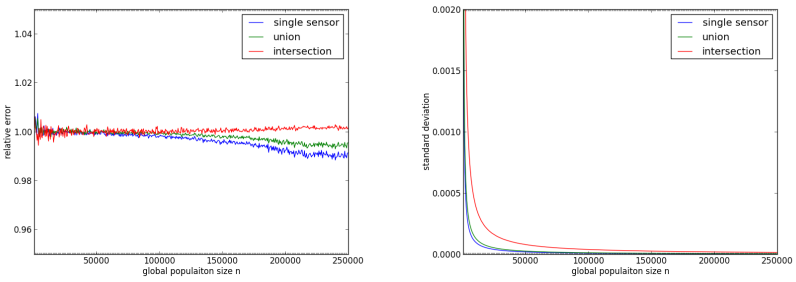
In this section the empirical analysis of our method is presented. The general set up of experiments is as follows. A set of n addresses is randomly sampled out of a pre-defined address range ($\mathcal{A} = \{1, \cdots, 5 \times 10^7\}$). Out of this set we repeatedly sample with duplicates. The set is partitioned into k subsets $S_1, \cdots, S_k$ where S_i represents the sensor readings of sensor i. For each sensor S_i, a sketch sk_i of size m is generated using a global hash function h for all sensors. The estimate of sketches, their unions and intersections are then calculated as explained above.

6.1 Properties of the Estimator

For the first experiment, we simulate 3 sensors and vary the number of persons inside the sensor range from 500 to 250,000. Results for the average ratio of

estimator and true value and the standard deviation of the ratio are shown in fig. 2(a) and fig. 2(b), respectively. The estimate is highly accurate—the error is always below 1%. Compared to the error introduced by the inference of the number of persons present in the area from the number of active Bluetooth devices [18], the error is negligible.

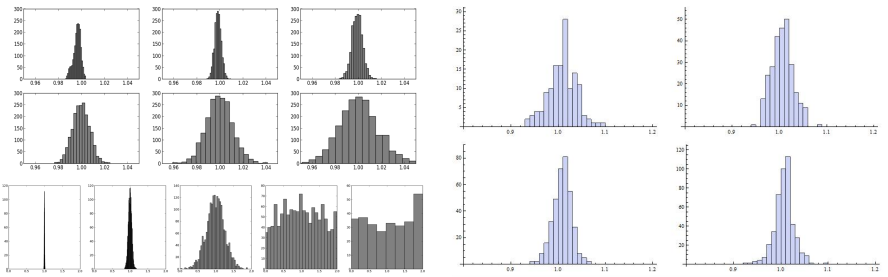
For simple Linear Counting, these results confirm existing expectations from theory and experimental studies. We need not go into a detailed comparison with other sketching methods in this paper, since two recent studies [19, 15] have done that already. One of the basic findings in these studies is that Linear Counting gives, using a suitable loadfactor, much more accurate estimates of the number of distinct objects than other sketching methods, e.g. FM-sketches or sampling based methods. This holds especially for small set sizes—where a number of 10,000 might already be considered small. For our application scenario this is important, since the size—especially of the intersections—can decrease to a few hundred persons. The experiment goes beyond the existing studies by showing that for the intersection of two sets the error can also be very low with a suitable loadfactor and that we can always set up a very accurate estimator using Linear Counting. A significant error of the estimator comes in only because we deliberately trade privacy against accuracy. As shown in section 5 the basic mechanism responsible for privacy is increasing the loadfactor. We analyze this trade-off, i.e., we investigate the impact of the loadfactor on the accuracy of estimates of one sensor as well as of intersections of up to five sensors. We simulate 5 sensors, and vary the loadfactor and the number of intersections. We average the results over 2,000 runs. The results are depicted in figure 3(a).



(a) (b)

Fig. 2: Average estimate $\widehat{n}$ (a) and average standard deviation (b) relative to the true value n for a loadfactor of 1.

The results confirm that the standard error increases with the loadfactor (fig. 3(a), upper part), and even more rapidly with the number of intersections (fig. 3(a), lower part). From this experiment we conclude that simple Linear Counting is indeed suitable for loadfactors smaller than 2 and intersection of at most two sensors. But for higher loadfactors or more intersections the trade-off can become unacceptable. This finding motivates the improved estimator investigated in the following.



(a) (b)

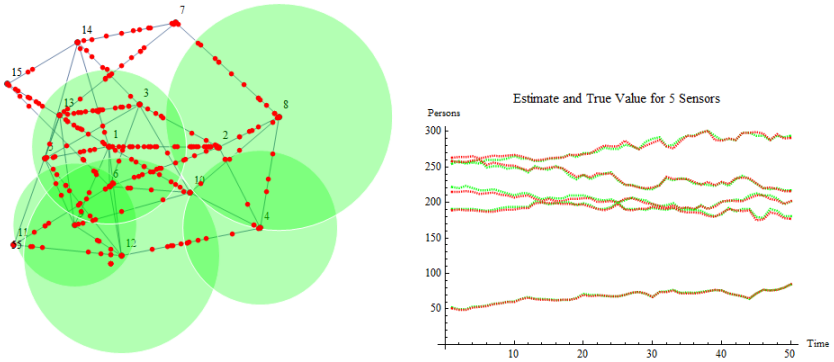
Fig. 3: (a) Distribution of estimates for loadfactors $0.5, 1, 2, 3, 4, 5$ (upper part) and numbers of sensors intersected, $1, 2, 3, 4, 5$ with a constant loadfactor of 5 (lower part). (b) Distribution of estimates using improved estimator with number of intermediate estimates r set to 5 (upper left), 15, 30 and 50 (lower right).

For that, we concentrate on a loadfactor of 5, because in regular k-anonymity 5 is a common value to ensure privacy. For each sensor, we take $r \in \{5, 15, 30, 50\}$ different random initializations of the hash function, resulting in r different sketches per sensor. Results, averaged over 500 runs, are shown in Fig. 3(b). The mean of estimates reduces the variance and is close to the true value. A good trade-off between the increase in accuracy and the higher memory requirements for storing multiple sketches is in the range between 15-30 sketches, since for higher numbers of sketches the variance reduction per additional sketch becomes insignificant. With these results we have demonstrated how to achieve a good trade-off between accuracy, privacy level, and memory consumption.

6.2 Real-World Simulation

To investigate the flow and crowd monitoring in a more realistic setting, we implemented a simulation environment as follows. A random graph of k nodes with Bernoulli Graph Distribution ($p = 0.4$) is generated; the position of nodes in 2D-space is calculated using an automatic graph layout method. A number s of node locations is attached with sensors with a predefined range. In general, a sensor may cover more than one node and several edges. A number of n objects, i.e. the global population, is created. For each object a random sequence of tour stops (nodes of the graph) is generated. For every pair of tour stops the shortest path is determined using Dijkstra's method and inserted into the sequence between the stops. Finally, to each object a velocity, starting time and a step size is assigned (the latter because objects are not only at the node positions, but travel along the edges). During the simulation, for each time step the objects follow the tour with the assigned velocity and starting time, and their position along the edges is calculated. Each sensor monitors at each time step the objects in its sensor range. For each sensor and time period a new sketch is calculated and stored. As a ground truth, also the object address are stored. The simulation stops when the last object has completed its tour.

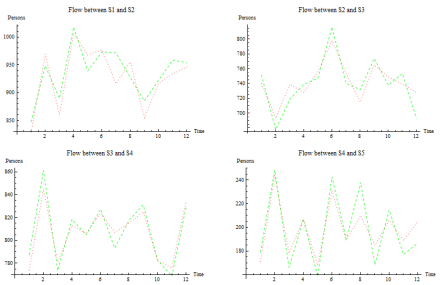
For the crowd monitoring scenario, overlapping sensors are simulated. Fig. 4(a) shows a snapshot from a simulation run. For the flow monitoring we use non-overlapping areas. The main difference compared to the experiments discussed in the last section is that the distribution of objects at nodes is not independent from each other because of flow constraints along the graph. The distribution is generated by a process very similar to real traffic flow, so that we have realistic flow properties over time. Fig. 4(b) shows an example for crowd monitoring



(a) (b)

Fig. 4: (a)Crowd simulation with Random Bernoulli Graph, 1000 persons (red dots), 5 Bluetooth sensors with overlapping range. (b) Estimated value (green dashed) and true value (red dotted) for each sensor for the first 50 time steps.

with 15 nodes, 5 sensors, 1000 objects and a loadfactor of 5. Evidently, the estimates closely tracks the true values, as expected from the theoretical analysis and the experiments reported in the last section. For the flow monitoring



(a)

	S1	S2	S3	S4	S5
S1	0.005	0.008	0.007	0.018	0.001
S2	0.026	0.026	0.01	0.006	0.015
S3	0.004	0.003	0.016	0.002	0.014
S4	0.02	0.03	0.011	0.004	0.012
S5	0.014	0.005	0.004	0.017	0.027

(b)

Fig. 5: Flow estimation (green) and true values (red) between four sensors (a) as well as relative error of an OD-matrix between five sensors (b). Results are from flow simulation with 20000 persons and 5 sensors.

scenario, Figure 5(a) shows the estimate (green, dashed) and true (red, dotted) value of the flows between 4 sensors over time. Next we simulated the OD-matrix construction problem. Table 5(b) shows the relative error of an OD-matrix construction for 20,000 persons moving over a period of 15 time steps in a system

with 12 nodes, tracked by 5 sensors. Each sensor uses the improved estimator with 30 sketches and a loadfactor of 5. From these results we conclude that even for moderately sized sets, the error is very low. Overall, we conclude from the experiments that Linear Counting behaves in the more complex setting of the simulation as expected from a theoretical point of view and is a very promising approach for deployment in real-world applications.

7 Discussion

In this paper we present a new, privacy aware method for estimating flows and tracks as well as for estimating OD-matrices. This way, we extended the Linear Counting approach to a general set of primitives for privacy-preserving mobility modeling. We show theoretically and empirically that two challenging application scenarios can be solved using and combining this set of primitives. To compensate the accuracy deficit of Linear Counting for strict privacy requirements we present a method for increasing the accuracy, while maintaining the privacy. This method is also applicable to boost accuracy in flow estimation, allowing to monitor even tracks.

In contrast to many privacy-preserving approaches, this one is easy to implement, has excellent accuracy and can be implemented efficiently. Our experiments suggest that it is immensely useful in a practical settings and can have a real impact on how stationary sensor based data collection is done.

While being accurate on count distinct and flow estimation, even for strict privacy, the accuracy of our method drops drastically with the length of monitored tracks. The improved estimator can compensate this drop to a certain level. However, experiments show that estimating tracks of length greater 5 leads to large errors. Therefore, we recommend using our method on count distinct and flows. When monitoring tracks, depending on their length, a user might have to reduce privacy requirements in order to maintain a certain accuracy.

The main drawback of Linear Counting when compared to other sketching techniques is the memory usage. Most sketching techniques, e.g., FM Sketches, use memory logarithmic in the number of items it estimates. The linear count sketches, however, have linear memory usage, leading to potentially large sketches. Fortunately, stationary sensors usually can be equipped with large memory (e.g., 32GB flash memory). Hence, this is unproblematic for our application scenarios. Still, the memory footprint can become problematic, because communication is in general costly. If large sketches have to be send very frequently, communication costs can become significant, or sketch sizes might even exceed network capacities.

In follow up research, we want to tackle the general problem of communication costs when using stationary sensors. However, when monitoring non-linear functions, like the union or intersection of sets, this task is not trivial. The LIFT-approach provides a framework for communication reduction in distributed systems, allowing communication efficient monitoring of non-linear functions. We want to apply the LIFT-approach to our monitoring system and test the benefits of employing the LIFT-approach in a real-world experiment.

8 Acknowledgements

This research has been supported by the EU FP7/2007-2013 under grant 255951 (LIFT) and by the German Science Foundation under 'GA 1615/2-1'.

References

[1] C.C. Aggarwal and P.S. Yu. A general survey of privacy-preserving data mining models and algorithms. *Privacy-preserving data mining*, pages 11–52, 2008.

[2] K. Ashok and M.E. Ben-Akiva. Dynamic origin-destination matrix estimation and prediction for real-time traffic management systems. In *International Symposium on the Theory of Traffic Flow and Transportation*, 1993.

[3] J. Barceló, L. Montero, L. Marquès, and C. Carmona. Travel time forecasting and dynamic origin-destination estimation for freeways based on bluetooth traffic monitoring. *Transportation Research Record: Journal of the Transportation Research Board*, 2175(1):19–27, 2010.

[4] L. Bergamini, L. Becchetti, and A. Vitaletti. Privacy-preserving environment monitoring in networks of mobile devices. In *NETWORKING 2011 Workshops*, pages 179–191. Springer, 2011.

[5] Giuseppe Bianchi, Lorenzo Bracciale, and Pierpaolo Loreti. better than nothing privacy with bloom filters: To what extent? In *Privacy in Statistical Databases*, pages 348–363. Springer, 2012.

[6] Andrei Broder and Michael Mitzenmacher. Network applications of bloom filters: A survey. *Internet Mathematics*, 1(4):485–509, 2004.

[7] G. Cormode and M. Garofalakis. Sketching probabilistic data streams. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 281–292. ACM, 2007.

[8] G. Cormode, M. Datar, P. Indyk, and S. Muthukrishnan. Comparing data streams using hamming norms (how to zero in). *Knowledge and Data Engineering, IEEE Transactions on*, 15 (3):529–540, 2003.

[9] G. Cormode, S. Muthukrishnan, and K. Yi. Algorithms for distributed functional monitoring. *ACM Transactions on Algorithms (TALG)*, 7(2):21, 2011.

[10] A.C. Davies, J.H. Yin, and S.A. Velastin. Crowd monitoring using image processing. *Electronics & Communication Engineering Journal*, 7(1):37–47, 1995.

[11] C. Dwork. Differential privacy. *Automata, languages and programming*, pages 1–12, 2006.

[12] Arik Friedman, Ran Wolff, and Assaf Schuster. Providing k-anonymity in data mining. *The VLDB Journal*, 17(4):789–804, 2008.

[13] S. Ganguly, M. Garofalakis, and R. Rastogi. Tracking set-expression cardinalities over continuous update streams. *The VLDB Journal*, 13(4):354–369, 2004.

[14] P.B. Gibbons. Distinct-values estimation over data streams. In *In Data Stream Management: Processing High-Speed Data*. Springer, 2009.

[15] N. Gonçalves, R. José, and C. Baquero. Privacy preserving gate counting with collaborative bluetooth scanners. In *On the Move to Meaningful Internet Systems: OTM 2011 Workshops*, pages 534–543. Springer, 2011.

[16] J. Heikkila and O. Silvén. A real-time system for monitoring of cyclists and pedestrians. In *Second IEEE Workshop on Visual Surveillance (VS'99)*, pages 74–81. IEEE, 1999.

[17] M. Langheinrich. Privacy by design - principles of privacy-aware ubiquitous systems. In *Ubicomp 2001: Ubiquitous Computing*, pages 273–291. Springer, 2001.

[18] T. Liebig, Z. Xu, M. May, and S. Wrobel. Pedestrian quantity estimation with trajectory patterns. In *Proceedings of the ECML/PKDD*, 2012.

[19] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. Why go logarithmic if we can go linear?: Towards effective distinct counting of search traffic. In *Proceedings of the 11th international conference on Extending database technology: Advances in database technology EDBT*, pages 618–629. ACM, 2008.

[20] D. Mir, S. Muthukrishnan, A. Nikolov, and R.N. Wright. Pan-private algorithms via statistics on sketches. In *Proceedings of the 30th symposium on Principles of database systems of data*, pages 37–48. ACM, 2011.

[21] A. Monreale. *Privacy by Design in Data Mining*. PhD thesis, University Pisa, 2011.

[22] Bart Preneel. *Analysis and design of cryptographic hash functions*. PhD thesis, Katholieke Universiteit te Leuven, 1993.

[23] F. Rusu and A. Dobra. Statistical analysis of sketch estimators. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 187–198. ACM, 2007.

[24] S.G. Self and K.Y. Liang. Asymptotic properties of maximum likelihood estimators and likelihood ratio tests under nonstandard conditions. *Journal of the American Statistical Association*, 82(398):605–610, 1987.

[25] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10.

[26] K.Y. Whang, B.T. Vander-Zanden, and H.M. Taylor. A linear-time probabilistic counting algorithm for database applications. *ACM Transactions on Database Systems (TODS)*, 15 (2):208–229, 1990.

Large-scale Online Mobility Monitoring with Exponential Histograms

Christine Kopp
Fraunhofer IAIS
St. Augustin, Germany
christine.kopp@iais.fraunhofer.de

Michael Mock
Fraunhofer IAIS
St. Augustin, Germany
michael.mock@iais.fraunhofer.de

Odysseas Papapetrou
Technical University of Crete
Chania, Greece
papapetrou@softnet.tuc.gr

Michael May
Fraunhofer IAIS
St. Augustin, Germany
michael.may@iais.fraunhofer.de

ABSTRACT

The spread of digital signage and its instantaneous adaptability of content challenges out-of-home advertising to conduct performance evaluations in an online fashion. This implies a tremendous increase in the granularity of evaluations as well as a complete new way of data collection, storage and analysis. In this paper we propose a distributed system for the large-scale online monitoring of poster performance indicators based on the evaluation of mobility data collected by smartphones. In order to enable scalability in the order of millions of users and locations, we use a local data processing paradigm and apply exponential histograms for an efficient storage of visit statistics over sliding windows. In addition to an immediate event centralization we also explore a hierarchical architecture based on a merging technique for exponential histograms. We provide an evaluation on the basis of a real-world data set containing more than 300 million GPS points corresponding to the movement activity of nearly 3,000 persons. The experiments show the accuracy and efficiency of our system.

1. INTRODUCTION

Advertising media are under the obligation to provide reliable performance indicators for the pricing of advertising campaigns. For the German out-of-home (OOH) advertising industry, generating yearly net sales of about 760 million Euro [2], this has meant to establish a system of geographically differentiating performance indicators over the past years[1]. However, with the spread of digital signage also a fine-grained *temporal* differentiation will be required in future. While current performance indicators inform about the poster contacts of seven or ten average days (the two standard durations of poster campaigns in Germany), digital out-of-home (DOOH) advertising spots have a duration of only a few seconds. Assuming an evaluation period of 10 seconds, the granularity of the performance indicators (and consequently of the required input data) increases by four orders of magnitude. DOOH therefore has to face the challenge of collecting and analyzing *big data*. In addition, digital content has the advantage that it can be instantly adapted to a changing audience. This adaptation, however, requires *online* performance information, which forms the second challenge of DOOH performance evaluation.

In this paper we propose a distributed system for the large-scale online monitoring of poster performance indicators based on the evaluation of mobility data collected by smartphones. We hereby consider two use cases which the system shall cover. First, we want to be able to perform online queries which obtain performance measures for the recent past in a sliding window style. Second, we want to analyze historic data for various time intervals. The first type of query allows the online monitoring of poster performance and thus the targeted placement of advertisement spots. The second type of query can be used for billing purposes or to analyze previously collected data sets (e.g. to find interesting visit patterns that can then be monitored in the online system). Although our use cases differ with respect to their system requirements (distributed online processing vs. analysis of massive amounts of centralized data), we want to keep the maintenance effort of the system as low as possible. Our goal is therefore to set up a scalable system architecture that allows an efficient re-use of code from the online scenario for historic data analysis.

The key component of our approach to handle massive streams of data is to use exponential histograms for data compression. This data structure has the advantage that it offers sliding window query capabilities with a guaranteed maximum relative error. In addition, exponential histograms can be applied in a distributed setting [12] thus allowing for scalability when the number of users increases.

Our online system relies on an Android implementation

[1] http://www.agma-mmc.de/media-analyse/plakat.html

that we have used in previous work [3] to detect visit patterns on mobile phones. For the analysis of historic data we have set up a Storm environment. In combination with the Kafka messaging system we are able to perform historic data analysis in a distributed streaming fashion. In this way we can apply the same system architecture for online and historic data analysis. We use the Storm/Kafka environment to perform the experiments in this paper.

We analyze the performance of our system using a real-world GPS data set containing trajectories of 2,967 persons containing more than 300 million GPS points over a period of one week. We extract visit events from this data set using 400,988 points of interest (POI) in Germany from OpenStreetMap (OSM). Our experiments show that the usage of exponential histograms results in an average error of less than 1/10 of the maximum acceptable error while reducing the storage space to an amount as small as 9.7% of the baseline storage space.

The remainder of our paper is organized as follows. Section 2 discusses related work. Section 3 shows our system architecture and Section 4 provides the experiments. We conclude our paper in Section 5.

2. RELATED WORK

2.1 Exponential Histograms

Exponential histograms [1] are a deterministic structure, proposed to address the basic counting problem, i.e., for counting the number of true bits in the last N stream arrivals. They belong to a family of methods that break the sliding window range into smaller windows, called buckets or basic windows, to enable efficient maintenance of the statistics. Each bucket contains the aggregate statistics, i.e., the number of arrivals and bucket bounds, for the corresponding sub-range. Buckets that no longer overlap with the sliding window are expired and discarded from the structure. To compute an aggregate over the whole (or a part of the) sliding window, the statistics from all buckets overlapping with the query range are aggregated. For example, for basic counting, aggregation is a summation of the number of true bits in the buckets. A possible estimation error can be introduced due to the oldest bucket inside the query range, which usually has only a partial overlap with the query. Therefore, the maximum possible estimation error is bounded by the size of the last bucket.

To reduce the space requirements, exponential histograms maintain buckets of exponentially increasing sizes. Bucket boundaries are chosen such that the ratio of the size of each bucket b with the sum of the sizes of all buckets more recent than b is upper bounded. In particular, the following invariant is maintained for all buckets j: $C_j/(2(1+\sum_{i=1}^{j-1} C_i)) \leq \varepsilon$ where ε denotes the maximum acceptable relative error and C_j denotes the size of bucket j (number of true bits arrived in the bucket range), with bucket 1 being the most recent bucket. Queries are answered by summing the sizes of all buckets that fully overlap the query range, and half of the size of the oldest bucket, if it partially overlaps the query. The estimation error is solely contained in the oldest bucket, and is therefore bounded by this invariant, resulting in a maximum relative error of ε.

Recently, Papapetrou et al. [12] showed how an arbitrary number of exponential histograms $EH_1, EH_2, ..., EH_n$ (each one corresponding to an individual stream) can be aggregated/merged, in order to produce a single exponential histogram $EH_\oplus$ that corresponds to the order-preserving union of the streams. More precisely, let ε denote the maximum error parameter of the original exponential histograms, and ε' the parameter of the merging algorithm. The algorithm supports the creation of an aggregated exponential histogram with a maximum relative error of $(\varepsilon + \varepsilon' + \varepsilon \cdot \varepsilon')$. In this work we use this merging algorithm to reduce the memory required for storing the exponential histograms of the visit events coming from various input sources.

2.2 Distributed Evaluation of Visit Events

In previous work we have provided a set of visit quantities that can be used to define performance measures in OOH advertisement [8]. In this paper we concentrate on the evaluation of *gross visits* which state the number of total visits to a certain location and which can be used to estimate the total contacts to a poster site. In addition, we have provided a methodology for the privacy-preserving, distributed collection of visit quantities in previous work [7].

The basic idea of the approach is to decentralize the data collection and evaluation process of movement data. Instead of constantly submitting location information of a user to some central server, the evaluation of visits (or visit patterns) is performed *locally* on a mobile device (e.g. smartphone). The device submits only aggregated and anonymized statistics to a central coordinator. In addition, web anonymization techniques such as onion routing [4] can be used to prevent that the coordinator reconstructs visit histories from several messages of a person based on the communication protocol. A similar, however analytically less powerful framework has previously been proposed by Hoh et al. [6] for the distributed, privacy-preserving monitoring of traffic. However, both papers do not consider the practical aspect of scaling the proposed method to thousands and potentially millions of users. In fact, considering movement statistics from our GPS data set, every person traverses more than 200 street segments per day. If we assume further that each person visits 10 different locations (e.g. work location, shops, bus stops) per day and 20 million persons participate in data collection, about 4.2 billion events occur every day. In order to cope with this number of events, sophisticated analysis and storage algorithms as well as a sophisticated system architecture have to be devised. The design and performance analysis of such a system is the scope of our paper.

3. SYSTEM ARCHITECTURE

Our architecture consists of two or alternatively three layers (see Figure 1). The lowest layer holds the user nodes, which collect the users' GPS data and extract visit events. The visit events are forwarded either directly to the central coordinator (flat setting) or to a layer of intermediate nodes (hierarchical setting). In the flat setting, the coordinator aggregates the visit information of each POI in an exponential histogram. I.e., for each POI an exponential histogram is maintained that records the visit events for this POI. As the exponential histogram stores a time aggregate with the event, queries over time windows can be answered. In the hierarchical setting, the exponential histograms reside already at the intermediate nodes. In regular time intervals the intermediate nodes submit the exponential histograms to the coordinator, which merges them and answers user queries.

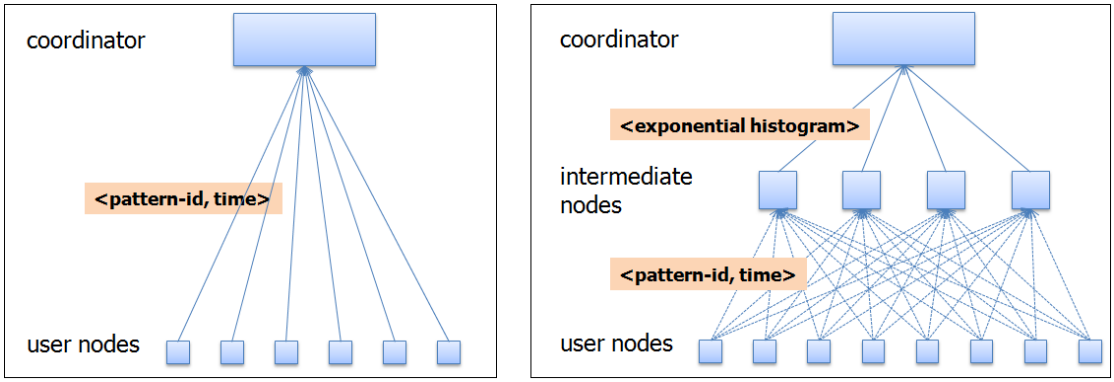


Figure 1: System architecture; left: flat setting; right: hierarchical setting

The exponential histograms cannot be applied at the user level because the number of visit events per user is too small to make the data structure efficient. The layer of intermediate nodes was introduced for horizontal scalability and to avoid an overload of the coordinator. However, it also serves a privacy purpose given that the intermediate nodes do not collude (see [7]). As a user can freely select an intermediate node when submitting a visit event, no intermediate node will obtain the whole event history of a single user. The intermediate nodes submit their data structure in regular time intervals to the coordinator, which finally merges the data structures and answers user queries.

As motivated by our use case, our system shall be able to perform analyses online as well as on historic data. The above architecture describes the online use case. For historic data analysis we have to substitute the layer of local nodes. This substitution should still allow to process data in parallel in order to scale to large amounts of data. In addition, a streaming environment would be preferable in order to re-use existing code. Both aspects can be met by using a distributed streaming processing system as, for example, Storm[2] or S4 [11]. We have ported the Android code of event detection to run as Storm bolts. The input is streamed into the system via the Kafka messaging system [9], which allows to handle each GPS point of the recorded trajectories as individual message. Thus, with this mechanisms we can scale the parallel simulation of event detection horizontally in the cluster. In our experiments described in the next section we used this technique to emulate the event detection on GPS traces of 2,967 test persons in an experimental cluster. Detected events are sent to the intermediate nodes similar to the online setting.

4. EXPERIMENTS

4.1 Data Set

For our experiments we use a subset of a large-scale GPS survey [10] commissioned by the Arbeitsgemeinschaft Media-Analyse e.V.[3], a joint industry committee of German advertising vendors and customers. The GPS data has been collected in the year 2011 and contains 2,967 persons with valid GPS data. The persons are recruited from 31 major cities in Germany and are asked to carry the GPS devices for one week.

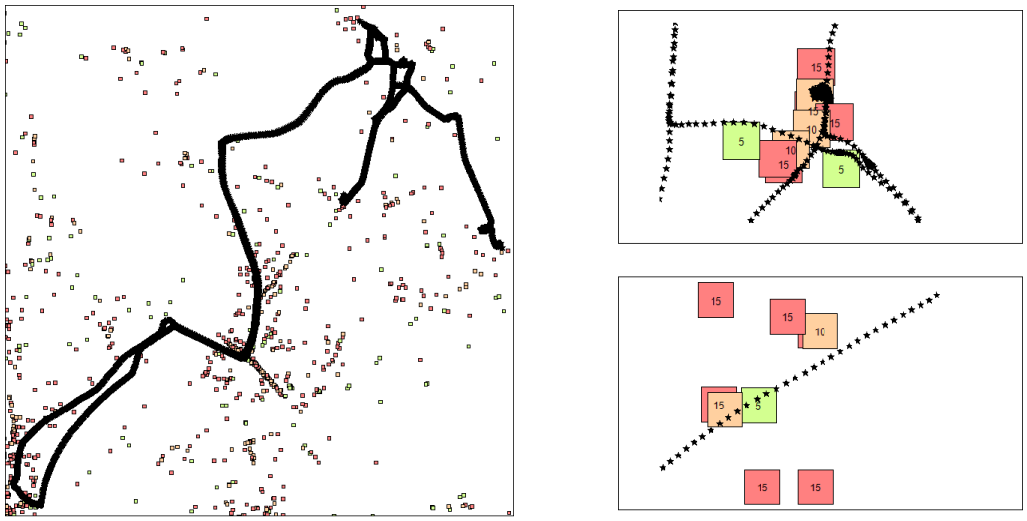
After clean-up the data set contains 304 million GPS points. In addition, we extracted 400,988 points of interest (POI) from OpenStreetMap[4] (OSM) [5] marked with the keys *shop, amenity, leisure, tourism, historic, sport, public transport, railway*. We grouped the POI into the following categories: shop, restaurant, leisure, education, parking and public transport stops. We limited our experiments to those POI because digital posters are still very expensive and therefore placed mostly at attractive places as train stations or shopping locations. For each POI category we defined a minimum stay time and a 50x50 meter spatial buffer in order to extract visit events. Table 1 shows the number of POI aggregated to the six types along with the assumed minimum stay times. Figure 2 left shows a one-day trajectory of one test person along with the extracted POI in its surrounding.

POI type	# POI	min. stay time
shop	89,789	10 min.
restaurant	105,665	15 min.
leisure	69,318	15 min.
education	24,151	15 min.
parking	63,602	5 min.
public transport stop	48,463	5 min.
total	**400,988**	–

Table 1: Number of POI extracted from OSM and minimum defined stay time per category

The extraction of visit events is performed by the *local nodes* (see Section 3). A visit results from the spatial intersection of a trajectory and a geographic location and has to last a given minimum period of time. For a formal definition of a visit see [8]. Figure 2 right shows exemplary the extraction of visit events. The POI are colored according to their minimum required stay time (green = 5 minutes, orange = 10 minutes, red = 15 minutes). In the top right picture one visit occurs in the orange colored POI (where a dense cluster of GPS points exists). In the bottom right picture the user passes the POI merely on his way. As the duration of spatial intersection lies below the minimum stay time, no visit events are generated. For the extraction of visits we apply an algorithm from previous work [3], which

[2]http://storm-project.net
[3]http://www.agma-mmc.de
[4]http://www.openstreetmap.org



(a) one-day trajectory of a test person

(b) trajectory excerpts showing one POI visit on top (dense cluster of points) and two POI passages on bottom

Figure 2: left: one-day trajectory of a test person along with OSM points of interest colored according to minimum stay time (green/orange/red = 5/10/15 minutes); top right: visit in POI with 10 minute stay time; bottom right: passages of POI without visiting

# visits per POI	# POI
1	7,590
2	2,176
3	824
4	458
5	223
6	136
7	101
8	53
9	56
≥ 10	192

Table 2: Frequency of visits per POI

was designed to extract visit patterns from a stream of GPS positions online on mobile phones.

In total we extracted 23,508 visit events to 11,809 different POI for all test persons. This number has been considerably below our expectations. Most likely it results from two reasons. First, the number of OSM POI are incomplete. From the online source *http://www.haltestellen-suche.de* we know to expect at least 217,000 stations of public transport in Germany, and also the number of shops in Germany is considerably above the extracted number of POI. Second, GPS signals are typically blocked inside of buildings. As we applied a light-weight event extraction algorithm (that can run on a mobile device), we may have lost a number of visit events.

Table 2 shows an overview of the number of visit events per POI. Most often, only a single visit occurred. This number is quite reasonable given our low number of visits and the independent movement behavior of the test persons.

In order to perform experiments also on a large-scale data set resembling more closely the real-world situation, we replicated the original visit data by a factor of 1,000. We set the time of each such visit by adding Gaussian noise to the current time with $\mu = 0$ and $\sigma = 10{,}000$ seconds.

4.2 Experimental Set-Up

In our experiments we conducted point queries in a sliding window fashion. I.e., we queried the number of events per POI in the past Δt seconds. The selected query windows were of length 30, 600, 1800, 3600 or 86400 seconds. We performed those queries every 10 minutes (in the hierarchical setting this coincides with the time interval of the force action). For our observation period of one week this resulted in $n_t = 1{,}008$ queries per query window for each of the $n_p = 11{,}809$ visited POI. In accordance with our maximum query interval, we set the sliding window parameter of the exponential histogram to 86,400 seconds in all experiments. Further, we varied the maximum acceptable relative error ε to take the values 0.01, 0.02, 0.04, 0.08 and 0.16. In the hierarchical setting we used 10 intermediate nodes which submitted their data structures every 600 seconds to the coordinator.

We measured the error for each experiment using the mean absolute percentage error (MAPE), which is defined as follows:

$$MAPE = \frac{\sum_{i=1}^{n_p} \sum_{j}^{n_t} \left| \frac{x_{ij} - \widehat{x}_{ij}}{x_{ij}} \right|}{n_p \cdot n_t}$$

where x_{ij} denotes the true number of visit events at POI i in query window j and $\widehat{x}_{ij}$ denotes the number of events returned from the exponential histogram. In the case of $x_{ij} = 0$ we added a relative error of zero if our estimate was correct ($\widehat{x}_{ij} = 0$) and a relative error of ∞ if $\widehat{x}_{ij} \neq 0$. This latter case, however, did not occur. We performed all experiments for the flat and hierarchical setting as well as for the original and multiplied data set.

4.3 Results

Figure 3 shows the results for the flat and hierarchical setting of the multiplied data set. The respective numbers are provided in Tables 3 and 4. Note that we display only the results for the multiplied data set because due to the few visit events in the original data set the error was nearly always zero there.

In general, the MAPE is very low, lying with one exception below 1%. For both the flat and hierarchical setting two trends can be observed. First, the MAPE decreases with smaller ε. Second, the MAPE decreases with decreasing size of the query window. The first effect is nearly linear for all query windows and can be expected from the characteristics of exponential histograms. The second is also expected because the error guarantees are given on the size of the sliding window, which was fixed to 86,400 seconds. Accordingly, the error for smaller time intervals has to be lower. However, the effect is linear to the logarithm of the query window sizes, i.e. when increasing the query window, the MAPE increases sublinearly.

When comparing the error between the flat and hierarchical setting, the merge operations result in only a small increase in error.

query wind.	**ε=0.01**	**ε=0.02**	**ε=0.04**	**ε=0.08**	**ε=0.16**
30 s	7E-6%	2E-5%	5E-5%	2E-4%	3E-3%
600 s	2E-4%	3E-3%	0.03%	0.18%	0.44%
1800 s	3E-3%	0.04%	0.12%	0.28%	0.54%
3600 s	0.02%	0.06%	0.15%	0.32%	0.59%
86400 s	0.06%	0.14%	0.28%	0.44%	0.79%
mem.	14.5 MB	8.8 MB	5.5 MB	3.4 MB	2.5 MB

Table 3: Mean absolute percentage error and memory usage for flat setting

query wind.	**ε=0.01**	**ε=0.02**	**ε=0.04**	**ε=0.08**	**ε=0.16**
30 s	8E-6%	2E-5%	6E-5%	2E-4%	3E-3%
600 s	2E-3%	3E-3%	0.03%	0.18%	0.45%
1800 s	3E-3%	0.04%	0.12%	0.28%	0.64%
3600 s	0.02%	0.06%	0.15%	0.36%	0.75%
86400 s	0.07%	0.16%	0.32%	0.53%	1.03%
mem.	14.5 MB	8.8 MB	5.5 MB	3.4 MB	2.5 MB

Table 4: Mean absolute percentage error and memory usage for hierarchical setting

In order to set the MAPE in perspective to the number of visit events, Table 5 shows the average and maximum number of visits per POI and query interval. The average is hereby calculated once for all POI and time slots and once only for those containing at least one event.

The memory usage of the exponential histogram at the end of the observation period is depicted in the last line in Tables 3 and 4. Assuming fixed 32-bit counters, it depends only on ε and the maximum possible count N in the sliding window of each POI, requiring $O(\frac{1}{\varepsilon}\log N)$ space [1]. As we maintain an exponential histogram for each POI, the required memory depends also linearly on the number of (distinct) visited POI which is, however, constant in our experiments.

query wind.	**avg. events**	**avg. events > 0**	**max. events**
30 s	0.1	1.6	1,916
600 s	2.0	12.1	2,029
1,800 s	5.9	32.0	2,260
3,600 s	11.8	60.0	3,118
86,400 s	270.3	709.8	36,037

Table 5: Number of average and maximum events per POI and query window in ground truth

4.4 Discussion

Our experiments show that the resultant error is very low. For all settings of ε the mean error (MAPE) is less than 1/10 of the maximum acceptable error. This is a very good result. Especially we can be sure for small total number of visits that the query results are always correct. For example, setting $\varepsilon = 0.01$ will result in no errors if less than 100 events occur per POI. This is an important characteristic because the visit frequency of POI is right-tailed, containing only few POI with very high frequencies.

Further the experiments show that our setting scales horizontally. By introducing a layer of 10 intermediate nodes, the MAPE was on average 7.5% higher and at most 23% higher than in the flat setting. Both numbers are considerably below the maximum acceptable error as well as the maximum relative error guaranteed for the join of exponential histograms.

Finally, to evaluate the memory usage, we can compare the numbers to the following baseline scenario. Whenever a visit event occurs, the POI identifier and timestamp are stored at the coordinator using two 4 Byte integers. As our sliding window covers only one day, we will assume that we have to store 1/7 of the total visit events. For the original 23,509 events this results in 0.026 MB. For the multiplied data set it results in 25.6 MB. The storage amount for the original events using exponential histograms varied between 0.54-4.7 MB. In this case we did not save on memory. However, using the more realistic multiplied data set with exponential histogram sizes between 2.5-14.5 MB, our experiments require only 9.7-56.6% of the baseline storage space depending on the selected ε.

When extrapolating to the envisioned setting of monitoring 20 million persons generating each 210 events per day on about 6,500,000 distinct POIs in Germany (including the 6,000,000 distinct street segments), just storing the raw event data would result in 31.3 GB memory consumption. This is considerably above the 1.3-7.8 GB required by the exponential histograms (by just taking into account that our memory consumption increases linearly with the number of distinct POIs).

Considering our entire approach including exponential histograms and local evaluation, the storage reduction is even much higher compared to a naive centralized setting where the users submit a GPS position every second to some central coordinator.

Also note that inserting into and querying an exponential histogram almost takes constant time far below a microsecond, which is much faster than searching an event database of raw events.

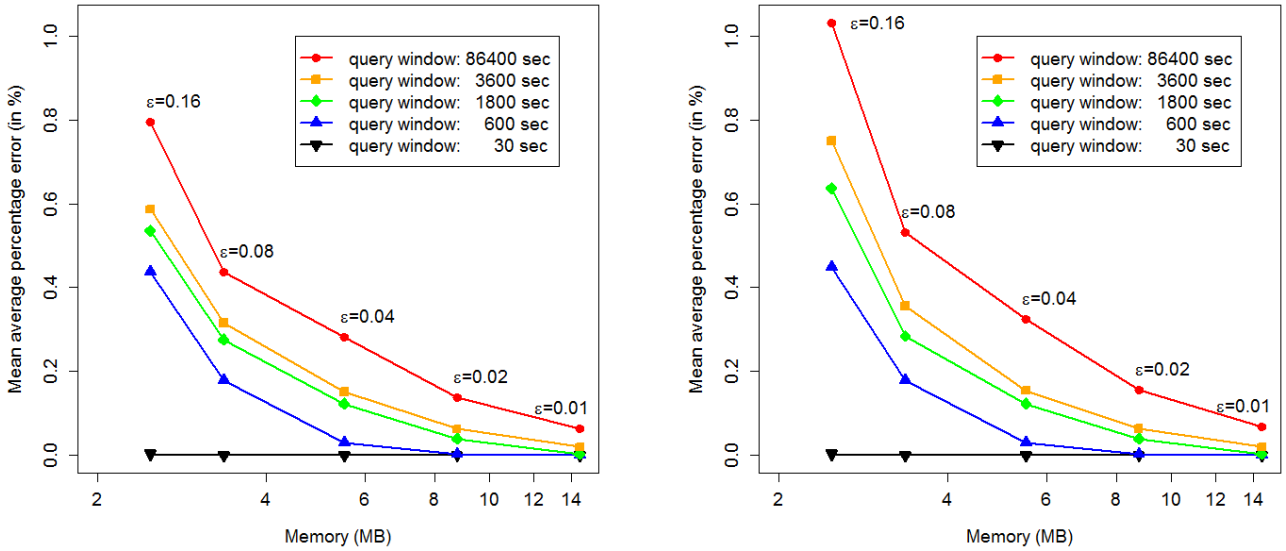


Figure 3: Mean average percentage error and memory usage for different maximum relative errors (ε) and query window sizes; left: without intermediate nodes; right: hierarchy with 10 intermediate nodes

5. CONCLUSIONS

In this paper we propose a distributed system for the large-scale online monitoring of poster performance indicators based on the evaluation of mobility data. Our system relies on the collection and local processing of mobility data via smartphones and uses exponential histograms for the efficient storage and querying of visit statistics in a sliding window fashion. Our experiments on a multiplied real-world data set with nearly 3,000 persons show that the usage of exponential histograms results in an average error of less than 1/10 of the maximum acceptable error while reducing the storage space to an amount as small as 9.7% of the baseline storage space.

6. ACKNOWLEDGMENTS

We thank our colleague Sebastian Bothe for supporting us to run the cluster-based version of the experiments and the Arbeitsgemeinschaft Media-Analyse e.V. for granting the use of the GPS data set. The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 255951 (LIFT).

7. REFERENCES

[1] M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows. *SIAM J. Comput.*, 31(6):1794–1813, 2002.

[2] Fachverband Außenwerbung e.V. Netto-Werbeeinnahmen erfassbarer Werbeträger in Deutschland, 2002-2010 (Net turnover of confirmable advertising media in Gemany, 2000-2010), 2011. `http://www.faw-ev.de/media/download/marktdaten/4\_Nettoumsaetze\_aller_Werbemedien\_ab\_2002.pdf`.

[3] S. Florescu, C. Körner, M. Mock, and M. May. Efficient mobility pattern stream matching on mobile devices. In *Proc. of the Ubiquitous Data Mining Workshop (UDM 2012)*, pages 23–27, 2012.

[4] D. Goldschlag, M. Reed, and P. Syverson. Onion routing for anonymous and private internet connections. *Comm. of the ACM*, 42:39–41, 1999.

[5] M. M. Haklay and P. Weber. OpenStreetMap: User-Generated Street Maps. *IEEE Pervasive Computing*, 7(4):12–18, 2008.

[6] B. Hoh, M. Gruteser, R. Herring, J. Ban, D. Work, J.-C. Herrera, A. M. Bayen, M. Annavaram, and Q. Jacobson. Virtual trip lines for distributed privacy-preserving traffic monitoring. In *Proc. of the 6th Int. Conf. on Mobile Systems, Applications, and Services (MobiSys'08)*, pages 15–28. ACM, 2008.

[7] C. Kopp, M. Mock, and M. May. Privacy-preserving distributed monitoring of visit quantities. In *SIGSPATIAL 2012 Int. Conf. on Advances in Geographic Information Systems(SIGSPATIAL/GIS)*, pages 438–441, 2012.

[8] C. Körner. *Modeling Visit Potential of Geographic Locations Based on Mobility Data*. PhD thesis, University of Bonn, 2012.

[9] J. Kreps, N. Narkhede, and J. Rao. Kafka: A distributed messaging system for log processing. In *Proceedings of 6th International Workshop on Networking Meets Databases (NetDB), Greece*, 2011.

[10] Media-Micro-Census GmbH. ma 2012 Plakat - Methoden-Steckbrief zur Berichterstattung, 2012. `http://www.agma-mmc.de/publikationen/methodische-berichte/methoden-steckbriefe.html?eID=dam\_frontend\_push\&docID=179Z`.

[11] L. Neumeyer, B. Robbins, A. Nair, and A. Kesari. S4: Distributed stream computing platform. In *Proceedings of the 2010 IEEE Int. Conf. on Data Mining Workshops*, ICDMW '10, pages 170–177, Washington, DC, USA, 2010. IEEE Computer Society.

[12] O. Papapetrou, M. N. Garofalakis, and A. Deligiannakis. Sketch-based querying of distributed sliding-window data streams. *PVLDB*, 5(10):992–1003, 2012.

Privacy-Aware Distributed Mobility Data Analytics

Francesca Pratesi[1], Anna Monreale[2], Hui Wang[3], Salvatore Rinzivillo[1], Dino Pedreschi[2], Gennady Andrienko[4], and Natalia Andrienko[4]

[1] ISTI-CNR, Italy, {francesca.pratesi,salvatore.rinzivillo}@isti.cnr.it
[2] Univ. of Pisa, Italy, {annam,pedre}@di.unipi.it
[3] Stevens Inst. of Technology, USA, Hui.Wang@stevens.edu
[4] Fraunhofer IAIS, Germany, {gennady,natalia}.andrienko@iais.fraunhofer.de

Abstract. We propose an approach to preserve privacy in an analytical processing within a distributed setting, and tackle the problem of obtaining aggregated information about vehicle traffic in a city from movement data collected by individual vehicles and shipped to a central server. Movement data are sensitive because they may describe typical movement behaviors and therefore be used for re-identification of individuals in a database. We provide a privacy-preserving framework for movement data aggregation based on trajectory generalization in a distributed environment. The proposed solution, based on the differential privacy model and on sketching techniques for efficient data compression, provides a formal data protection safeguard. Using real-life data, we demonstrate the effectiveness of our approach also in terms of data utility preserved by the data transformation.

1 Introduction

The availability of low cost GPS devices enables collecting data about movements of people and objects at a large scale. Understanding of the human mobility behavior in a city is important for improving the use of city space and accessibility of various places and utilities and managing the traffic network. Generalization and aggregation of individual movement data can provide an overall description of traffic flows. Paper [3] proposes a method for generalization and aggregation of movement data that requires having all individual data in a central station. This centralized setting entails two important problems: a) the amount of information to be collected and processed may exceed the computational resources, and b) the raw data describe the mobility behavior of the individuals with great detail enabling the inference of very sensitive information related to the personal private sphere.

We address these problems and propose a privacy-preserving distributed computation framework for the aggregation of movement data. We assume that on-board location devices in vehicles continuously trace the positions of the vehicles and can periodically send derived information about their movements to a central station, which stores it. This information can be used to compute a summary of the traffic conditions on the whole territory. To protect individual privacy, we propose a data transformation method based on the well-known differential privacy model. We also propose and analyze some methods to reduce the amount of information that each vehicle transmits to the central station that allow us to obtain a compressed data representation. The central station, called *coordinator*, is able to reconstruct the compressed and private movement data that, although transformed to guarantee privacy, preserve some important properties of the original data making them useful for mobility analysis.

The remainder of the paper is organized as follows. Section 2 introduces background information and definitions. Section 3 describes the system architecture and states the problem. Section 4 presents our privacy-preserving solutions. In Section 5, we discuss how to compact the communications. In Section 6, we briefly illustrate the behavior of the coordinator. Experimental results from applying our method to real-world data are presented and discussed in Section 7. Section 8 discusses the related work and Section 9 concludes the paper.

2 Preliminaries

2.1 Movement Data Representation

Given a 2-D space territory $\mathbf{R}^2$, a *trajectory* is a sequence of pairs $T = \{\langle l_1, t_1\rangle, \ldots, \langle l_n, t_n\rangle\}$, where t_i (with $i = 1 \ldots n$) denotes a timestamp such that $\forall_{1 \leq i < n}\ t_i < t_{i+1}$ and $l_i = (x_i, y_i)$ are points in $\mathbf{R}^2$. Intuitively, each pair $\langle l_i, t_i\rangle$ indicates that the object is in the position $l_i = \langle x_i, y_i\rangle$ at time t_i. In a time interval τ, each moving object can have multiple trajectories. We do not require that each trajectory is *complete*, i.e., locations may be missing at some timestamps.
We assume that the territory $\mathbf{R}^2$ is subdivided into cells $C = \{c_1, c_2, \ldots, c_p\}$ which compose a partition of the territory. During travel a user may move from one cell to another. We use g to denote the function that applies the spatial generalization to a trajectory. Given a trajectory T this function generates the generalized trajectory $g(T)$, i.e. a sequence of *moves* with temporal annotations, where a *move* is an pair (l_{c_i}, l_{c_j}) indicating that the moving object moves from the cell c_i to the *adjacent* cell c_j. Note that l_{c_i} denotes the pair of spatial coordinates representing the centroid of the cell c_i; in other words $l_{c_i} = \langle x_{c_i}, y_{c_i}\rangle$. The *temporal annotated move* is the quadruple $(l_{c_i}, l_{c_j}, t_{c_i}, t_{c_j})$ where l_{c_i} is the location of the origin, l_{c_j} is the location of the destination and the t_{c_i}, t_{c_j} are the time information associate to l_{c_i} and l_{c_j}. As a consequence, we define a generalized trajectory as follows.

Definition 1 (Generalized Trajectory). Given a trajectory $T = \langle l_1, t_1\rangle, \ldots, \langle l_n, t_n\rangle$. Let $C = \{c_1, c_2, \ldots, c_p\}$ be the territory partition. A generalized version of T is a sequence of temporal annotated moves $T_g = \{(l_{c_1}, l_{c_2}, t_{c_1}, t_{c_2})\ (l_{c_2}, l_{c_3}, t_{c_2}, t_{c_3})\ \ldots (l_{c_{m-1}}, l_{c_m}, t_{c_{m-1}}, t_{c_m})\}$ with $m <= n$.

Now, we show how to construct *frequency distribution vectors* of generalized trajectories.
First, we define the function *Move Frequency* (*MF*) to compute how many times the move appears in a generalized trajectory T_g within a given time interval. More formally:

Definition 2 (Move Frequency). Let T_g be a generalized trajectory and let (l_{c_i}, l_{c_j}) be a move. Let τ be a temporal interval. The *move frequency* function is defined as:

$$MF(T_g, (l_{c_i}, l_{c_j}), \tau) = |\{(l_{c_i}, l_{c_j}, t_i, t_j) \in T_g | t_i \in \tau \wedge t_j \in \tau\}|.$$

For any move (l_{c_i}, l_{c_j}), the value of $MF(T_g, (l_{c_i}, l_{c_j}), \tau)$ can be any non-negative integer. This function can be easily extended for taking into consideration a set of generalized trajectories $\mathscr{T}^{\mathscr{G}}$. In this case, the information computed by the function represents the total number of movements from the cell c_i to the cell c_j in a time interval in the set of trajectories.

Definition 3 (Global Move Frequency). Let $\mathscr{T}^{\mathscr{G}}$ be a set of generalized trajectories and let (l_{c_i}, l_{c_j}) be a move. Let τ be a time interval. The *global move frequency* function is defined as: $GMF(\mathscr{T}^{\mathscr{G}}, (l_{c_i}, l_{c_j}), \tau) = \sum_{\forall T_g \in \mathscr{T}^{\mathscr{G}}} MF(T_g, (l_{c_i}, l_{c_j}), \tau)$.

The number of movements between two cells computed by either the function *MF* or *GMF* describes the amount of traffic flow between the two cells in a specific time interval. This information can be represented by a frequency vector. To define the frequency vector, we first define *vector of moves*.

Definition 4 (Vector of Moves). Let $C = \{c_1, c_2, \ldots, c_p\}$ be the set of the cells composing the territory partition. The *vector of moves* M is a vector of size $s = |\{(c_i, c_j) | c_i \textit{ is adjacent to } c_j\}|$, in which each element $M[k] = (l_{c_i}, l_{c_j})$, where $1 \leq k \leq s$, is the *move* from the cell c_i to the adjacent cell c_j.

Definition 5 (Frequency Vector). Let $C = \{c_1, c_2, \ldots, c_p\}$ be the cells that compose the territory partition and let M be its vector of moves. Given a set of generalized trajectories $\mathscr{T}^{\mathscr{G}}$ in a time interval τ, its *frequency vector* f is a vector of size $s = |\{(c_i, c_j) | c_i \textit{ is adjacent to } c_j\}|$, in which each element $f[k] = GMF(\mathscr{T}^{\mathscr{G}}, M[k], \tau)$.

2.2 Differential Privacy

Differential privacy implies that adding or deleting a single record does not significantly affect the result of any analysis. Intuitively, differential privacy can be understood as follows. Let a database D include a private data record d_i about an individual i. By querying the database, it is possible to obtain certain information I(D) about all data and information I(D-d_i) about the data without the record d_i. The difference between I(D) and I(D-d_i) may enable inferring some private information about the individual i. Hence, it must be guaranteed that I(D) and I(D-d_i) do not significantly differ for any individual i. The formal definition [10] is the following. Here the parameter, ε, specifies the level of privacy guaranteed.

Definition 6 (ε-differential privacy). *A privacy mechanism A gives ε-differential privacy if for any dataset D_1 and D_2 differing on at most one record, and for any possible output D' of A we have $Pr[A(D_1) = D'] \leq e^{\varepsilon} \times Pr[A(D_2) = D']$ where the probability is taken over the randomness of A.*

A fundamental concept of this model is the global sensitivity of a function mapping underlying datasets to (vectors of) reals.

Definition 7 (Global Sensitivity). *For any function $f : D \to R^d$, the sensitivity of f is $\Delta f = max_{D_1,D_2} ||f(D_1) - f(D_2)||_1$ for all D_1, D_2 differing in at most one record.*

For the analysis whose outputs are real, a standard mechanism to achieve differential privacy is to add Laplace noise to the true output of a function. Dwork et al. [10] propose the Laplace mechanism which takes as inputs a dataset D, a function f, and the privacy parameter ε. The magnitude of the noise added conforms to a Laplace distribution with the probability density function $p(x|\lambda) = \frac{1}{2\lambda} e^{-|x|/\lambda}$, where λ where $\lambda = \Delta f/\varepsilon$.

Theorem 1. *[10] For any function $f : D \to R^d$ over an arbitrary domain D, the mechanism A $A(D) = f(D) + Laplace(\Delta f/\varepsilon)$ gives ε-differential privacy.*

A relaxed version of differential privacy, named (ε, δ)-differential privacy [10], allows a little privacy loss due to a variation in the output distribution for the privacy mechanism A.

Definition 8 ((ε, δ)-differential privacy). *A privacy mechanism A gives (ε, δ)-differential privacy if for any dataset D_1 and D_2 differing on at most one record, and for any possible output D' of A we have $Pr[A(D_1) = D'] \leq e^{\varepsilon} \times Pr[A(D_2) = D'] + \delta$ where the probability is taken over the randomness of A.*

Note that when $\delta = 0$, (ε, δ)-differential privacy is equivalent to ε-differential privacy.

3 Problem Definition

System Architecture. We consider a system architecture as that one in described in [6]. In particular, we assume a distributed-computing environment comprising a collection of k (trusted) remote sites (nodes) and a designated coordinator site. Each remote site exchanges messages only with the coordinator, providing it with state information on its (locally observed) streams. There is no communication between remote sites.
In our scenario, the coordinator is responsible for computing the aggregation of movement data on a territory by combining the information received by each node. In order to obtain the aggregation of the movement data in the centralized setting, we need to generalize all the trajectories by using the cells of a partition of the territory. We assume that the partition of the territory, is known by both all the nodes and the coordinator.

Formally, each remote node $j \in \{1,\ldots,k\}$ (that represents a vehicle that moves in this territory) observes local update streams that incrementally render a distinct frequency distribution vector f^j over data elements; that is, $f^j[v]$ denotes the frequency of element v observed locally at remote node j. The coordinator for each computes the *global frequency distribution vector* $F = \sum_{j=1}^{k} f^j$.

Privacy Model. In our setting, we assume that each node in our system is honest; in other words we do not consider attacks at the node level. Instead, we take into consideration possible attacks from any intruder between the node and the coordinator (i.e., attacks during the communications), and any intruder at coordinator site. We consider sensitive information as any information from which the typical mobility behavior of a user may be inferred. This information is considered sensitive for two main reasons: 1) typical movements can be used to identify the drivers who drive specific vehicles even when a simple de-identification of the individual in the system is applied; and 2) the places visited by a driver could identify specific sensitive areas such as clinics, hospitals, the user's home.

Unfortunately, releasing frequency of moves instead of raw trajectory data to the coordinator is not privacy-preserving, as the intruder may still infer the sensitive typical movement information of the driver. As an example, the attacker could learn the driver's most frequent move. Therefore, we need to find effective privacy mechanisms on the real count associate to each move, in order to generate uncertainty. As a consequence, the goal of our framework is to compute a distributed aggregation of movement data for a comprehensive exploration of them while preserving privacy.

Definition 9 (Problem Definition). Given a set of cells $C = \{c_1,\ldots,c_p\}$ and a set $V = \{V_1,\ldots V_k\}$ of vehicles, the *privacy-preserving distributed movement data aggregation problem* (DMAP) consists in computing, in a specific time interval τ the $f^{\tau}_{DMAP}(V) = [f_1, f_2, \ldots, f_s]$, where each $f_i = GMF(\mathscr{T}^{\mathscr{G}}, M[i], \tau)$ and $s = |\{(c_i, c_j) | c_i \textit{ is adjacent to } c_j\}|$, while preserving privacy. Here, $\mathscr{T}^{\mathscr{G}}$ is the set of generalized trajectories related to the k vehicles V in the time interval τ and M is the vector of moves defined on the set of cells C.

In this paper, we propose two different privacy-preserving solutions, based on the differential privacy model, to address the above problem. This privacy model is a strong and independent on the background knowledge of an adversary.

4 Privacy-preserving Node Computation

We assume that each node represents a vehicle that moves in a specific territory. Each vehicle in a given time interval observes sequences of spatio-temporal points (trajectories) and computes the corresponding frequency vector that is to be sent to the coordinator. Before sending the frequency vector each node applies a transformation for guaranteeing privacy and then transformation to reduce the amount of information if necessary (Section 5).

4.1 Frequency Vector Construction

Trajectory Generalization. Given a specific division of the territory, a trajectory is generalized in the following way. We apply place-based division of the trajectory into segments. The area c_1 containing its first point l_1 is found. Then, the second and following points of the trajectory are checked for being inside c_1 until finding a point l_i not contained in c_1. For this point l_i, the containing area c_2 is found. The trajectory segment from the first point to the i-th point is represented by the vector (c_1, c_2). Then, the procedure is repeated: the points starting from l_{i+1} are checked for containment in c_2 until finding a point l_k outside c_2, the area c_3 containing l_k is found, and so forth up to the last point of the trajectory.

There may be also a case when all points of a trajectory are contained in one and the same area c_1. Then, the whole trajectory is represented by the sequence $\{c_1\}$. Since, globally we want to compute aggregation of moves we discard this kind of trajectories.

Move Vector Computation. After the generalization of a trajectory, the node computes the *Move Frequency* function for each move (l_{c_i}, l_{c_j}) in that trajectory and updates its frequency vector f^{V_j} associated to the current time interval τ. Intuitively, the vehicle populates the frequency vector f^{V_j} according the generalized trajectory observed. So, at the end of the time interval τ the element $f^{V_j}[i]$ contains the number of times that the vehicle V_j moved from m to n in that time interval, if $M[i] = (m,n)$.

4.2 How to Achieve Privacy

As discussed above, if a node sends the original frequency vector without any data transformation to the coordinator, the intruder may still be able to infer the sensitive typical movements of the vehicle represented by the node. An attacker could also infer if during a trip a user went from a location a to a location b and how many times. The questions are, *how can we hide the event that the user moved from a location "a" to a location "b" during the time interval τ? And how can we hide the real count of a move in that time window?* To answer these questions, we propose two solutions based on a rigorous privacy model named ε-differential privacy (Section 2.2). Each solution provides a different balance between privacy and data utility. The key point of this model is the definition of the sensitivity. Given a move (a,b) its sensitivity is straightforward: releasing its frequency have sensitivity 1 as adding or removing a single flow can affect its frequency by at most 1. Given the sensitivity of the count of a move we can define a differential private mechanism in various ways. In the following, we present the two solutions.

***UniversalNoise* Approach.** Our first approach, named *UniversalNoise*, is based on the classic ε-differential privacy model. In particular, at the end of the time interval τ, before sending the frequency vector to the coordinator, each node adds the Laplace noise $Lap(\frac{\Delta f}{\varepsilon})$, where Δf is fixed to 1, to each element in the frequency vector the value in that position of the vector. At the end of this step the node transforms f^{V_j} into $\tilde{f}^{V_j}$.

Privacy Analysis. We can show that the privacy transformation presented just now satisfies ε-differential privacy.

Theorem 2. *Given the total privacy budget ε, for each frequency value x, UniversalNoise approach ensures ε-differential privacy.*

The correctness of Theorem 2 is straightforward due to how the noise is added according to the Laplace mechanism [10].

***BoundedNoise* Approach.** Differential privacy in some specific contexts could lead to the destruction of the data utility because of the added noise that, although with small probability, can reach values of arbitrary magnitude. Moreover, adding noise drawn from the Laplace distribution could generate negative values for the flow in a move and negative flows does not make sense. To prevent this two problems, that characterize the *UniversalNoise* approach, we propose an approach called *BoundedNoise* that draws the noise from a cutting version of the Laplace distribution. In particular, for each value x of the vector f^{V_j} we draw the noise from $Lap(\frac{1}{\varepsilon})$ bounding the noise value to the interval $[-x,x]$. In other words, if we have the original flow $f^{V_j}[i] = x$ in the perturbed version we obtain a flow value in the interval $[0,2x]$. The use of a truncated version of the Laplace distribution can lead to privacy leaks and in the following we show that our privacy mechanism satisfies (ε,δ)-differential privacy, where δ represents this privacy loss.

Privacy Analysis. As pointed out in [12] differential privacy must be applied with caution. The privacy protection provided by differential privacy relates to the data generating mechanism and deterministic aggregate level background knowledge. We observe that bounding

the Laplace noise will lead to some privacy leakage on some values. For instance, from the noisy frequency values that are large, the attacker can infer that these values should not be transformed from small ones. To analyze the privacy leakage of our bound-noise approach, we first explain the concept of *statistical distance*. Statistical distance is defined in [4]. Formally, given two distributions X and Y, the *statistical distance* between X and Y over a set U is defined as $[d(X,Y) = max_{S \in U}(Pr[X \in S] - Pr[Y \in S])$.

Lemma 1. *[4] Given two probabilistic functions F and G with the same input domain, where F is (ε, δ_1)-differentially private. If for all possible inputs x we have that the statistical distance on the output distributions of F and G is: $[d(F(x), G(x)) \leq \delta_2$, then G is $(\varepsilon, \delta_1 + (e^{\varepsilon} + 1)\delta_2)$-differentially private.*

Let F and F' be the frequency distribution before and after adding Laplace noise. We can show that the statistical distance between F and F' can be bounded as follows:

Lemma 2. *[4] Given an (ε, δ)-differentially private function F with $F(x) = f(x) + R$ for a deterministic function f and a random variable R. Then for all x, the statistical distance between F and its throughput-respecting variant F' with the bound b on R is at most*

$$d(F(x) - F'(x)) \leq Pr[|R| > b].$$

[4] has the following lemma to bound the probability $Pr[|R| > b]$.

Lemma 3. *[4] Given a function F with $F(x) = f(x) + Lap(\frac{\Delta f}{\varepsilon})$ for a deterministic function f, the probability that the Laplacian noise $Lap(\frac{\Delta f}{\varepsilon})$ applied to f is larger than b is bounded by: $Pr(Lap(\frac{\Delta f}{\varepsilon}) > b) \leq \frac{2(\Delta f)^2}{b^2 \varepsilon^2}$.*

We stress that in our approach, the bound b of each frequency value x is not fixed. Indeed, $b = x$. Therefore, each frequency value x has different amounts of privacy leakage. Our approach thus achieves different degree of (ε, δ)-differentially privacy guarantee on each frequency value x. Theorem 3 shows more details.

Theorem 3. *Given the total privacy budget ε, for each frequency value x, BoundedNoise approach ensures $(\varepsilon, (e^{\varepsilon} + 1)\frac{2}{x^2 \varepsilon^2})$-differential privacy.*

The correctness of Theorem 3 derives from Lemma 1 and Lemma 3.

5 How to Achieve Compact Communications

In a distributed system an important issue to be considered is the amount of data to be communicated. In fact, real life systems usually involve thousands vehicles (nodes) that are located in any place of the territory. Each vehicle has to transmit to the coordinator the information contained in its frequency vector that has a size depending on the number of cells that represent the partitions of the territory. The number of cells in a territory can be very huge and this can make large frequency vectors. As an example, in the dataset of real-life trajectories used in our experiments, there are approximately $4,200$ vehicles and we use a territory tessellation of about $2,400$ cells. So, considering as possible moves only pairs of adjacent cells we obtain frequency vectors containing about $15,900$ positions (moves). These considerations make the optimization of communicated information necessary. To address this problem it is possible to compress the transmitted data by sketching algorithms [7]. Here, we propose the application of *AGMS*, *Count-Min* and *Count* sketch algorithms. These algorithms map a frequency vector f onto a more compressed vector. The size of the sketched vectors depends on

two parameters: α indicating the accuracy (i.e. the approximation error), while the parameter γ representing the probability of exceeding the accuracy bounds.
The *AGMS* sketch consists of an array C of r counters and a four-wise independent hash function g_i, which maps the items uniformly onto $\{-1,+1\}$. The sketch is built as follows: $\forall j, 1 \leq j \leq |C|, C[j] = \sum_{i=1}^{M} f[i] * g_j[i]$. AGMS sketch was designed to estimate (self-)join and only few works use this kind of sketches to estimate the single items. Aggarwal and Yu in [2] explain how to estimate any original vector component. Let $E_i^k = C[j] * g_i[j]$ be the sketch derivative. For the estimation of the k-th component we compute $|C|$ values of E^k (one for each component of the sketch) and then we compute the mean of these E^k. Therefore the estimation for $f[k]$ is $\tilde{f}[k] = E[E^k]$. Setting $r = O(\frac{1}{\alpha^2} \log \frac{1}{\gamma})$ ensures that the estimation of $f[k]$ has error at most αn with probability at least $1-\gamma$.
The *Count-Min* sketch consists of an array C of $d \times w$ counters and for each of d rows a pairwise independent hash functions h_j, that maps items onto $[w]$. Each item is mapped onto d entries in the array, by adding to the previous value the new item. Given a sketch representation of a vector we can estimate the original value of each component of the vector by the following function $f[i] = min_{1 \leq j \leq d} C[j, h_j(i)]$; this kind of estimation makes *Count-Min* sketch suitable for compressing non-negative values, while does not work well in case of presence of negative values. The estimation of each component j is affected by an error, but it is showed that the overestimate is less than n/w, where n is the number of components. So, setting $d = \log \frac{1}{\gamma}$ and $w = O(\frac{1}{\alpha})$ ensures that the estimation of $f[i]$ has error at most αn with probability at least $1-\gamma$.
The *Count* sketch consists of an array C of $d \times w$ counters. For each of d rows, there are two hash functions: h_j that maps items, that our case are moves, onto one of the elements of the $j-th$ row, and g_j that maps each item onto $\{-1,1\}$. For each item i, it will be mapped onto d entries in the array by adding the value $f[i] \times g_j(i)$ on the entry $C[j, h_j(i)]$ in row j, for $1 \leq j \leq d$. Given a sketch representation of a vector we can estimate the original value of each component of the vector by the following function $\hat{f}[i] = median_{1 \leq j \leq d} g_j(i) C[j, h_j(i)]$. Setting $d = \log \frac{1}{\gamma}$ and $w = log(\frac{4}{\alpha^2})$ this sketch ensures that the estimation of $f[i]$ has error at most αn with probability at least $1-\gamma$.
Adding this data summarization step before transmitting the vector to the coordinator does not change the privacy guarantee provided by the above methods. This is due to the fact that the sketching functions only access a differentially private frequency vector, not the underlying database. As proven by Hay et al. [11], a post-processing of differentially private results remains differentially private. On the other hand, the application of sketching algorithms has an impact on the data utility because the estimation of each component is an approximation. Sometimes, in the case of very sparse frequency vectors could be more convenient to avoid the sketching algorithm and to send to the coordinator only the components with non-zero values. In this case the node has to transmit for each non-zero component the pair $\langle index, value \rangle$, where *index* denotes the position of the vector to be updated and *value* denotes the corresponding frequency. The problem is to understand when it is better sketching the vector and when sending the list of non-zero component. In our experiments on real data we found that the data quality becomes unacceptable with a sketch vector with size less than 1500. This means that the use of the sketches is more convenient when the number of non-zero values is more than 750, considering that for sending the list of pairs $\langle index, value \rangle$ we need at least 2 integers for each pair. In Section 7 show the impact on data utility of each kind of sketching algorithm presented above.

6 Coordinator Computation

The computation of the coordinator is composed of two main phases: 1) computation of the set of moves and 2) computation of the aggregation of global movements.

Move Vector Computation. The coordinator in an initial setting phase has to send to the nodes the *vector of moves* (Definition 4). The computation of this vector depends on the set of cells that represent the partition of the territory. This partition can be a simple grid or a more sophisticated territory subdivision such as the Voronoi tessellation. The sharing of vector of moves is a requirement of the whole process because each node has to use the same data structure for allowing the coordinator the correct computation of the global flows.

Global Flow Computation. The coordinator has to compute the global vector that corresponds to the global aggregation of movement data in a given time interval τ by composing all the local frequency vectors. As explained in Section 5, the coordinator can receive from the node the information about its frequency vector in two different ways. If it receives the sketch vector $sk(\tilde{f}^{V_j})$ then it reconstructs each frequency vector from the sketch vector, by using the estimation function related to the kind of sketch (see Section 5). Finally, the coordinator computes the global frequency vector by summing the estimate vectors component by component. Clearly the estimate global vector is an approximated version of the global vector obtained by summing the local frequency vectors after the only privacy transformation. While if the coordinator receives from the node the list of non-zero components in the form of list of pairs $\langle index, value \rangle$, then it simply updates the position *index* of the global frequency vector by adding to the frequency *value* to the current value. In this case the global vector is only affected by the transformation due to the privacy; in other words any other approximation is introduced in the data.

7 Experiments

Dataset and Space Tessellation. For our experiments we used a large dataset of GPS vehicles traces, collected in a period from 1st May to 31st May 2011. In our simulation, the coordinator collects the Frequency Vectors (FV) from all the vehicles to determine the Global Frequency Vector (GFV), i.e. the sum all the trajectories crossing any link, at the end of each day, so we defined a series of time intervals τ_i, where each τ_i spans over a single day. In the following we show the resulting GFV for the 25th May 2011, but similar accuracy is observed also for the other days. The GPS traces were collected in the geographical areas around Pisa, in central Italy, and it counts for around 4,200 vehicles, generating around 15,700 trips.

The generalization and aggregation of movement data is based on space partitioning. Our method for territory partitioning extends the data-driven method suggested in [3]. In particular, we allow that dense point clusters can be subdivided into smaller ones, so that the sizes of the resulting Voronoi polygons depends on the point density (large polygons in data-sparse areas and small polygons in data-dense areas).

	α	γ	*size*
$AGMS_{1k}$	0.0447	0.1	1,000
$AGMS_{1.5k}$	0.0447	0.05	1,500
$AGMS_{3k}$	0.03162	0.05	3,000
$AGMS_{5k}$	0.03162	0.01	5,000
$AGMS_{10k}$	0.03162	0.00005	10,000

(a) AGMS sketch

	α	γ	Col.(w)	Rows(d)	$w \times d$
CM_{1k}	0.004	0.1	500	2	1,000
$CM_{1.5k}$	0.004	0.05	500	3	1,500
CM_{3k}	0.002	0.05	1,000	3	3,000
CM_{5k}	0.0008	0.1	2,500	2	5,000
CM_{10k}	0.0008	0.02	2,500	4	10,000

(b) Count-Min Sketch

	α	γ	Col.(w)	Rows(d)	$w \times d$
C_{1k}	0.0447	0.1	500	2	1,000
$C_{1.5k}$	0.0447	0.05	500	3	1,500
C_{3k}	0.03162	0.05	1,000	3	3,000
C_{5k}	0.03162	0.01	1,000	5	5,000
C_{10k}	0.03162	0.00005	1,000	10	10,000

(c) Count Sketch

Table 1. Sketch sizes for different values of α and γ.

Utility Evaluation. The GFV represents the flow values for each link of the spatial tessellation. Each FV received from the vehicles is perturbed by means of a privacy transformation and a possible sketch summarization These two transformations are regulated by two set of parameters: ε for the differential privacy transformation, and α and γ for the sketch summarization. When ε tends to 1 very little perturbation is introduced and this yields a low privacy

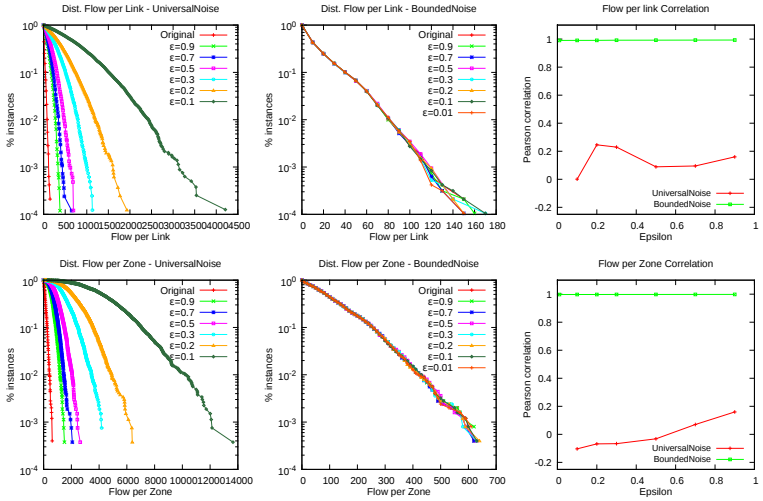


Fig. 1. Cumulative Distribution and PCC distribution of *fpl* and *fpz* after Privacy Transformation.

protection. On the contrary, better privacy guarantees are obtained when ε tends to zero. The two parameters α and γ regulate the compression of the FV to be sent to the coordinator. Table 1 shows how the choice of these two parameters influences the final size of the FV. For example, using Count-Min sketching algorithm with $\alpha = 0.004$ and $\gamma = 0.1$ the original FV of 16k entries is reduced to a vector of 1k cells.

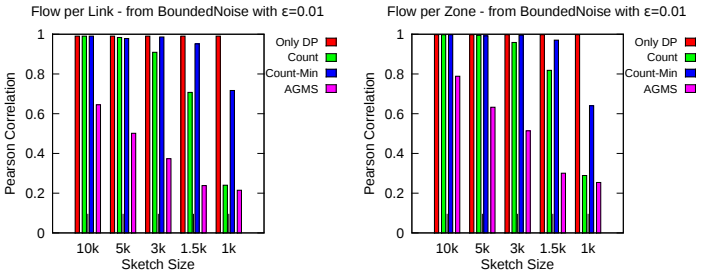


Fig. 2. PCC of *fpl (left)* and *fpz (right)* after the various Sketch Transformation.

Since the two transformations operate on the entries of the FV, and hence on the flows, we study two measures: (1) the *flow per link* (*fpl*), i.e. the directed volume of traffic between two adjacent zones; (2) the *flow per zone* (*fpz*), i.e. the sum of the incoming and outgoing flows in a zone. In the following we first analyze the data utility preserved after the application of the *UniversalNoise* and *BoundedNoise* approaches and then, we also present some empirical results on the impact of different sketching algorithms on the data utility. To this end we compare the cumulative distributions of the above measures before and after the perturbation and we study the flow correlations by using the well-know Pearson Correlation Coefficient (PCC). Finally, we also show how the private data enable some mobility visual analysis.

The plots in the first column of Figure 1 show the resulting cumulative distributions of different *fpl* and *fpz* before and after the privacy transformation *UniversalNoise* with different ε values; while in the second column, we show the impact of the *BoundedNoise* method on the same distributions. We can observe that the *BoundedNoise* approach it is better in terms of data utility as expected from the theoretical results. These results are also confirmed by the third column, where we depict the distribution of the PCC for *fpl* and *fpz* when one of the privacy approaches is applied to the original data by varying ε.
The application of one of the sketching algorithms tends to introduce additional perturbation to the private data. We studied the impact of this perturbation starting from the data transformed by both *UniversalNoise* and *BoundedNoise*. In the following we only discuss the results obtained when we apply the sketch-based transformations to the data obtained by the second approach because the *UniversalNoise* leads to a data with an unreasonable quality that can be only makes worst with an additional transformation.

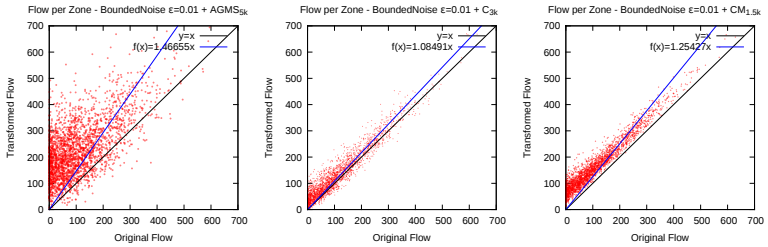


Fig. 3. Correlation between original *fpz* and transformed ones with *BoundedNoise* with $\varepsilon = 0.01$ and various sketching transformations.

In Figure 2 we show how the PCC of *fpl* and *fpz* changes by varying the type of sketching algorithm and the sketch size. Clearly, increasing the size of a sketch we can observe a better correlation due to a more accurate reconstruction of the private flows. The figure depicts the results obtained starting from private data perturbed by *BoundedNoise* with $\varepsilon = 0.01$, that is the setting that provides the best privacy. The general result obtained from the analysis of the different sketches is that *AGMS* algorithm is that with worst performance in terms of data utility and this is clear by analyzing the PCC values in Figure 2. For a more exhaustive analysis of this aspect we also show the scatter plots in Figure 3 that highlight the correlation between each single *fpz* value and its transformed version. From the first plot we can see that the data utility for AGMS already becomes unacceptable when we use a sketch size of $5k$ cells. In general, *Count* and *Count-Min* present good results in terms of PCC even with low sizes. From the results obtained we can argue that the correlation for *Count* is good until sketch size of $3k$ cells and for *Count-Min* we can also use a size of $1.5K$ cells. This is also highlighted in the remaining plots in Figure 3.
Qualitatively, Figures 4 (on the left) show a visually comparison of each Sketch summarization with the original flows. Each flow is draw with arrows with thickness proportional to the volume of trajectories observed on a link. From the figure it is evident how the relevant flows are preserved in all the transformed GFV, revealing the major highways and urban centers.
Similarly, the flow per zone is also preserved, as it is shown in Figure 4 (on the right), where the flow per each cell is rendered with a circle of radius proportional to the difference from the median value of each GFV. The maps allow us to recognize the dense areas (red circles, above the median) separated by sparse areas (blu circle below the median). The high density traffic zones follow the highways and the major city centers along their routes. The comparisons

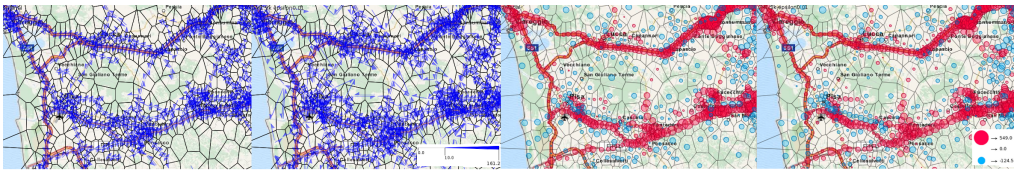


Fig. 4. Traffic and density analysis: comparison between original data and transformed data with both *BoundedNoise* approach with $\varepsilon = 0.01$ and sketch transformation $CM_{1.5k}$

proposed above give the intuition that, while the transformations protect individual sensitive information, the utility of data is preserved.

8 Related Work

The existing methods of privacy-preserving publishing of trajectories can be categorized into two classes: (1) generalization/suppression based data perturbation, and (2) differential privacy. The first category contains some recent works on privacy-preserving publishing of spatio-temporal moving points by using the generalization/suppression techniques [1, 17, 14]. Here, the mostly widely used privacy model is k-anonymity [16], which requires that an individual should not be identifiable from a group of size smaller than k. A general problem of these k-anonymity based privacy preserving techniques is that these techniques assume a certain level of background knowledge of the attackers, which may not be available to the data owner in practice. The recently proposed concept of *differential privacy* (DP) [10] addresses the above issue. The most popular mechanisms to achieve differential privacy is the *Laplace* mechanism that supports queries whose outputs are numerical [10]; it has been widely adopted in many existing work for various data applications (see for example [8, 11]). Regarding publishing differentially private spatial data, Chen et al. [5] propose to release a prefix tree of trajectories with injected Laplace noise. Each node in the prefix tree contains a doublet in the form of $< tr(v), c(v) >$, where $tr(v)$ is the set of trajectories of the prefix v, and $c(v)$ is a version of $|tr(v)|$ with Laplace noise. Compared with our work, the prefix tree in [5] is *data-dependent*, while our frequency vector is *data-independent*. Cormode et al. present a solution to publish differentially private spatial index (e.g., quadtrees and kd-trees) to provide a private description of the data distribution [8]. The spatial index only stores the count of a specific spatial decomposition. It does not store the movement information (e.g., how many individuals move from location i to location j) as in our work. In another paper, Cormode et al. [9] proposes to publish a contingency table of trajectory data. The contingency table can be indexed by specific locations so that each cell in the table contains the number of people who commute from the given source to the given destination. The contingency table is very similar to our frequency vector structure.
There are some work on publishing time-series data with differential privacy guarantee [13, 15]. Since we only consider spatial data, these work are complement to our work.

9 Conclusion

In this paper, we have studied the problem of computing movement data aggregation based on trajectory generalization in a distributed system while preserving privacy. We have proposed two solutions based on the well-known notion of differential privacy that provides very nice data protection guarantees. The two solutions provide a different trade-off between privacy

and data quality. In particular, in our framework each vehicle, before sending the information about its movements within a time interval, applies to the data a transformation for achieving privacy and then, creates a summarization of the private data (by using a sketching algorithm when is convenient) for reducing the amount of information to be communicated. The results obtained in our experiments highlighted the different data utility of the proposed approaches and show how some important properties of the original data are preserved allowing the analyst to use them for important mobility data analysis.
Future investigations could be directed to explore other methods for achieving differential privacy; as an example, it would be interesting to understand the impact of the use of the geometric mechanism instead of the Laplace one for achieving differential privacy.

Acknowledgments. This work has been partially supported by EU FET-Open project LIFT (FP7-ICT-2009-C n. 255951) and EU FET-Open project DATA SIM (FP7-ICT 270833)

References

1. O. Abul, F. Bonchi, and M. Nanni. Never walk alone: Uncertainty for anonymity in moving objects databases. In *ICDE*, pages 376–385, 2008.
2. C. C. Aggarwal and P. S. Yu. On Privacy-Preservation of Text and Sparse Binary Data with Sketches. In *SDM*, 2007.
3. N. Andrienko and G. Andrienko. Spatial generalization and aggregation of massive movement data. *IEEE Transactions on Visualization and Computer Graphics*, 17:205-219, 2011.
4. M. Backes, S. Meiser. Differentially Private Smart Metering with Battery Recharging. *IACR Cryptology ePrint Archive* 2012: 183 (2012)
5. R. Chen, B. C.M. Fung, B. C. Desai, and N. M. Sossou. Differentially private transit data publication: a case study on the Montreal transportation system. *KDD*, 213–221, 2012.
6. G. Cormode and M. N. Garofalakis. Approximate continuous querying over distributed streams. *ACM Trans. Database Syst.*, 33(2), 2008.
7. G. Cormode, M. N. Garofalakis, P. J. Haas, and C. Jermaine. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends in Databases*, 4(1-3):1–294, 2012.
8. G. Cormode, C. M. Procopiuc, D. Srivastava, E. Shen, and T. Yu. Differentially private spatial decompositions. In *ICDE*, pages 20–31, 2012.
9. Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Thanh T. L. Tran. Differentially private summaries for sparse data. In *ICDT*, pages 299–311, 2012.
10. C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284.
11. M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. *Proc. VLDB Endow.*, 3(1-2):1021–1032, 2010.
12. D. Kifer and A. Machanavajjhala. No free lunch in data privacy. In *SIGMOD*, pages 193–204. ACM, 2011.
13. F. McSherry and R. Mahajan. Differentially-private network trace analysis. In *SIGCOMM*, pages 123–134, 2010.
14. A. Monreale, G. Andrienko, N. Andrienko, F. Giannotti, D. Pedreschi, S. Rinzivillo, and S. Wrobel. Movement data anonymity through generalization. *TDP*, 3(2):91–121, 2010.
15. V. Rastogi and S. Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *SIGMOD*, pages 735–746, 2010.
16. P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppresion. In *IEEE Symp. on Research in Security and Privacy*, pages 384–393, 1998.
17. R. Yarovoy, F. Bonchi, L. V. S. Lakshmanan, and W. H. Wang. Anonymizing moving objects: how to hide a mob in a crowd? In *EDBT*, pages 72–83, 2009.

A

Distributed Geometric Query Monitoring using Prediction Models

1. INTRODUCTION

A wide variety of modern applications relies on the *continuous* processing of vast amounts of arriving data in order to support decision making procedures in real time. Examples include network administration, stock market analysis, environmental, surveillance and other application scenarios. These settings are, more often than not, inherently *distributed* in nature. For instance, consider the case of a network operation center where data is produced by hundreds or thousands of routers [Cormode and Garofalakis 2005; Cormode et al. 2005; Cormode and Garofalakis 2008] or the case of environmental as well as control applications where wireless sensor network adoption has become of great importance [Madden et al. 2005].

Due to the distributed nature of data production in the aforementioned scenarios, the major challenge confronted by algorithms dealing with their manipulation is to reduce communication [Cormode and Garofalakis 2005; Cormode et al. 2005; Cormode and Garofalakis 2008; Sharfman et al. 2006; 2007b; 2008; Das et al. 2004]. This happens because the central collection of data is not feasible in large-scale applications. Furthermore, in the case of sensor network deployments, central data accumulation results in depleting the power supply of individual sensors reducing the network lifetime [Madden et al. 2005].

An important query type that is of the essence in the aforementioned fields regards the monitoring of a trigger condition defined upon the range of values a function of interest receives [Sharfman et al. 2006; 2007b; 2008; Huang et al. 2006; Jain et al. 2004; Keralapura et al. 2006; Huang et al. 2007]. For instance, in order to perform spam detection on a number of dispersed mail servers, algorithms base their decisions on whether the value of the information gain function globally exceeds a given threshold [Sharfman et al. 2007b]. Moreover, in the example of the network operation center, Denial-of-Service attacks are detected by attempting to pinpoint strangely high (based on a given threshold) number of distinct source addresses routing packets across various destinations within the network [Das et al. 2004].

Recently, the work in [Sharfman et al. 2006; 2007b] has introduced a generic paradigm for monitoring general (non-linear) functions defined over the average of local vectors maintained at distributed sites. Their proposed geometric approach essentially monitors the area of the input domain where the average vector may lie, rather than monitoring the function's value itself. The monitoring is performed in a distributed manner, by assigning each node a monitoring zone, expressed as a hypersphere, which is nothing more than a subset of the input domain where the average vector may lie. Communication is shown to be necessary only if at least one site considers it likely that the condition of the monitored function may have changed since the last communication between the sites.

In this work, we examine the potentials of a simple (yet powerful), easy to locally maintain approach in order to further reduce transmissions towards the central source. In particular, we foster prediction models so as to describe the evolution of local streams. The adoption of prediction models has already been proven beneficial in terms of bandwidth preservation [Cormode and Garofalakis 2005; Cormode et al. 2005; Cormode and Garofalakis 2008] in distributed settings. Initially, we extend the geometric monitoring framework of [Sharfman et al. 2006; 2007b] and illustrate how it can incorporate predictors, in order to forecast the evolution of local data vectors of sites. We exhibit the way the geometric monitoring framework is modified to encompass constructed predictors and identify the peculiarities occurred upon predictors' adoption. In contrast to the findings of prior works [Cormode and Garofalakis 2005; Cormode et al. 2005; Cormode and Garofalakis 2008], we prove that the mere utilization of local predictions is hardly adequate to guarantee communication preservation even when predictors are quite capable of describing local stream distributions. We then proceed by establishing a theoretically solid monitoring framework that incorporates condi-

tions managing to guarantee fewer contacts with the central source. Eventually, we develop a number of mechanisms, along with extensive speculative analysis, that relax the previously introduced framework, base their function on simpler criteria, and in practice yield significant transmission reduction. Our main contributions are:

- We introduce the adoption of prediction models in the setting of tracking general, non-linear functions utilizing the geometric approach [Sharfman et al. 2006; 2007b]. We exhibit the way prediction models can be locally adopted by sites and we show the characteristics they attribute to the geometric approach. We then illustrate that the initial geometric monitoring framework of [Sharfman et al. 2006; 2007b] is a special case of our, more general, prediction-based geometric monitoring framework.
- We point out the failure of conventional notions of good predictors to be applied in this setting and manage to establish a solid theoretic framework consisting of sufficient conditions that do render prediction models capable of guaranteeing reduced bandwidth consumption.
- We expose a number of novel tracking mechanisms relaxing the previously (hard to verify in a distributed manner) identified sufficient conditions. Using the simplest possible primitives regarding prediction models' behavior, we thoroughly study a number of simplified alternative tracking techniques, possessing the potentials for communication preservation.
- We present a probabilistic analysis on the expected performance of our simplified alternative tracking mechanisms. This analysis describes an ideal case where continuous knowledge of global statistics is available. As in practice that type of statistics requires constant central data collection, we then use the extracted intuition to propose empirical decision making procedures with respect to the best (i.e., the one that protracts central data collection) alternative choice.
- We look into extensions of our techniques for approximate function monitoring scenarios as the core of more generic query answering procedures. We come up with problem transformations which render accurate predictors sufficient to reduce bandwidth consumption. In our study, we introduce novel monitoring techniques tailored for the new setup and accordingly compare their function with the rest of the alternative schemes previously developed in our work.
- We present an extensive experimental analysis using a variety of real data sets, parameters and functions of interest. Our evaluation shows that our approaches can provide significant communication load reduction with savings ranging from 2 times and in some cases reaching 3 orders of magnitude compared to the transmission cost of the original bounding algorithm.

This paper proceeds as follows. In the next section we present related work. In Section 3 we formally present the geometric monitoring framework and we exhibit exemplary prediction models useful for the applications we consider. Section 4 explains the motivation for predictors' adoption and shows how prediction models can be incorporated within the geometric monitoring framework, pointing out how conventional notions of "good" predictors fail to adapt in the current setting. Section 5 presents solid theoretic frameworks that manage to dictate the sufficient conditions for prediction models adoption with provable communication reduction. In Section 6, based on relaxed versions of the previously identified conditions, we propose practical alternatives built upon as simple as possible assumptions on the ability of prediction models to describe incoming data distributions while Section 7 describes decision making procedures for choosing amongst those alternatives. Section 8 explains how our techniques can be adapted to efficiently support prediction-based approximate query answering procedures. Our experimental analysis is incorporated in Section 9. Eventually, Section 10 includes concluding remarks.

2. RELATED WORK

Recently, substantial efforts have been devoted on tracking and querying distributed data streams [Cormode and Garofalakis 2007]. The geometric monitoring framework which is leveraged by our approaches was introduced in [Sharfman et al. 2006; 2007b] and was later enhanced in [Sharfman et al. 2008]. The optimizations proposed in [Sharfman et al. 2008] are orthogonal to

our approaches, but note that the techniques of [Sharfman et al. 2008] either require data to conform with a multivariate normal distribution or entail a number of solutions to a series of optimization problems that may increase the computational load. The latter renders their adoption unaffordable in resource constraint environments such as [Sharfman et al. 2007a; Burdakis and Deligiannakis 2012]. On the contrary, our approaches are based on simple predictors' adoption that remain adaptable to changing data distributions and are easy to maintain even when resource constraints exist. In other work related to the geometric monitoring approach, [Sharfman et al. 2007a] discusses an application of the framework of [Sharfman et al. 2006; 2007b] to clustered sensor network settings, while [Burdakis and Deligiannakis 2012] shows how the tracking scheme can be utilized so as to detect outliers produced by motes. The recent work of [Sagy et al. 2010] adopts the geometric approach and proposes a tentative bound algorithm to monitor threshold queries in distributed databases (rather than distributed data streams) for functions with bounded deviation.

Prediction models in the context of distributed data streams have already been fostered in previous work to monitor one-dimensional quantiles [Cormode et al. 2005] and randomized sketch summaries [Cormode and Garofalakis 2008]. Their adoption has been proven beneficial in terms of reducing the communication burden. Contrary to previous approaches our focus is on the benefits they can provide in the context of the geometric monitoring framework for tracking non-linear threshold functions.

In related work regarding distributed trigger monitoring, [Keralapura et al. 2006] provides a framework for monitoring thresholded counts over distributed data streams, while [Jain et al. 2004] designs techniques that decompose the problem of detecting when the sum of a distributed set of variables exceeds a given threshold. Based on [Jain et al. 2004] anomaly detection techniques are studied in [Huang et al. 2006] and [Huang et al. 2007]. The recent work of [Cormode et al. 2011] provides upper and lower communication bounds for approximate monitoring of thresholded F_p moments, with $p = 0, 1, 2$.

Other works focus on tracking specific types of functions over distributed data streams. The work of [Olston et al. 2003] considers simple aggregation queries over multiple sources, while [Babcock and Olston 2003] focuses on monitoring top-k values. Furthermore, [Das et al. 2004] monitors set-expression cardinalities in a distributed system using a scheme for charging local changes against single site's error tolerance. [Yi and Zhang 2009] considers the problem of tracking heavy hitters and quantiles in a distributed manner establishing optimal algorithms to accomplish the task. Eventually, [Cormode et al. 2007] studies the problem of clustering distributed data streams, while [Zhang et al. 2008] generalizes the previous approach to hierarchical environments.

In the conference version of this paper [Anonymous], we presented the initial framework for monitoring non-linear functions using prediction models. In this work we build on the work of [Anonymous] mainly by extending this framework to perform approximate query answering (Section 8). We also enhance the discussion on the selection of the model to use at each time by adding a theoretical viewpoint on this process (Section 7.1). We further manage to elaborate on the proofs of Corollary 6.3 and Proposition 6.4. Finally, the experimental evaluation in Section 9.4 validates the usability of our newly proposed techniques in approximate query answering scenarios.

3. PRELIMINARIES

In this section we first provide helpful background work related to function monitoring using the geometric approach. We then describe local stream predictors, which have been utilized in past work. The notation used in this paper appears in Table I.

3.1. The Geometric Monitoring Framework

As in previous works [Cormode et al. 2005; Sharfman et al. 2007b; Cormode and Garofalakis 2005; 2008; Sharfman et al. 2008], we assume a distributed, two-tiered setting, where data arrives continuously at n geographically dispersed sites. At the top tier, a central coordinator exists that is capable of communicating with every site, while pairwise site communication is only allowed via the coordinating source.

Table I. Notation used

Symbol	Description
n	The number of sites
S_i	The i-th site
t_s	Timestamp of the last synchronization
$v(t)$	Global measurements vector at time t ($\sum_{i=1}^{n} w_i v_i(t) / \sum_{i=1}^{n} w_i$)
$e(t)$	Estimate vector at time t (equal to $v(t_s)$)
$e^p(t)$	The predicted estimate vector ($\sum_{i=1}^{n} w_i v_i^p(t) / \sum_{i=1}^{n} w_i$)
$v_i(t)$	Local measurements vector at S_i at time t
w_i	Number of data points at S_i
$u_i(t)$	Drift vector (equals to $e(t) + v_i(t) - v_i(t_s)$)
$v_i^p(t)$	Local predictor of S_i at time t
$u_i^p(t)$	Prediction deviation vector ($e^p(t) + v_i(t) - v_i^p(t)$)
$B_c^{\|r\|}$	Local constraint (ball) centered at c with radius $\|r\|$
D_e	Radius of maximum sphere (ball), centered at e, that can be inscribed without violating the threshold surface
D_{e^p}	Radius of maximum sphere (ball), centered at e^p, that can be inscribed without violating the threshold surface

Each site S_i, $i \in [1..n]$ participating at the bottom tier receives updates on its local stream and maintains a d-dimensional local measurements vector $v_i(t)$. The *global measurements vector* $v(t)$ at any given timestamp t, is calculated as the weighted average of $v_i(t)$ vectors, $v(t) = \frac{\sum_{i=1}^{n} w_i v_i(t)}{\sum_{i=1}^{n} w_i}$, where $w_i \geq 0$ refers to the weight associated with a site. Usually, w_i corresponds to the number of data points received by S_i [Sharfman et al. 2006]. Our aim is to continuously monitor whether the value of a function $f(v(t))$, defined upon $v(t)$, lies above/below a given threshold T. We use the term *threshold surface* to denote the area of the input domain where $f(v(t)) = T$.

During the monitoring task using the geometric approach [Sharfman et al. 2006; 2007b], the coordinator may request that all sites transmit their local measurements vectors and subsequently calculates $v(t)$, performs the required check on $f(v(t))$, and transmits the $v(t)$ vector to all sites. The previous process is referred to as a *synchronization* step. Let $v_i(t_s)$ denote the local measurements vector that S_i communicated during the last synchronization process at time t_s. The global measurements vector computed during a synchronization step is denoted as the *estimate vector* e, where $e = \sum_{i=1}^{n} w_i v_i(t_s) / \sum_{i=1}^{n} w_i$.

After a synchronization, sites keep up receiving updates of their local streams and accordingly maintain their $v_i(t)$ vectors. At any given timestamp, each site S_i individually computes $v_i(t) - v_i(t_s)$ and the local *drift vector* $u_i(t) = e + (v_i(t) - v_i(t_s))$. Since

$$v(t) = \frac{\sum_{i=1}^{n} w_i v_i(t)}{\sum_{i=1}^{n} w_i} = e + \frac{\sum_{i=1}^{n} w_i (v_i(t) - v_i(t_s))}{\sum_{i=1}^{n} w_i} = \frac{\sum_{i=1}^{n} w_i u_i(t)}{\sum_{i=1}^{n} w_i}$$

$v(t)$ constitutes a convex combination of the drift vectors. Consequently, $v(t)$ will always lie in the convex hull formed by the $u_i(t)$ vectors: $v(t) \in Conv\ (u_1(t), \ldots, u_n(t))$, as depicted in Figure 1.

Please note that each site can compute the last known value of the monitored function as $f(e)$ and can, thus, determine whether this value lies above/below the threshold T. Since $v(t) \in Conv\ (u_1(t), \ldots, u_n(t))$, if the value of the monitored function in the entire convex hull lies in the same direction (above/below the threshold T) as $f(e)$, then it is guaranteed that $f(v(t))$ will lie in that side. In this case, the function will certainly not have crossed the threshold surface. The key question is: "how

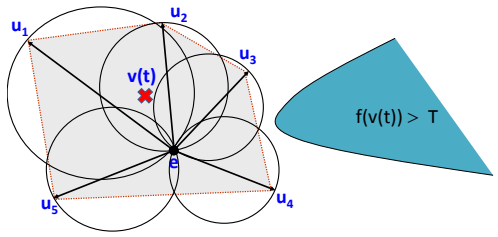


Fig. 1. Demonstration of the geometric framework rationale. *Conv* $(u_1, \ldots, u_n)$ is depicted in gray, while the actual position of e and the current $v(t)$ are shown as well. Black spheres refer to the local constraints constructed by sites to assess possible threshold crossing. $v(t)$ is guaranteed to lie within the union of these locally constructed spheres. Since none of the spheres crosses the threshold surface, the sites are certain that $f(v(t))$ and $f(e)$ are at the same side relative to the threshold T. Hence, no synchronization needs to be performed.

can the sites check the value of the monitored function in the entire convex hull, since each site is unaware of the current drift vectors of the other sites"? This test can be distributively performed as described in Theorem 3.1, while an example (in 2-dimensions) is included in Figure 1.

THEOREM 3.1. *[Sharfman et al. 2006; 2007b] Let* $x, y_1, \ldots, y_n \in R^d$ *be a set of d dimensional vectors. Let Conv* $(x, y_1, \ldots, y_n)$ *be the convex hull of* $x, y_1, \ldots, y_n$. *Let* $B^{\|\frac{x-y_i}{2}\|}_{\frac{x+y_i}{2}}$ *be a ball centered at* $\frac{x+y_i}{2}$ *with a radius of* $\|\frac{x-y_i}{2}\|$ *that is,* $B^{\|\frac{x-y_i}{2}\|}_{\frac{x+y_i}{2}} = \{z \in R^d : \; \|z - \frac{x+y_i}{2}\| \leq \|\frac{x-y_i}{2}\|\}$. *Then, Conv* $(x, y_1, \ldots, y_n) \subset \bigcup_{i=1}^{n} B^{\|\frac{x-y_i}{2}\|}_{\frac{x+y_i}{2}}$.

With respect to our previous discussion x corresponds to e while y_i vectors refer to the drift vectors $u_i(t)$. Hence, sites need to compute their local constraints in the form of $B^{\|\frac{e-u_i(t)}{2}\|}_{\frac{e+u_i(t)}{2}}$ and independently check whether a point within these balls may cause a threshold crossing. If this indeed is the case (an event termed as a *local violation*), a synchronization step takes place. Note that since *Conv* $(e, u_1, \ldots, u_n)$ is a subset of the union of local ball constraints, the framework may produce a synchronization in cases where the convex hull has not actually crossed the threshold surface (false positives).

In summary, each site in the geometric monitoring framework manages to track a subset of the input domain. The overall approach achieves communication savings since the coordinator needs to collect the local measurement vectors of the sites only when a site locally detects (in its monitored area of the input domain) that a threshold crossing may have occurred.

3.2. Local Stream Predictors

We now outline the properties of some prediction model options that have already been proven useful in the context of distributed data streams [Cormode and Garofalakis 2008; Cormode et al. 2005]. Please note, beforehand, that the concept of their adoption is to keep such models as simple as possible, and yet powerful enough to describe local stream distributions. It can easily be conceived that more complex model descriptors can be utilized, which however incur extra communication burden when sites need to contact the coordinating source [Cormode and Garofalakis 2008; Cormode et al. 2005]. In our setting, this translates to an increased data transmission overhead during each synchronization step. In our discussion, hereafter, we utilize the term *predictor* to denote a prediction estimator for future values of a local measurements vector. Using a similar notation to the one of Section 3.1, we employ $v_i^p(t)$ to denote the prediction for the local measurements vector of site S_i at timestamp t.

Table II. Local Stream Predictors' Summary

Predictor	Info.	Pred. Local Vector (v_i^p)
Static	$\emptyset$	$v_i(t_s)$
Linear Growth	$\emptyset$	$\frac{t}{t_s}v_i(t_s)$
Velocity/Acceleration	vel_i	$v_i(t_s)+(t-t_s)vel_i+(t-t_s)^2 accel_i$

The Static Predictor. The simplest guess a site may take regarding the evolution of its local measurements vector is that its coordinates will remain unchanged with respect to the values they possessed in the last synchronization: $v_i^p(t) = v_i(t_s)$. It is also evident that this predictor is trivial to maintain in both the sites and the coordinator. Moreover, it requires no additional information to be communicated towards the coordinator upon a synchronization step. Using the static predictor, in the absence of a synchronization step, the coordinator estimates that $v(t) = e$.

The static predictor may be a good choice only in settings where the evolution of the values in each local measurements vector is unpredictable, or local measurements vectors change rarely.

The Linear Growth Predictor. The next simple, but less restrictive, assumption that can be made is that local vectors will scale proportionally with time. In particular, $v_i^p(t) = \frac{t}{t_s}v_i(t_s)$ which is the only calculation individual sites and the coordinating source need to perform in order to derive an estimation of $v_i(t)$ at any given time. Please note that, using this predictor, the best guess that a coordinator can make for the value of $v(t)$ is equal to $\frac{t}{t_s}v(t_s) = \frac{t}{t_s}e$. As with the Static Predictor, the Linear Growth Predictor requires no additional information to be transmitted upon a synchronization.

We can deduce that the Linear Growth Predictor is built on the assumption that $v_i(t)$ vectors evolve, but that their evolution involves no direction alterations. Consequently, it can be adopted so as to approximate local streams in which $v_i(t)$ vectors' coordinates are expected to uniformly increase by a time dependent factor.

The Velocity/Acceleration Predictor. The Velocity/Acceleration (*VA*) Predictor is a much more expressive predictor. VA employs additional vectors that attempt to capture both the scaling and directional change that $v_i(t)$ may undertake. More precisely, in the *VA* predictor the future value of the local measurements vector is estimated as $v_i^p(t) = v_i(t_s)+(t-t_s)vel_i+(t-t_s)^2 accel_i$. Since the velocity vel_i and the acceleration $accel_i$ of the local stream are capable of expressing both possible types of $v_i(t)$ alterations, it provides an enriched way to approximate its behavioral pattern.

In a way similar to [Cormode and Garofalakis 2008], when a synchronization is about to take place, S_i is required to compute the velocity vector vel_i utilizing a window of the W most recent updates it received. Given that window, the velocity vector can be calculated by computing the overall disposition as the difference between $v_i(t)$ and the local vector instance corresponding to the first position of the window.[1] Scaling this outcome by the time difference between the window extremes provides vel_i. In addition, the $accel_i$ value can be computed as the difference between the current velocity and corresponding velocity calculated in the previous synchronization. Scaling the previous result by $1/(t-t_s)$ computes a proper $accel_i$ vector. Additional approaches based on use of vel_i and $accel_i$ values can be found in [Cormode and Garofalakis 2008].

It is easy to see that the flexibility provided by the *VA* Predictor comes at the cost of the transmission of vel_i (along with $v_i(t)$) during each synchronization. We note that $accel_i$ does not need to be communicated to the coordinator, since the coordinator is already aware of the previously computed velocity vectors of each site.

Table II summarizes the described predictor characteristics. It is important to emphasize that the prediction-based monitoring framework described in the next sections can utilize any predictor and is, thus, not restricted to the predictors presented in this section.

[1]Please note that each update may not arrive at each timestamp. Thus, the timestamp of the first update in the window may in general be different than $t-W+1$.

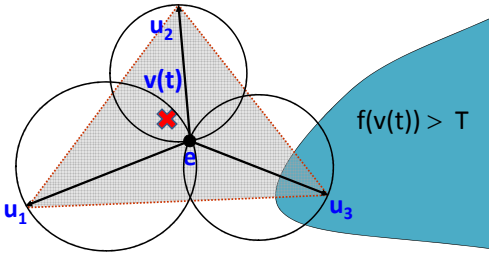


Fig. 2. Motivation for incorporating predictors in the geometric approach. The drift vector u_3 would cause a synchronization in the original framework. However, if the changes could have been predicted, then the coordinator could have expected the value of the estimate vector to actually be close to e and far from the threshold surface.

4. PREDICTION-BASED GEOMETRIC MONITORING

In this section we first motivate the need to incorporate predictors in the geometric monitoring framework and then demonstrate how this can be achieved. We then illustrate that the initial geometric monitoring framework of [Sharfman et al. 2006; 2007b] is a special case of our, more general, prediction-based geometric monitoring framework. Subsequently, we define the notion of a good predictor and demonstrate that good predictors lead to monitoring a smaller subset of the domain space, thus potentially leading to fewer synchronizations and, hence, fewer transmitted messages.

Motivation for Predictors. Figure 2 demonstrates a motivating example of why it may be beneficial to incorporate predictors in the geometric monitoring framework. In the illustrated example, the drift vector u_3 has crossed the threshold surface. Based on their definition, the direction of each drift vector essentially depicts how the values of the corresponding local measurements vector have changed since the last synchronization. Using the geometric monitoring approach, a synchronization will take place because S_3 will detect a threshold crossing. A plausible question is: "Could we avoid such a synchronization step, if the changes in the values of the three local measurements vectors could have been predicted fairly accurate"? For example, if we could have predicted the change (drift) in the local measurements vectors of each site fairly accurately, then we would have determined that $v(t)$ has probably not moved closer to the threshold surface and, thus, avoid the synchronization step. The above example motivates the need for prediction-based geometric monitoring.

How to Incorporate Predictors. As explained in Section 3.2, the coordinator can receive, during a synchronization step, information regarding the predicted local measurements vector $v_i^p(t)$ of each site. Thus, the coordinator will be able to compute an estimation of $v(t)$ provided by the local predictors as: $e^p(t) = \frac{\sum_{i-1}^{n} w_i v_i^p(t)}{\sum_{i-1}^{n} w_i}$, which we will term as the *predicted estimate vector*. Based on $e^p(t)$, we now show that the coordinator can continuously check the potential threshold crossings. However, in this case a synchronization is required only when $e^p(t)$ and $v(t)$ are likely to be placed in different sides of the threshold surface.

In the context of the geometric monitoring framework, we first observe that:

$$v(t) = \frac{\sum_{i=1}^{n} w_i v_i(t)}{\sum_{i=1}^{n} w_i} = e^p(t) + \frac{\sum_{i=1}^{n} w_i (v_i(t) - v_i^p(t))}{\sum_{i=1}^{n} w_i} = \frac{\sum_{i=1}^{n} w_i u_i^p(t)}{\sum_{i=1}^{n} w_i}$$

where $u_i^p(t) = e^p(t) + (v_i(t) - v_i^p(t))$ denotes the vector expressing the *prediction deviation*. Thus, similar to our analysis in Section 3.1, $v(t) \in Conv\ (u_1^p(t), \ldots, u_n^p(t)) \subset \bigcup_{i=1}^{n} B^{\| \frac{e^p(t)-u_i^p(t)}{2} \|}_{\frac{e^p(t)+u_i^p(t)}{2}}$. Since $v(t)$ lies in the convex hull $Conv(u_1^p(t), \ldots, u_n^p(t))$, each site S_i can monitor the ball that has as endpoints of its diameter the estimated predicted vector $e^p(t)$ and its prediction deviation $u_i^p(t)$.

Please note that the geometric monitoring approach of [Sharfman et al. 2006; 2007b] corresponds to utilizing a static predictor (this leads to $v_i^p(t) = v(t_s)$, $e^p(t) = e$ and $u_i^p(t) = u_i(t)$) and is, thus, a special case of our, more general, prediction-based monitoring framework.

Defining a Good Predictor. Upon utilizing a predictor, as long as local forecasts ($v_i^p(t)$) remain sound, we expect that they will approximate the true local vectors $v_i(t)$ to a satisfactory degree at any given timestamp. This means that each $v_i^p(t)$ will be in constant proximity to the $v_i(t)$ vector, when compared to $v_i(t_s)$. Formally:

PROPERTY 1. *A* **Good Predictor** *possesses the property:*

$$\|v_i(t) - v_i^p(t)\| \leq \|v_i(t) - v_i(t_s)\|\ \forall t \geq t_s$$

Property 1 lies, implicitly or not, in the core of predictors' adoption in distributed stream settings. It expresses the notion of a useful, in terms of bandwidth consumption reduction, predictor present in previous works [Cormode and Garofalakis 2005; 2008; Cormode et al. 2005] which have managed to exhibit important improvements by exploiting the above fact. Hence, we start by exploring the benefits of the notion of good predictors expressed by Property 1 within the geometric monitoring setting.

Predictors satisfying Property 1 yield stricter local constraints for the bounding algorithm compared to the original monitoring mechanism (Section 3.1). This happens because $\|v_i(t) - v_i^p(t)\| \leq \|v_i(t) - v_i(t_s)\| \Leftrightarrow \|u_i^p(t) - e^p(t)\| \leq \|u_i(t) - e\|$ and the radius of the constructed balls will always be smaller. An example of prediction-based monitoring is depicted in Figure 3.

Consequently, a good predictor results in the sites monitoring a tighter convex hull, namely *Conv* $(u_1^p(t), \ldots, u_n^p(t))$, than the corresponding convex hull of the original geometric monitoring framework. This yields the construction of tighter local constraints and, as already mentioned, a synchronization is required only when $e^p(t)$ is likely to be placed in a different side of the threshold surface to the one of $v(t)$. Hence, a synchronization is again caused when any ball $B^{\| \frac{e^p(t)-u_i^p(t)}{2} \|}_{\frac{e^p(t)+u_i^p(t)}{2}}$ crosses the threshold surface.

Despite the fact that this mechanism may in practice be useful, it cannot guarantee fewer synchronizations because *Conv* $(u_1^p(t), \ldots, u_n^p(t))$, although tighter, might still be placed closer than *Conv* $(u_1(t), \ldots, u_n(t))$ to the threshold surface. This in turn will cause some $B^{\| \frac{e^p(t)-u_i^p(t)}{2} \|}_{\frac{e^p(t)+u_i^p(t)}{2}}$ to cross the threshold before any $B^{\| \frac{e-u_i(t)}{2} \|}_{\frac{e+u_i(t)}{2}}$ does (Figure 3). This observation shows that the conventional concept of good predictors fails to adapt in the current setting since it does not guarantee by itself fewer synchronizations.

5. STRONG MONITORING MODELS

The concluding observations of Section 4 raise a concern regarding the sufficient conditions that should be fulfilled for the predictors to always yield fewer synchronizations than the original framework. Apparently, this happens when the surface of the monitoring framework devised by the predictors is contained inside the monitored surface of the original framework. In other words, we need to define the prerequisites for constructing local constraints that are always included in $\bigcup_{i=1}^{n} B^{\| \frac{e-u_i(t)}{2} \|}_{\frac{e+u_i(t)}{2}}$

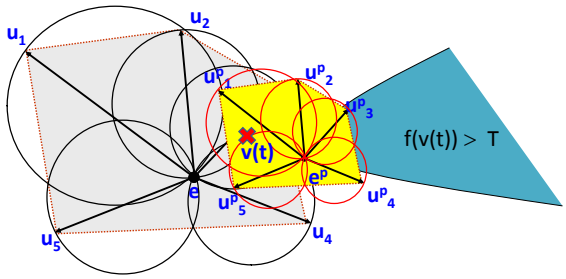


Fig. 3. The red balls demonstrate the local constraints of sites when using a sample good predictor. A good predictor results in the tighter convex hull $Conv\,(u_1^p(t),\ldots,u_n^p(t))$ (depicted in yellow). Here, fewer synchronizations are not guaranteed, since $\bigcup_{i=1}^{n} B^{\|\frac{e^p(t)-u_i^p(t)}{2}\|}_{\frac{e^p(t)+u_i^p(t)}{2}}$ crosses the threshold before $\bigcup_{i=1}^{n} B^{\|\frac{e-u_i(t)}{2}\|}_{\frac{e+u_i(t)}{2}}$.

utilized by the original framework. Let $Sur(P)$ be the surface monitored by any alternative mechanism that adopts predictors while operating. A monitoring model is defined as *strong* if the following property holds:

PROPERTY 2. *A* **Strong** *predictor-based* **Monitoring Model** *possesses the property:* $Sur(P) \subseteq \bigcup_{i=1}^{n} B^{\|\frac{e-u_i(t)}{2}\|}_{\frac{e+u_i(t)}{2}}$

5.1. Containment of Convex Hulls

According to Theorem 3.1, after computing the local drift vectors and prediction deviations, we are free to choose any common, *reference vector* in order to perform the monitoring task. Thus, it is not mandatory for the sites to use e and $e^p(t)$ as a common reference point in order to construct their monitoring zones. In fact, the sites could use any common point as an endpoint of the diameter of their monitoring zones.

An important observation that we prove in this section is that a predictor-based monitoring model satisfies Property 2 when (1) every prediction deviation vector is contained in the convex hull of the estimate vector and the drift vectors defined by the original bounding algorithm, and (2) an appropriate reference vector is selected.

Before proving our observation, we first show that for any triplet of vectors $z,y,x \in R^d$, the condition $z \in B^{\|\frac{y-x}{2}\|}_{\frac{y+x}{2}}$ is equivalent to $\langle x-z, y-z\rangle \leq 0$ where the notation $\langle .,. \rangle$ refers to the inner product of two vectors. Whenever it is appropriate, we omit the temporal reference symbol (t) in the vectors to simplify the exposition.

LEMMA 5.1. $z \in B^{\|\frac{y-x}{2}\|}_{\frac{y+x}{2}}$ *if and only if* $\langle x-z, y-z\rangle \leq 0$.

PROOF. Recall that if $z \in B^{\|\frac{y-x}{2}\|}_{\frac{y+x}{2}}$, then $\|z-\frac{x+y}{2}\| \leq \|\frac{x-y}{2}\|$. This is equivalent to $\frac{1}{4}\langle 2z-(x+y),$ $2z-(x+y)\rangle - \frac{1}{4}\langle x-y, x-y\rangle \leq 0$. Recall that the inner product is distributive, i.e. $\langle a+b, c\rangle = \langle a\,, c\rangle$ $+\langle b\,, c\rangle$, and symmetric, i.e. $\langle a\,, b\rangle = \langle b\,, a\rangle$ Therefore: $\frac{1}{4}\,\langle 2z-(x+y), 2z-(x+y)\rangle - \frac{1}{4}\,\langle x-y, x-y\rangle$ $=$
$\frac{1}{4}\,\langle (z-x)+(z-y), (z-x)+(z-y)\rangle -$
$\frac{1}{4}\,\langle (z-x)-(z-y), (z-x)-(z-y)\rangle =$
$\langle z-x, z-y\rangle = \langle x-z, y-z\rangle\,.$ □

We now proceed to prove in Lemma 5.2 that a predictor-based monitoring model that maintains each prediction deviation vector contained in the convex hull of the drift vectors defined by the

original bounding algorithm is a strong predictor-based monitoring model if it also selects the same reference vector (e.g., e instead of e^p) as the original framework. A direct result is that the area monitored by the sites is a subset of the corresponding area of the original framework. This, in turn, leads to fewer synchronizations, since every time a site detects a potential threshold crossing in the predictor-based monitoring model, at least one site would also have detected the same threshold crossing (for the same vector of the input domain) in the original framework.

LEMMA 5.2. *Let* $u_i^p \in Conv(u_1, ..., u_n)\ \forall i \in \{1..n\}$. *Then*

$$B^{\|\frac{e-u_i^p}{2}\|}_{\frac{e+u_i^p}{2}} \subseteq \bigcup_{i=1}^{n} B^{\|\frac{e-u_i}{2}\|}_{\frac{e+u_i}{2}}.$$

PROOF. For each $u_i^p \in Conv(u_1, ..., u_n)$ there exist λ_1,λ_2, ..., λ_n such that $\lambda_i \geqslant 0$ ($i \in \{1..n\}$), $\sum_{i=1}^{n} \lambda_i = 1$ and $u_i^p = \sum_{i=1}^{n} \lambda_i u_i$. Let $h \in B^{\|\frac{e-u_i^p}{2}\|}_{\frac{e+u_i^p}{2}}$. We will show that for at least one of the u_i vectors, $h \in B^{\|\frac{e-u_i}{2}\|}_{\frac{e+u_i}{2}}$.

According to Lemma 5.1:$\langle h-e, h-u_i^p \rangle \leq 0$. Therefore:

$$\begin{aligned} \langle h-e, h-u_i^p \rangle = \langle h-e, \sum \lambda_i h - \sum \lambda_i u_i \rangle \ = \\ \langle h-e, \sum \lambda_i (h-u_i) \rangle = \sum \lambda_i \langle h-e, h-u_i \rangle \ \leq \ 0 \end{aligned}$$

Since $\lambda_i \geqslant 0$, it follows that for at least one u_i with $\lambda_i > 0$, $\langle h-e,\ h-u_i \rangle \leq 0$, which implies (Lemma 5.1) that $h \in B^{\|\frac{e-u_i}{2}\|}_{\frac{e+u_i}{2}}$. □

A trivial example of a strong predictor-based monitoring model is the static predictor which, as mentioned in Section 4, is equivalent to the original framework of [Sharfman et al. 2006; 2007b].

Unfortunately, the containment constraints are not easily abided by any other chosen predictor and, even if they are, it appears hard to dictate a way that allows sites to distributively identify that fact. We will revisit the convex hull containment issues in Section 6.1.

5.2. Convex Hull Intersection Monitoring

An important observation that we make is that, since $v(t) \in Conv(u_1, \ldots, u_n)$ and $v(t) \in Conv(u_1^p, \ldots, u_n^p)$, these two convex hulls cannot be disjoint (Fig. 3). One could, thus, seek ways to exploit this fact, which limits the possible locations of $v(t)$, in order to reduce the size of the monitoring zones of each site which, in turn, will potentially lead to fewer detected threshold crossings. We thus seek to come up with new local constraints in the context of predictor-based monitoring models that cover the intersection of the two convex hulls and which also fulfill Property 2. To proceed towards that goal we first formally formulate an enhanced version of Property 1.

PROPERTY 3. *A* **Universally Good Predictor** *possesses the property:* $\|v_i(t) - v_i^p(t)\| \leq \|v_j(t) - v_j(t_s)\|$ *for any pair of sites* S_i, S_j

In other words for universally good predictors:

$$\min_{k=1..n} \|e - u_k\| \geq \max_{k=1..n} \|e^p - u_k^p\| \tag{1}$$

Property 3 yields $\|u_i^p - e^p\| \leq \|u_j - e\|$ for any pair of sites S_i, S_j. The latter result is produced by simply adding as well as subtracting e^p, e to the left and right side of its inequality, respectively. The following lemma utilizes this fact to devise appropriate local constraints to be fostered at each site S_i.

LEMMA 5.3. *If Property 3 holds, each site S_i needs to examine whether $B^{\|\frac{e-u_i}{2}\|}_{\frac{e+u_i}{2}} \cap B^{\|e-u_i\|}_{e^p}$ crosses the threshold, since:*

$$Conv(u_1,\ldots,u_n) \cap Conv(u_1^p,\ldots,u_n^p) \subset \bigcup_{k=1}^{n} B^{\|\frac{e-u_k}{2}\|}_{\frac{e+u_k}{2}} \cap B^{\|e-u_k\|}_{e^p}$$

PROOF. We will demonstrate than any vector $h \in R^d$ that lies in $Conv\ (u_1, \ldots, u_n) \cap Conv\ (u_1^p, \ldots, u_n^p)$ is also included in at least one intersection $B^{\|\frac{e-u_k}{2}\|}_{\frac{e+u_k}{2}} \cap B^{\|e-u_k\|}_{e^p}$ of a site S_k ($k \in \{1..n\}$). What is certain is that, due to Property 3, $h \in Conv\ (u_1^p, \ldots, u_n^p) \Rightarrow h \in \bigcup_{k=1}^{n} B^{\|\frac{e^p-u_k^p}{2}\|}_{\frac{e^p+u_k^p}{2}} \Rightarrow h \in \bigcup_{k=1}^{n} B^{\|e^p-u_k^p\|}_{e^p} \Rightarrow$ $h \in B^{\|e-u_k\|}_{e^p}$. Since h is definitely contained as well in at least one of the balls $B^{\|\frac{e-u_k}{2}\|}_{\frac{e+u_k}{2}}$ (Theorem 3.1) constructed by the sites, which implies that h will be examined by at least one site. The proof follows immediately. □

According to Lemma 5.3, universally good predictors guarantee Property 2, thus leading to a decreased synchronization frequency. Nevertheless, Lemma 5.3 simultaneously assumes that predictors are always universally good and no information sharing exists between the sites. In a large scale distributed scenario, however, some sites may adhere to neither Property 3, nor Property 1. Obviously, any efficient monitoring algorithm has to take into consideration such situations and guarantee the "correctness" of the monitoring model together with Property 2. Note that the term correctness refers to the ability of the monitoring algorithm to ensure that the intersection is always covered by the union of sites' local constraints. Our evaluation of the overhead required to monitor that Lemma 5.3 is satisfied, showed that this overhead is significant and that it outweighted the benefits of the mechanism. Thus, we seek to devise alternative implementations that are based on more relaxed conditions.

6. SIMPLIFIED ALTERNATIVES

6.1. Relaxing the Containment Condition

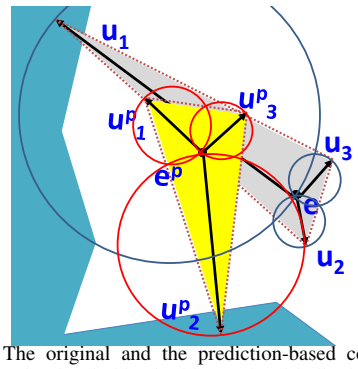
The containment of convex hulls (Section 5.1), as a sufficient prerequisite to achieve accordance with Property 2 is seemingly hard to achieve, let alone come up with ways to continuously check it in a distributive manner. To confront the above drawbacks we investigate an alternative approach which relaxes that condition. Instead of distributively checking the containment condition, we direct our interest to the more practical alternative of making it likely. Intuitively, we are looking for a way to monitor $v(t)$ such that:

Requirement 1: the local constraints in the shape of constructed balls are tighter than those of the original framework (Section 3.1)

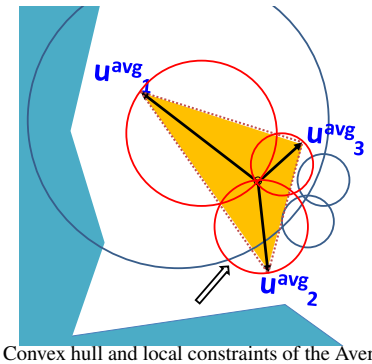
Requirement 2: the choice of the reference point should be as close as possible to e (due to the establishment of Lemma 5.2)

since this pair of requirements renders the containment of new constraints in $\bigcup_{i=1}^{n} B^{\|\frac{e-u_i}{2}\|}_{\frac{e+u_i}{2}}$ more likely. Furthermore, we wish to invent an algorithm that avoids any communication among the sites, unless a threshold crossing is observed.

Since $v(t) = \frac{\sum_{i=1}^{n} w_i u_i^p}{\sum_{i=1}^{n} w_i}$ and $v(t) = \frac{\sum_{i=1}^{n} w_i u_i}{\sum_{i=1}^{n} w_i}$, for any $\mu \varepsilon R$ we can express the true global vector as $v(t) = \frac{\sum_{i=1}^{n} w_i(\mu u_i^p + (1-\mu)u_i)}{\sum_{i=1}^{n} w_i}$. So, in order to monitor the current status of the true global vector we may reside



(a) The original and the prediction-based convex hulls cross the threshold. The blue and red balls depict the areas monitored by each site, correspondingly, for the original and the prediction-based frameworks. S_2 violates Property 1 producing a larger prediction deviation (u_2^p) than the corresponding drift vector's (u_2) length.



(b) Convex hull and local constraints of the Average model. Threshold crossing is prevented with stricter local constraints (except for S_2) and increased $\bigcup_{i=1}^{n} B^{\|\frac{e^p+e}{4}-\frac{u_i^p+u_i}{4}\|}_{\frac{e^p+e+u_i^p+u_i}{4}}$ area contained in the constraints of the original bounding algorithm.

Fig. 4. The effect of the Average model adoption

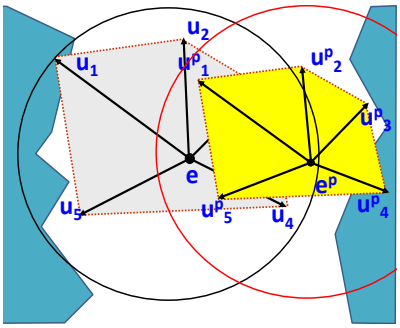


Fig. 5. Loosened Intersection Monitoring. $\max_{i=1..n} B_e^{\|e-u_i\|}$, $\max_{i=1..n} B_{e^p}^{\|e-u_i\|}$ are produced by S_1 which is the one that checks $\max_{i=1..n} B_e^{\|e-u_i\|} \cap \max_{i=1..n} B_{e^p}^{\|e-u_i\|}$. No threshold crossing occurs despite that individual convex hulls violate the threshold surface.

to a new convex hull, namely $Conv\ (\mu u_1^p + (1-\mu)u_1, \ldots, \mu u_n^p + (1-\mu)u_i)$. We then find ourselves concerned with identifying a value for μ that may fulfill Requirements 1 and 2.

LEMMA 6.1. *For any $\frac{1}{2} \leq \mu \leq 1$, when Property 1 holds, tighter local constraints compared to the framework of Section 3.1 are guaranteed, i.e.:* $\|(\mu u_i^p + (1-\mu)u_i) - (\mu e^p + (1-\mu)e)\| \leq \|u_i - e\|$

PROOF.

$$\|u_i^p - e^p\| \leq \|u_i - e\| \overset{\mu \geq 0}{\Leftrightarrow} \|\mu u_i^p - \mu e^p\| \leq \|\mu u_i - \mu e\| \Leftrightarrow$$

$$\|\mu u_i^p - \mu e^p + (1-\mu)(u_i - e) + (\mu - 1)(u_i - e)\| \leq \|\mu u_i - \mu e\|$$

By the triangle inequality:

$$\|\mu u_i^p - \mu e^p + (1-\mu)(u_i - e)\| - \|(\mu - 1)(u_i - e)\| \leq \|\mu u_i - \mu e\| \Leftrightarrow$$

$$\|\mu u_i^p - \mu e^p + (1-\mu)(u_i - e)\| \leq (2\mu - 1)\|u_i - e\|$$

Obviously, $(2\mu-1) \geq 0 \Leftrightarrow \mu \geq \frac{1}{2}$. Additionally, the balls that are build by the original framework possess a radius of $\frac{\|u_i-e\|}{2}$ and as a result, for $\mu \leq 1$ we obtain:

$$\|\mu u_i^p - \mu e^p + (1-\mu)(u_i - e)\| \leq (2\mu-1)\|u_i - e\| \leq \|u_i - e\|$$

The latter inequality completes the proof. □

The lemma above provides a rough upper, as well as lower, bound to the value of μ such that $\|(\mu u_i^p + (1-\mu)u_i) - (\mu e^p + (1-\mu)e)\| \leq \|u_i - e\|$ in every site. This means that sites construct tighter constraints than the ones they possessed using the original framework.

The Average Model. Lemma 6.1 shows that setting $\mu = \frac{1}{2}$ meets Requirement 1 and simultaneously provides beforehand some minimum knowledge with respect to the closest we can move $\mu e^p + (1-\mu)e$ towards e for Requirement 2 to be satisfied as well. Based on these, we are able to devise a first simpler alternative to the containment of convex hulls notion, which we term as the *"Average model"*. The Average model monitors $Conv\ (\frac{u_1^p+u_1}{2}, \ldots, \frac{u_n^p+u_n}{2}) \subseteq \bigcup_{i=1}^{n} B\left(\frac{e^p+e+u_i^p+u_i}{4}, \|\frac{e^p+e}{4} - \frac{u_i^p+u_i}{4}\|\right)$ by a priori picking a value of $\mu = \frac{1}{2}$.

Figure 4 depicts an example of the Average model adoption, where both the original and the prediction-based convex hulls cross the threshold surface in different areas (we included three sites in this example to simplify the exposition). In Figure 4(a), notice that for S_2 Property 1 is violated. Despite this fact, as shown in Figure 4(b), the Average model can still ward off threshold crossing, nearly achieving containment of its constraints (balls) in those of the original bounding algorithm.

The Safer Model. We now discuss an alternative model that relaxes Requirement 2. Following a rationale similar to that of [Sharfman et al. 2008], we observe that at any given timestamp, the sites can individually choose the reference point $\mu e^p + (1-\mu)e, \frac{1}{2} \leq \mu \leq 1$ which is farther from the threshold surface and, at the same time, ensures smaller local constraints. Note that by being far from the threshold surface, a reference point makes the local constraints of any predictor based monitoring model less possible to cause a crossing of the threshold surface [Sharfman et al. 2008]. This second alternative is termed as the *"Safer model"*.

At the first step of the algorithm, every site starts with $\mu_1 = \frac{1}{2}$ and calculates $\mu_1 e^p + (1-\mu_1)e$. In addition, let e_1^* denote the vector lying on the threshold surface and being the closest to $\mu_1 e^p + (1-\mu_1)e$ [2]. Every site is capable of individually computing $\|\mu_1 e^p + (1-\mu_1)e - e_1^*\|$ and, thus, determine the distance the first examined reference point yields. To restrain the computational intensiveness of the technique, we define a number of allowed steps θ, such that in every subsequent step $1 \leq j \leq \theta$ the sites employ a value of $\mu_j = \mu_{j-1} + \frac{1}{2\theta}$ until $\mu_\theta = 1$. Eventually, the μ_j value that induces the largest distance is chosen. Notice that using this framework, the sites can reach a consensus regarding μ without any additional communication. This happens due to the fact that the choice of the final μ is based on common criteria related to the threshold surface and the e, e^p vectors that are known to all sites.

6.2. Loosened Intersection Monitoring

So far in this section we have proposed simplistic alternatives that relax the convex hull containment condition that was discussed in Section 5.1. The presented (*Average* and *Safer*) predictor based monitoring models do manage to avoid any direct communication between the sites unless a threshold crossing is detected. Although they do not necessarily abide by Property 2, these models encompass Requirements 1 and 2 (for the Average model) and are, thus, in practice likely to substantiate a condition that is hard to check in a distributed manner.

We next aim at inventing a loosened version for the intersection monitoring model of Section 5.2. As previously, we wish to come up with a mechanism that avoids any communication between sites

[2]The details on how to compute e^* can be found in [Sharfman et al. 2008]

unless a threshold crossing happens and simultaneously make Property 2 highly likely. Property 1 is again set as a simple prerequisite, but note that all our algorithms in this section remain correct even if it does not hold, since local constraints still totally cover the monitored area of the input domain. In Section 5.2 we saw that $v(t) \in Conv(u_1, \cdots, u_n) \cap Conv(u_1^p, \cdots, u_n^p)$ while in this section we demonstrated that $v(t)$ also lies in any $Conv\ (\mu u_1^p + (1-\mu)u_1, \cdots, \mu u_n^p + (1-\mu)u_i)$ which for $\frac{1}{2} \leq \mu \leq 1$ possesses the desired characteristics formulated in Requirements 1 and 2. The following lemma provides a primitive result on how the intersection monitoring can be achieved using the aforementioned logic. For ease of exposition, we use $Conv_{\cap}^3$ to denote the triple intersection of these three (original, predicted and weighted) convex hulls, while $Sur(Conv_{\cap}^3) \equiv \underset{i=1..n}{max} B_e^{\|e-u_i\|} \cap \underset{i=1..n}{max} B_{e^p}^{\|e^p-u_i^p\|} \cap \underset{i=1..n}{max} B_{\mu e^p+(1-\mu)e}^{\|\mu e^p+(1-\mu)e-\mu u_i^p-(1-\mu)u_i\|}$, where $\underset{i=1..n}{max} B_c^{\|r\|}$ denotes the corresponding (in each maximization term) ball of maximum radius.

LEMMA 6.2. *For any $\mu \in R$, the area inscribed in $Conv_{\cap}^3$ is covered by the region induced by $Sur(Conv_{\cap}^3)$.*

PROOF. Initially notice that:

$$Conv(u_1, \ldots, u_n) \subset \bigcup_{i=1}^{n} B_{\frac{e+u_i}{2}}^{\|\frac{e-u_i}{2}\|} \subset \underset{i=1..n}{max} B_e^{\|e-u_i\|} \tag{2}$$

$$Conv(u_1^p, \ldots, u_n^p) \subset \bigcup_{i=1}^{n} B_{\frac{e^p+u_i^p}{2}}^{\|\frac{e^p-u_i^p}{2}\|} \subset \underset{i=1..n}{max} B_{e^p}^{\|e^p-u_i^p\|} \tag{3}$$

$$Conv(\mu u_1^p + (1-\mu)u_1, \ldots, \mu u_n^p + (1-\mu)u_n) \subset$$
$$\bigcup_{i=1}^{n} B_{\frac{\mu e^p+(1-\mu)e+\mu u_i^p+(1-\mu)u_i}{2}}^{\|\frac{\mu e^p+(1-\mu)e-\mu u_i^p-(1-\mu)u_i}{2}\|} \subset \underset{i=1..n}{max} B_{\mu e^p+(1-\mu)e}^{\|\mu e^p+(1-\mu)e-\mu u_i^p-(1-\mu)u_i\|} \tag{4}$$

So each time the maximum balls cover the corresponding convex hulls. We want to prove that the intersection of the latter balls also covers the intersection of the convex hulls. The proof will be derived by contradiction.

Suppose that a vector $h \in Conv_{\cap}^3$ exists. Now assume that the vector h does not lie in at least one of $\underset{i=1..n}{max} B_e^{\|e-u_i\|}$, $\underset{i=1..n}{max} B_{e^p}^{\|e^p-u_i^p\|}$, or $\underset{i=1..n}{max} B_{\mu e^p+(1-\mu)e}^{\|(\mu e^p+(1-\mu)e)-(\mu u_i^p+(1-\mu)u_i)\|}$. However, this would violate at least one of the Propositions 2,3,4, which is a contradiction. This concludes the proof. □

Reckon, however, that the ascertainment of Lemma 6.2 cannot be tracked in a distributed manner. This happens because the site that determines each maximum ball may be different. Should Property 1 and, thus, (for $\frac{1}{2} \leq \mu \leq 1$) Lemma 6.1 hold, what sites actually need to perform so that they can distributively track $Conv_{\cap}^3$ is to use $B_e^{\|e-u_i\|}$, $B_{e^p}^{\|e-u_i\|}$ and $B_{\mu e^p+(1-\mu)e}^{\|e-u_i\|}$. To understand this, please observe that if both $\|e^p - u_i^p\|$, $\|(\mu e^p + (1-\mu)e) - \mu u_i^p - 1 - \mu)u_i\| \leq \|e - u_i\|$ (due to Property 1 and Lemma 6.1, respectively), it is evident that: $Sur(Conv_{\cap}^3) \subset \underset{i=1..n}{max} B_e^{\|e-u_i\|} \cap \underset{i=1..n}{max} B_{e^p}^{\|e-u_i\|} \cap \underset{i=1..n}{max} B_{\mu e^p+(1-\mu)e}^{\|e-u_i\|}$.

Hence, the site that possesses the maximum $\|e - u_i\|$ will check whether the intersection of its locally constructed balls crosses the threshold. Provided that the intersection of the local balls does not cross the threshold at any site (and, thus, at the site with the maximum $\|e-u_i\|$ as well), synchronization can safely be avoided. At this point, we would be interested in identifying proper values for μ that refine the range ($\frac{1}{2} \leq \mu \leq 1$) established in Lemma 6.1. Nonetheless, the following corollary

shows that if we have to employ $\|e-u_i\|$ as the radius of the balls, $\underset{i=1..n}{max} B^{\|e-u_i\|}_{\mu e^p+(1-\mu)e}$ does not refine the intersection outcome.

COROLLARY 6.3. *For any* $0 \leq \mu \leq 1, \underset{i=1..n}{max} B^{\|e-u_i\|}_{\mu e^p+(1-\mu)e}$ *does not refine the region induced by* $\underset{i=1..n}{max} B^{\|e-u_i\|}_{e} \cap \underset{i=1..n}{max} B^{\|e-u_i\|}_{e^p}$.

PROOF. Assume that there exists a vector $h \in \underset{i=1..n}{max} B^{\|e-u_i\|}_{e} \cap \underset{i=1..n}{max} B^{\|e-u_i\|}_{e^p}$. Then:

$$\|h-e\| \leq \underset{i=1..n}{max}\|e-u_i\| \overset{1\geq\mu}{\Rightarrow} (1-\mu)\|h-e\| \leq (1-\mu)\underset{i=1..n}{max}\|e-u_i\|$$

and $\|h-e^p\| \leq \underset{i=1..n}{max}\|e-u_i\| \overset{\mu\geq 0}{\Rightarrow} \mu\|h-e^p\| \leq \mu \underset{i=1..n}{max}\|e-u_i\|$.

Summing the above and merely applying the triangle inequality we obtain:

$$\|h-(\mu e^p+(1-\mu)e)\| \leq \mu\|h-e^p\|+(1-\mu)\|h-e\| \leq \underset{i=1..n}{max}\|e-u_i\|$$

The latter result exhibits that if a vector lies in the intersection of $\underset{i=1..n}{max} B^{\|e-u_i\|}_{e} \cap \underset{i=1..n}{max} B^{\|e-u_i\|}_{e^p}$, it will definitely lie in the $\underset{i=1..n}{max} B^{\|e-u_i\|}_{\mu e^p+(1-\mu)e}$ as well. Consequently that ball does not contribute to refining the monitored area. □

Corollary 6.3 is true for any $0 \leq \mu \leq 1$, but note that in order to ensure $\|e^p-u^p_i\| \leq \|e-u_i\|$ and $\|(\mu e^p+(1-\mu)e)-\mu u^p_i-(1-\mu)u_i\| \leq \|e-u_i\|$, we had already assumed that $\frac{1}{2} \leq \mu \leq 1$. Hence it suffices to check whether the pair $B^{\|e-u_i\|}_{e} \cap B^{\|e-u_i\|}_{e^p}$ crosses the threshold in at least one site [3]. Figure 5 provides an exemplary application of the intersection monitoring procedure described so far, where $\underset{i=1..n}{max} B^{\|e-u_i\|}_{e}$, $\underset{i=1..n}{max} B^{\|e-u_i\|}_{e^p}$ are produced by S_1.

On the other hand, following an intuition similar to the one utilized in the Average model, an alternative is to track $Conv\ (u^p_1, \cdots, u^p_n) \cap Conv\ (\frac{u^p_1+u_1}{2}, \cdots, \frac{u^p_n+u_n}{2})$ instead. In other words, this time each site S_i needs to individually construct two balls using e^p and $\frac{e^p+e}{2}$ as centers and $M = max\{\|e^p-u^p_i\|, \|\frac{e^p+e}{2}-\frac{u^p_i+u_i}{2}\|\}$ as the common radius (please note that M refers to the maximum of the pair of local radii). Subsequently, a synchronization is caused when at least one S_i detects that the locally constructed intersection crosses the threshold.

We conclude our study by showing the condition which makes the latter intersection tracking preferable as it results in smaller local constraints compared to $\underset{i=1..n}{max} B^{\|e-u_i\|}_{e} \cap \underset{i=1..n}{max} B^{\|e-u_i\|}_{e^p}$.

PROPOSITION 6.4. *When Property 1 holds and* $\underset{i=1..n}{max} B^{\|e-u_i\|}_{e} \supseteq \underset{i=1..n}{max} B^{M}_{\frac{e^p+e}{2}}$, *then:* $\underset{i=1..n}{max} B^{M}_{e^p} \cap \underset{i=1..n}{max} B^{M}_{\frac{e^p+e}{2}} \subseteq \underset{i=1..n}{max} B^{\|e-u_i\|}_{e} \cap \underset{i=1..n}{max} B^{\|e-u_i\|}_{e^p}$

PROOF. For any vector $h \in \underset{i=1..n}{max} B^{M}_{e^p} \cap \underset{i=1..n}{max} B^{M}_{\frac{e^p+e}{2}}$ we have:

$$\|h-e^p\| \leq M \overset{\|e^p-u^p_i\|\leq\|e-u_i\|}{\leq} \underset{}{max}\|e-u_i\|$$

[3] Even if the site that determines the maximum radius finds that Property 1 does not hold, Corollary 6.3 is still valid upon replacing $\|e-u_i\|$ with $\|e^p-u^p_i\|$.

which entails that $h \in \underset{i=1..n}{max} B_{e^p}^{\|e-u_i\|}$. If in addition $\underset{i=1..n}{max} B_{e}^{\|e-u_i\|} \supseteq \underset{i=1..n}{max} B_{\frac{e^p+e}{2}}^{M}$ then $h \in \underset{i=1..n}{max} B_{e}^{\|e-u_i\|}$ as well. Hence $h \in \underset{i=1..n}{max} B_{e}^{\|e-u_i\|} \cap \underset{i=1..n}{max} B_{e^p}^{\|e-u_i\|}$ □

7. CHOOSING AMONGST ALTERNATIVES

So far, we investigated a number of simpler alternatives that loosen the strong monitoring frameworks of Section 5, i.e., the convex hull containment as well as the intersection monitoring framework. We based our analysis on Property 1 as an intuitive assumption also employed in past studies [Cormode et al. 2005; Cormode and Garofalakis 2005; 2008] and evolved it to practical tracking mechanisms together with appropriate speculative analysis. Nonetheless, upon relaxing the monitoring conditions we also relaxed their conformity to Property 2, i.e., the prerequisite for strong predictor-based monitoring models. Since the coordinator is supposed to a priori dictate the predictor-based tracking alternative that should be uniformly utilized by sites at least until the next synchronization, we need to provide a decision making mechanism that enables it choose among the available options and adjust its decisions on their anticipated performance with respect to communication savings.

The available tracking options that do not belong (excluding the trivial choice of the original framework) to the strong predictor-based monitoring models' class include:

- Model 0: Monitoring of $Conv(u_1, \ldots u_n)$ as in Section 3.1
- Model 1: Monitoring of $Conv(u_1^p, \ldots, u_n^p)$ as in Section 4
- The Average model
- The Safer model
- Intersection 1: Tracking of $\underset{i=1..n}{max} B_{e}^{\|e-u_i\|} \cap \underset{i=1..n}{max} B_{e^p}^{\|e-u_i\|}$
- Intersection 2: Tracking of $\underset{i=1..n}{max} B_{e^p}^{M} \cap \underset{i=1..n}{max} B_{\frac{e^p+e}{2}}^{M}$

In this section we first seek to examine in Section 7.1 whether the selection of the used monitoring model can be decided based on probabilistic models that try to estimate the probability of a threshold violation using each model. We describe the limitations and difficulties of basing our decisions on the results of our probabilistic analysis and then (in Section 7.2) seek to devise an adaptive way of chosing amongst the alternative models, based on statistics collected during the operation of the algorithm.

7.1. A Probabilistic Viewpoint

We initially seek to analyze the estimated benefit of all the presented tracking mechanisms from a probabilistic point of view. Our analysis formulates and subsequently examines a worst case scenario for each of the proposed models. Eventually, it attempts to compare their expected performance with respect to the communication load.

Going bach to the example depicted in Figure 3, we recall that a tighter convex hull by a prediction models may not necessarily result in fewer local threshold crossings (local violations). For example, in the example of Figure 3, the main reason why the constraints of the prediction based convex hull cause a threshold crossing is the distance of e^p from the threshold surface. As a consequence, despite the fact that $Conv(u_1, ..., u_n)$ yields larger balls, the second factor that affects the communication savings of the mere $Conv\ (u_1^p, \ldots, u_n^p)$ monitoring is the distance of the common reference point from the threshold surface. This fact was also noted when presenting the rationale behind the safer model's adoption throughout our study.

Figure 6 depicts the maximum balls, centered at e and e^p respectively, that can be inscribed without crossing the threshold surface. Let D_e and D_{e^p} denote the radius of these maximum balls, respectively. Then:

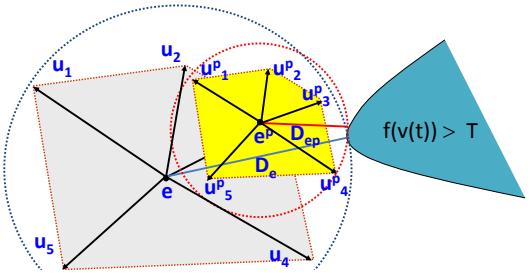


Fig. 6. Maximum spheres centered at the reference points e and e^p, that can be inscribed without violating the threshold surface. The radius of these spheres is denoted as D_e and D_{e^p}, respectively. Apart from the size of the local constraints the position of the utilized reference points determines the probability of a synchronization occurrence.

$$u_i \in B_e^{D_e} \Rightarrow B^{\|\frac{e-u_i}{2}\|}_{\frac{e+u_i}{2}} \subset B_e^{\|\,e-u_i\|} \subset B_e^{D_e}, \forall i \in \{1..n\}$$

and similarly:

$$u_i^p \in B_{e^p}^{D_{e^p}} \Rightarrow B^{\|\frac{e^p-u_i^p}{2}\|}_{\frac{e^p+u_i^p}{2}} \subset B_{e^p}^{\|\,e^p-u_i^p\|} \subset B_{e^p}^{D_{e^p}}, \forall i \in \{1..n\}$$

For any drift vector u_i to cause a violation, it needs to exit the corresponding ball. Therefore, utilizing Markov's Inequality [Garofalakis et al. 2002], we can bound the probability of a threshold crossing in the original monitoring framework [Sharfman et al. 2007b] as follows:

$$Pr^{Model0}_{violation_i} \leq Pr\{\|e-u_i\| \geq D_e\} \leq \frac{E(\|e-u_i\|)}{D_e}$$

In the same way, for $Conv(u_1^p, ..., u_n^p)$ monitoring:

$$Pr^{Model1}_{violation_i} \leq Pr\{\|e^p-u_i^p\| \geq D_{e^p}\} \leq \frac{E(\|e^p-u_i^p\|)}{D_{e^p}}$$

Notice that as long as the norm of the *expected drift* does not exceed the radius of $B_e^{D_e}$, then $E(\|e-u_i\|) \leq D_e$ (a similar discussion also holds for the prediction deviation vectors case). This ensures that the provided upper bound receives a meaningful value ≤ 1. Similar bounds can be extracted for the case of the average (for $\mu = \frac{1}{2}$) as well as the safer model:

$$Pr^{Safer}_{violation_i} \leq Pr\{\|\mu e^p + (1-\mu)e - (1-\mu)u_i + \mu u_i^p\| \geq D_{\mu e^p+(1-\mu)e}\}$$

$$\leq \frac{E(\|\mu e^p + (1-\mu)e - (1-\mu)u_i + \mu u_i^p\|)}{D_{\mu e^p+(1-\mu)e}}$$

Please recall that the Safer Model essentially seeks to maximize the denominator of the above fraction. Moreover, due to Property 1, the Safer Model can also yield a reduced nominator compared to the original monitoring framework. Based on Lemma 6.1, for $\frac{1}{2} \leq \mu \leq 1$:

$$\|(\mu u_i^p + (1-\mu)u_i) - (\mu e^p + (1-\mu)e)\| \leq \|u_i - e\| \Rightarrow$$

$$E(\|\mu e^p + (1-\mu)e - (1-\mu)u_i + \mu u_i^p\|) \leq E(\|e-u_i\|).$$

Of course, these ascertainments should come as no surprise given our discussion in Section 6.

For a synchronization to take place, at least one out of the n sites needs to cause a violation. Assuming that the data streams arriving at the various sites do not exhibit any dependency, the probability of a synchronization, taking Model 1 for instance, can be bounded by:

$$Pr_{Sync}^{Model1} = 1 - \Pi_{i=1}^{n}(1 - Pr_{violation_i}^{Model1}) \leq 1 - \frac{\Pi_{i=1}^{n}(D_{e^p} - E(\|e^p - u_i^p\|)}{(D_{e^p})^n}.$$

Similar results can be extracted for both the original and the safer model.

Regarding the intersection monitoring frameworks, we point out that the probability of violating the threshold surface can be upper bounded by the probability that local spheres ($\max_{i=1..n} B_e^{\|e-u_i\|}$ and $\max_{i=1..n} B_{e^p}^{\|e-u_i\|}$) simultaneously cause a violation. The upper bound is valid since, while these local spheres may both cause a threshold crossing at the same time, the monitored intersection may still avoid a violation. However, the contrary is not possible - if the monitored intersection results in a threshold crossing, then both local spheres result in a threshold crossing as well. Therefore:

$$Pr_{violation}^{Intersection1} \leq Pr\{\max_{i=1..n}\|e-u_i\| \geq D_e \cap \max_{i=1..n}\|e-u_i\| \geq D_{e^p}\} =$$

$$Pr\{\max_{i=1..n}\|e-u_i\| \geq max\{D_e, D_{e^p}\}\} \leq \frac{E(\max_{i=1..n}\|e-u_i\|)}{max\{D_e, D_{e^p}\}}$$

The corresponding bound for $Intersection2$ can be derived as $\frac{E(M)}{max\{D_{e^p}, D_{\frac{e^p+e}{2}}\}}$. Our intersection monitoring schemes determine whether to call for a synchronization relying only on the maximum inscribed sphere. Therefore, the bound on the violation probability provided above also constitutes a bound on the probability of a synchronization. Given the above, Table III summarizes the worst case bounds for the probability of synchronization per epoch.

Table III. Probability of synchronizations of the proposed alternative tracking schemes

Alternative	Probability of Synchronizations
Model 0	$O(1 - \frac{\Pi_{i=1}^{n}(D_e - E(\|e-u_i\|)}{(D_e)^n})$
Model 1	$O(1 - \frac{\Pi_{i=1}^{n}(D_{e^p} - E(\|e^p-u_i^p\|)}{(D_{e^p})^n})$
Safer & Average Models	$O(1 - \frac{\Pi_{i=1}^{n}(D_{\mu e^p+(1-\mu)e} - E(\|(\mu u_i^p+(1-\mu)u_i)-(\mu e^p+(1-\mu)e)\|)}{(D_{\mu e^p+(1-\mu)e})^n})$
Intersection 1	$O(\frac{E(\max_{i=1..n}\|e-u_i\|)}{max\{D_e, D_{e^p}\}})$
Intersection 2	$O(\frac{E(M)}{max\{D_{e^p}, D_{\frac{e^p+e}{2}}\}})$

Difficulties of Using our Probabilistic Viewpoint. In contrast with the nominator, the denominator of all the extracted bounds can be computed during the synchronization, as is the case with the constant D_e. D_{e^p} is altered due to the movement of e^p but can be determined individually by sites as they share the common information about the posed threshold and e^p at any given timestamp. Obviously, the nominator changes as time passes and sites need to be continuously aware of the expected drifts of the candidate models so as to consult the corresponding bounds and dynamically choose the best monitoring model.

As a matter of fact, it is difficult to extract any generic decision-making mechanism regarding the best choice of the tracking mechanism, unless we are aware of the exact probability density function of $Pr_{Sync}^{Chosen\_Model}(t)$ at any given t. To simplify things, one may assume that, not only $Pr_{Sync}^{Chosen\_Model}$,

but also the quantities that bound this probability, remain relatively stable and compute all probabilities given past statistics that are transmitted to nodes during some synchronization.

Despite the fact that our above analysis may be interesting, it incorporates strong assumptions which tremendously limit its applicability to practical tracking scenarios (such as lack of dependency between local data streams and $Pr_{Sync}^{Chosen\_Model}$'s stability). Moreover, any global statistics that need to be continuously transmitted for computing the probabilities of Table III increase the overall bandwidth consumption, which is clearly not desirable. Because of the aforementioned limitations, in what follows we devise a more empirical approach, which selects the monitoring model to be adopted based on the recent performance of these models, while also not requiring any statistics to be transmitted by the coordinator to the nodes.

7.2. An Empirical Viewpoint

In order to provide an appropriate decision making mechanism, we require that sites keep up monitoring all the six options mentioned above. This monitoring will take place **only** for models that would not result in **any** local transmission since the last synchronization (i.e., we stop monitoring an alternative model for which we detect that a transmission would have been caused). Notice that one model has been chosen as the main model after the last synchronization. Thus, for each of the 6 alternatives, the sites maintain 6 bits, where the i-th bit is set iff the corresponding monitoring mode would have resulted in at least one transmission since the last synchronization. A synchronization can still be caused only by the main model. Upon a synchronization, however, together with $v_i(t)$ and the velocity vector in the case of the velocity acceleration model choice, sites attach 5 bits (they do not need to send a bit for the current model being used) on their messages. The 5 transmitted bits per node obviously constitute a very small overhead. Please note that the following facts hold in our adaptive algorithm:

- No site had a violation using the current model in a previous time instance (since the previous synchronization).
- An alternative model that has its corresponding bit to 1 in **any** of the sites would not have been better than the model currently being used, since it would have resulted in a transmission in a prior (or the current) time instance.
- Based on the above observation, we decide to switch to an alternative model only if the corresponding bits for this model were equal to 0 in **all** the sites.

In case of multiple alternatives with unset bits, a random choice among such alternatives is performed.

8. APPROXIMATE QUERY ANSWERING

8.1. Absolute Threshold Monitoring

In the previous sections we focused on tracking functions where the posed threshold T, and thus the threshold surface of the input domain, remained stable throughout the monitoring process. Nevertheless, in many cases [Cormode and Garofalakis 2005; 2007; 2008; Cormode et al. 2005; Cormode et al. 2007; Olston et al. 2003] it is particularly interesting to continuously maintain at the coordinator an approximation (i.e., given some accuracy constant $\varepsilon > 0$) of the true value of the monitored function. In terms of the concepts previously presented in our prediction-based monitoring framework, such a need can be expressed in the following tracking task:

$$|f(v(t)) - f(e^p(t))| \leq \varepsilon \tag{5}$$

In this scenario, the coordinator is capable of constantly providing at any given timestamp t $\varepsilon-$approximate answers ($f(e^p(t))$) regarding the value of the monitored function, utilizing the predicted estimate vector. Central data collection is caused only when the error in the answer may exceed the specified accuracy bound ε.

It is not difficult to see that Inequality 5 can be decomposed into two tracking tasks, namely $f(v(t)) \leq \varepsilon + f(e^p(t))$ and $f(v(t)) \geq f(e^p(t)) - \varepsilon$, which can be handled separately by the empirical CAA approach devised in the previous section. This shows that our techniques are general enough to accomondate any approximate function monitoring requirement. However, there exists a class of functions for which we are able to ensure that good predictors (those satisfying Property 1) can always yield reduced communication burden by applying a simple transformation of the monitoring task.

To achieve the desired transformation, we reside to the notion of Holder continuity. More precisely, a function $f : R^d \longrightarrow R$ is called Holder continuous on R^d if, for Holder constants $H_f, \alpha > 0$, the change in the output of f can be bounded by a corresponding change in the input. In our notation:

$$|f(e^p(t)) - f(v(t))| \leq H_f \cdot \|e^p(t) - v(t)\|^\alpha \tag{6}$$

For $\alpha = 1$ the function is called Lipschitz continuous. Examples of functions that can be easily verified that conform with Holder continuity include, but are not limited to, general L_p norms that abide by the reverse triangle inequality as well as trigonometric functions such as the cosine similarity. All these functions have been adopted within the geometric monitoring framework for detecting outliers in sensor network settings [Burdakis and Deligiannakis 2012].

Given Inequalities 5 and 6, we can actually come up with a value $\delta > 0$ such that:

$$\|e^p(t) - v(t)\| \leq \delta \Rightarrow H_f \cdot \|e^p(t) - v(t)\|^\alpha \leq H_f \cdot \delta^\alpha \leq \varepsilon \tag{7}$$

which forms the final version of our monitoring task, since $|f(e^p(t)) - f(v(t))| \leq H_f \cdot \|e^p(t) - v(t)\|^\alpha \leq H_f \cdot \delta^\alpha \leq \varepsilon$. Taking L_p norm monitoring [Burdakis and Deligiannakis 2012] as an example, the corresponding constants are set to $H_f = 1, a = 1$ and $\delta = \varepsilon$. Despite the fact that the above Holder condition is not always achievable for any tracked f on R^d, after a synchronization, we can define a neighborhood $U \subset R^d$ of the input domain for which Holder's condition holds and have sites call for a synchronization whenever the resulted vectors may exit this neighborhood (in addition to tracking the $\varepsilon-$approximation requirement). Hence, we expect that a wide variety of functions can be covered by the analysis that we present in this section.

Inequality 7 actually transforms the original monitoring target of Inequality 5 to a $\delta-$approximation on the values of the input domain $\|e^p(t) - v(t)\| \leq \delta$. In the remainder of this section we discuss how this special tracking task can be performed by two alternative approaches:

— By decomposing the problem into a set of simple local constraints (DLC algorithm, described in Section 8.1.1). Using this technique, each site will be able to monitor a simple local constraint that is based only on the deviation of its actual local measurements vector $v_i(t)$ from its corresponding predicted vector $v_i^p(t)$.
— By adopting the prediction-based monitoring framework (Section 8.1.2). While this technique shares some similarities with the DLC algorithm, it also allows the application of intersection monitoring techniques, thus enabling the use of our CAA algorithm.

8.1.1. Distributed Monitoring by Simple Decomposition to Local Constraints (DLC). In order to achieve the $\delta-$approximation target, we first decompose the problem of distributively ensuring that $\|e^p(t) - v(t)\| \leq \delta$ to simple local constraints that sites can monitor, in order to decide whether a synchronization process needs to take place. The upcoming lemma elaborates on the later issue, introducing an appropriate sufficient condition.

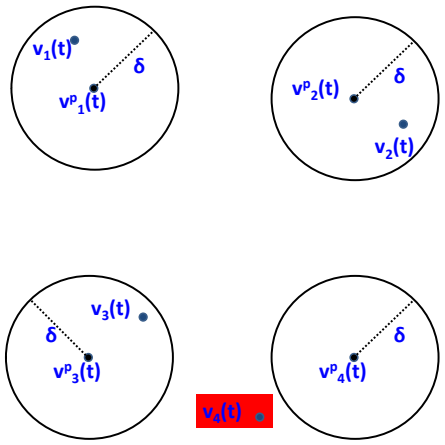
LEMMA 8.1. *When the local constraint*

$$\|v_i^p(t) - v_i(t)\| \leq \delta$$

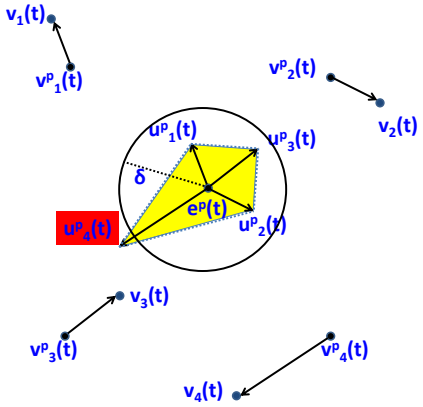
holds for any site $S_i, i \in \{1..n\}$ *at time t, then*

$$\|e^p(t) - v(t)\| \leq \delta$$

i.e., the estimation of the true global vector that is kept at the coordinating source, is always a $\delta-$*approximation of the true global vector.*



(a) Distributed δ−approximate monitoring by simple decomposition to local constraints for four sites. The local vector $v_4(t)$ of site S_4 exits the ball of radius δ, centered at $v_4^p(t)$.



(b) Application of Model 1 for δ−approximate tracking. The prediction deviation vector of site S_4 exits the ball centered at e^p with a radius of δ size.

Fig. 7. Comparison of approximate query answering approaches

PROOF. Starting by the local constraint of a single site S_i we obtain:

$$\|v_i^p(t) - v_i(t)\| \leq \delta \overset{w_i \geq 0}{\Leftrightarrow} \|w_i \cdot v_i^p(t) - w_i \cdot v_i(t)\| \leq w_i \cdot \delta$$

Summing for n sites, we have

$$\sum_{i=1}^{n} \|w_i \cdot v_i^p(t) - w_i \cdot v_i(t)\| \leq \sum_{i=1}^{n} w_i \cdot \delta \overset{\substack{triangle \\ inequality}}{\Rightarrow}$$

$$\|\sum_{i=1}^{n} w_i \cdot v_i^p(t) - \sum_{i=1}^{n} w_i \cdot v_i(t)\| \leq \sum_{i=1}^{n} w_i \cdot \delta \overset{\sum_{i=1}^{n} w_i > 0}{\Leftrightarrow}$$

$$\|\frac{\sum_{i=1}^{n} w_i \cdot v_i^p(t)}{\sum_{i=1}^{n} w_i} - \frac{\sum_{i=1}^{n} w_i \cdot v_i(t)}{\sum_{i=1}^{n} w_i}\| \leq \delta \Leftrightarrow$$

$$\|e^p(t) - v(t)\| \leq \delta$$

which concludes the proof. □

As long as Lemma 8.1 holds, no transmission is required. Each site keeps up acquiring updates and locally maintains the required information for the computation of predictors at the next synchronization, as described in Section 3.2. If $\|v_i^p(t) - v_i(t)\| \leq \delta$ is not satisfied in at least one site, then a synchronization occurs. Thus, the decomposition to local constraints essentially results in sites constructing balls centered at $v_i^p(t)$ with a radius of δ and monitoring whether $v_i(t)$ lies within this ball (see Figure 7(a)). Moreover, notice that the above tracking scheme does not require any information regarding e^p to be broadcasted to sites, as e^p is not involved in the local constraint that sites need to check.

8.1.2. Applying the Prediction-Based Geometric Approach. Our second option is that of fostering the prediction-based geometric monitoring framework. Before doing so, we need to identify the basic elements of the tracking process. We start by studying the function of interest

$q(v(t)) = \|e^p(t) - v(t)\| \leq \delta$ according to Inequality 7. Note that $e^p(t)$ is made known to all the sites after each synchronization process and it can thus be treated as a constant in the monitored function $q(v(t))$.

Again, let us assume that the coordinator has collected all the local vectors of monitored sites at a previous timestep. Then, the central source extracts e and e^p, which are broadcasted to the network. Having received the estimate as well as the predicted estimate vectors, every site S_i individually computes the drift vector $u_i(t)$ and the prediction deviation vector $u_i^p(t)$.

Nonetheless, it is worth focusing on the shape of the threshold surface in more detail. The region where the monitored convex hull $Conv\ (u_1^p(t), \ldots, u_n^p(t))$ is allowed to lie is again the ball $B_{e^p}^{\delta}$ (depicted in Figure 7(b)), centered at $e^p(t)$ with radius δ, which is convex. Due to the convexity of $B_{e^p}^{\delta}$, the convex hull of interest cannot cause threshold crossing without having at least one of its vertices $(u_1^p(t), \ldots, u_n^p(t))$ doing so. Hence, for Model 1 and Model 0 it suffices for every site to check if $u_i^p(t) = e^p + (v_i(t) - v_i^p(t))$ is included in $B_{e^p}^{\delta}$. Overall, we again check a spherical constraint of radius δ, which is violated only when $\|v_i(t) - v_i^p(t)\| > \delta$. Our previous discussion renders the presented approaches of Model 1, Model 0 and DLC equivalent in terms of the size of the consulted local constraints. Note that, as explained in Section 4, the size of the constructed constraints is an important factor in determining the communication cost of a given monitoring scheme.

However, we have also noted that the choice of the reference point is the other major criterion in tuning the (minimum) distance of the tracked convex hull (or intersection) from the threshold surface. Notice that the adoption of an alternative tracking mechanism (such as the Average or the Safer models) does not have an impact on the threshold surface. In fact, based on the estimation $e^p(t)$ on which the coordinator bases its $\varepsilon-$approximate query answers, the surface of $B_{e^p}^{\delta}$ forms the threshold surface independently of the used alternative tracking mechanism.

Given the above, the adoption of the Average model of Section 6.1 enforces sites to check whether $\frac{u_i^p+u_i}{2}$ exits $B_{e^p}^{\delta}$. As regards the Safer model, notice that since $B_{e^p}^{\delta}$ forms the threshold surface, it will always be reduced to the choice of Model 1, since e^p will always be the reference point that is farther from that surface. Eventually, the loosened intersection monitoring schemes of Section 6.2 can be applied by having sites check if the local intersection of the corresponding balls exceeds the surface of $B_{e^p}^{\delta}$. As a result, the intersection monitoring schemes together with the Average model are added tools which the DLC approach cannot exploit.

Hence, the empirical CAA approach of Section 7.2 is still applicable and capable of providing additional flexibility compared to the framework of Section 8.1.1 to the distributed tracking process. This is because CAA can tune both the size of the local constraints and their position, as well as choosing the preferable monitoring technique depending on the recent performance of the corresponding alternative tracking mechanisms.

8.2. Relative Threshold Monitoring

Another form of approximate function tracking which is often utilized in the context of data streams [Cormode and Garofalakis 2008; Cormode et al. 2005; Liu et al. 2012] is that of relative threshold monitoring. In particular, this case reflects to the following monitoring task (for $f(v(t)) \geq 0$ – otherwise, the displayed inequality is reversed):

$$(1-\varepsilon)f(v(t)) \leq f(e^p(t)) \leq (1+\varepsilon)f(v(t)) \tag{8}$$

In other words, we want to ensure that the approximate answer provided by the coordinator using $f(e^p(t))$ is within $\pm\varepsilon f(v(t))$ units from the current value of the tracked f. Nevertheless, since sites are not aware of $f(v(t)$, we need to modify our prior monitoring approach to render it capable of accomplishing relative threshold monitoring.

$$\begin{cases} \frac{f(e^p(t))}{1+\varepsilon} \leq f(v(t)) \\ \frac{f(e^p(t))}{1-\varepsilon} \geq f(v(t)) \end{cases} \Leftrightarrow \begin{cases} f(e^p(t)) - \frac{f(e^p(t))}{1+\varepsilon} \geq f(e^p(t)) - f(v(t)) \\ \frac{f(e^p(t))}{1-\varepsilon} - f(e^p(t)) \geq f(v(t)) - f(e^p(t)) \end{cases}$$

The above two conditions can both be ensured by checking whether:

$$|f(e^p(t)) - f(v(t))| \leq \min\{|f(e^p(t)) - \frac{f(e^p(t))}{1+\varepsilon}|, |\frac{f(e^p(t))}{1-\varepsilon} - f(e^p(t))|\} \leq \min\{\frac{\varepsilon f(e^p(t))}{1+\varepsilon}|, |\frac{\varepsilon f(e^p(t))}{1-\varepsilon}|\}$$

An equivalent result can be obtained for the case when $f(v(t)) \leq 0$. The latter inequality resembles Inequality 5, with the difference being that the right side is now dynamically adjusted based on the (known to the sites) current value of $f(e^p(t))$. Thus, the tracking task can be conducted utilizing the concepts presented in Sections 8.1.1 and 8.1.2.

8.3. Discussion

A question that naturally arises is whether the absolute or relative threshold monitoring tasks can also be performed in cases of functions that either are not Holder or Lipschitz continuous or their conformity with Holder continuity is hard to be verified. As previously noted, one option is to identify a neighborhood $U \subset R^d$ where local continuity for the tracked $f()$ exists. In addition to that option, in the following examples we study functions where our framework may be rendered applicable by exploiting properties of the adopted predictor (Section 8.3.1) or by properly transforming the monitored $f()$ (Section 8.3.2). We choose to elaborate on these particular functions due to their popularity in distributed data stream monitoring settings [Cormode and Garofalakis 2008; 2007; 2005] and within the geometric monitoring scheme [Burdakis and Deligiannakis 2012].

8.3.1. Exploiting Properties of our Predictor. We first describe a case in which the relative threshold monitoring task can be performed even if the function $f()$ under study is not globally Holder or Lipschitz continuous. To tackle this problem, we demonstrate that it may be possible to exploit the properties of the adopted predictors in order to achieve bounding the change in the output by a corresponding change in the input, as in Inequality 6.

Example 8.2. Assume that, as in [Cormode and Garofalakis 2008], we would like to perform relative threshold monitoring on the self join $\|v(t)\|^2$ function, which is not Lipschitz continuous function over all reals. Our local vectors are frequency distribution vectors [Cormode and Garofalakis 2008] i.e., streaming tuples apart from either +1 or -1 updates on d coordinates, and we choose to adopt the LG predictor (Table II). If the distributed system currently is at the $N-th$ epoch of its operation and N_s denotes the epoch when the last synchronization process took place, we have (we keep $e^p(t)$ in $\|v(t) - e^p(t)\|$ to show the correspondence with Inequality 5):

$$|f(v(t)) - f(e^p(t))| = \|f(v(t)) - f(e^p(t))\| = \|\|v(t)\|^2 - \|e^p(t)\|^2\| =$$

$$\|v(t) + e^p(t)\|\|v(t) - e^p(t)\| = \|v(t) + \frac{N}{N_s}e\|\|v(t) - e^p(t)\| \overset{\substack{triangle \\ inequality}}{\leq}$$

$$(\|v(t)\| + \|\frac{N}{N_s}e\|)\|v(t) - e^p(t)\| \leq (nN\sqrt{d} + \|\frac{N}{N_s}e\|)\|v(t) - e^p(t)\| \leq$$

$$(nN\sqrt{d} + \|\frac{N}{N_s}e\|)\delta$$

Thus, in order to enforce the relative threshold monitoring task, the sites need to continuously check whether:

$$(nN\sqrt{d}+\|\frac{N}{N_s}e\|)\delta \leq min\{\frac{\varepsilon f(e^p(t))}{1+\varepsilon},\frac{\varepsilon f(e^p(t))}{1-\varepsilon}\}.$$

This yields the desired value of δ as: $\delta = \frac{min\{\frac{\varepsilon f(e^p(t))}{1+\varepsilon},\frac{\varepsilon f(e^p(t))}{1-\varepsilon}\}}{(nN\sqrt{d}+\|\frac{N}{N_s}e\|)}$, which can be individually computed by sites at any given epoch, as long as they share the common information about e, e^p and ε, the dimensionality d of $v_i(t)$, the size of the network n and the values of N_s, N. □

8.3.2. Exploiting Function Transformations. We now provide an example of absolute threshold monitoring tasks that can be performed even if it is not straightforward whether the monitoring functions are Lipschitz or Holder continuous. The basic idea is to properly transform the monitoring function to other functions that are well known that abide by that type of continuity.

Example 8.3. Suppose we have installed a number of sensors in a server room so as to continuously measure the environmental conditions (such as temperature, radiance, noise levels and so on) under which the computing machines operate. In order to assure unhindered operation, we let motes collect a number of w recent observations based on which we intend to monitor the resemblance the global vector $v(t)$ exhibits with a given vector g representing the "normal" values of the quantities sampled by the sensor nodes. The resemblance between the $g, v(t)$ pair of vectors can be determined based on the Extended Jacccard (or Tanimoto) Coefficient (Tan) [Burdakis and Deligiannakis 2012; Giatrakos et al. 8005]:

In order to perform the desired transformation of the Tanimoto Coefficient:

$$Tan(g,v(t)) = \frac{g \cdot v(t)}{\|g\|^2 + \|v(t)\|^2 - g \cdot v(t)}$$

the initial requirement of:

$$|Tan(g,e^p(t)) - Tan(g,v(t))| \leq \varepsilon$$

is first decomposed in two constraints:

$$\begin{cases} T_1(t) = Tan(g,e^p(t)) - \varepsilon \ \leq \ Tan(g,v(t)) \\ T_2(t) = Tan(g,e^p(t)) + \varepsilon \ \geq \ Tan(g,v(t)) \end{cases}$$

and then transformed as follows [Giatrakos et al. 8005] (for $T_1(t), T_2(t) \in [0,1]$):

$$\begin{cases} \|\frac{T_1(t)+1}{2T_1(t)}g - v(t)\| \ \leq \ \frac{\sqrt{-4T_1(t)^2+(T_1(t)+1)^2}}{2T_1(t)}\|g\| \\ \|\frac{T_2(t)+1}{2T_2(t)}g - v(t)\| \ \geq \ \frac{\sqrt{-4T_2(t)^2+(T_2(t)+1)^2}}{2T_2(t)}\|g\| \end{cases}$$

Notice that since g, $e^p(t)$), ε have been broadcasted, $T_1(t)$ and $T_2(t)$ can be individually computed by sites. Now then, subtracting the previous inequalities we get

$$\|\frac{T_1(t)+1}{2T_1(t)}g - v(t)\| - \|\frac{T_2(t)+1}{2T_2(t)}g - v(t)\| \leq \frac{\sqrt{-4T_1(t)^2+(T_1(t)+1)^2}}{2T_1(t)}\|g\| - \frac{\sqrt{-4T_2(t)^2+(T_2(t)+1)^2}}{2T_2(t)}\|g\|$$

which can be ensured if

$$\|\frac{T_1(t)+1}{2T_1(t)}g - v(t)\| - \|\frac{T_2(t)+1}{2T_2(t)}g - v(t)\|| \leq |\frac{\sqrt{-4T_1(t)^2+(T_1(t)+1)^2}}{2T_1(t)} - \frac{\sqrt{-4T_2(t)^2+(T_2(t)+1)^2}}{2T_2(t)}|\|g\|$$

The left side of the above can be bounded as:

$$
\begin{aligned}
|\|\tfrac{T_1(t)+1}{2T_1(t)}g - v(t)\| - \|\tfrac{T_2(t)+1}{2T_2(t)}g - v(t)\|| &\overset{\substack{reverse\\ triangle\\ inequality}}{\leq} \\
\|(\tfrac{T_1(t)+1}{2T_1(t)} + \tfrac{T_2(t)+1}{2T_2(t)})g - 2v(t)\| &= \\
\|(\tfrac{T_1(t)+1}{2T_1(t)} + \tfrac{T_2(t)+1}{2T_2(t)})g - 2e^p(t) - 2(v(t) - e^p(t))\| &\overset{\substack{triangle\\ inequality}}{\leq} \\
\|(\tfrac{T_1(t)+1}{2T_1(t)} + \tfrac{T_2(t)+1}{2T_2(t)})g - 2e^p(t)\| + 2\|v(t) - e^p(t)\| &\leq \\
\|(\tfrac{T_1(t)+1}{2T_1(t)} + \tfrac{T_2(t)+1}{2T_2(t)})g - 2e^p(t)\| + 2\delta &\leq \\
|\tfrac{\sqrt{-4T_1(t)^2+(T_1(t)+1)^2}}{2T_1(t)} - \tfrac{\sqrt{-4T_2(t)^2+(T_2(t)+1)^2}}{2T_2(t)}|\|g\| &
\end{aligned}
$$

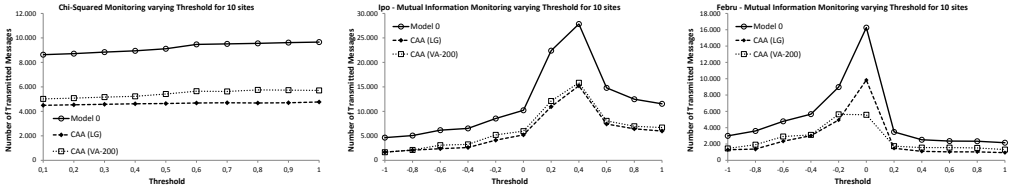
Obviously, $\delta = |\frac{\sqrt{-4T_1(t)^2+(T_1(t)+1)^2}}{2T_1(t)} - \frac{\sqrt{-4T_2(t)^2+(T_2(t)+1)^2}}{2T_2(t)}|\|g\| - \|(\frac{T_1(t)+1}{2T_1(t)} + \frac{T_2(t)+1}{2T_2(t)})g - 2e^p(t)\|$ and $H_f = 2$, $\alpha = 1$. □

9. EVALUATION RESULTS

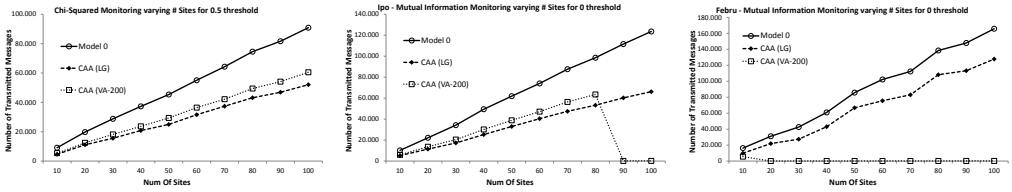
In order to evaluate our algorithms we developed a simulation environment in Java. We utilized two real data sets to derive data stream tuples arriving at every site in the network. "Corpus", consists of 804,414 records present in the Reuters Corpus (RCV1-v2) [Lewis et al. 2004] collection. Each record corresponds to a news story to which a list of terms (features) and appropriate categorization have been attributed. As in [Sharfman et al. 2008; 2007b] we focused on the following features: *Bosnia, Ipo, Febru* while monitoring their coexistence with the *CCAT* (the CORPORATE / INDUSTRIAL) category. Aiming at identifying the relevance of these features to the CCAT category at any given time, we monitored two different functions involving the Chi-Square(χ^2) and Mutual Information(*MI*) score. We utilized the Corpus data set in order to test our techniques using the Cash Register streaming paradigm i.e. taking into consideration the whole history of the tuples arriving at the various sites. In any given timestamp, after the receipt of a new tuple each site forms a vector which consists of four dimensions for the χ^2 and three dimensions for the *MI* case. These vectors have one of their positions set, while the rest remain zero. In particular, for both the functions the first position of the vector is set if the term and the category co-occur, the second if the term occurs without the CCAT category, the third in case CCAT is present without the term, while the fourth (only for χ^2 score) if neither of them appeared in the incoming tuple.

Due to the nature of the incoming (binary) vectors and the utilized Cash Register paradigm the previously described environment may be considered moderate to change and be thought of as easily predictable by our techniques. In order to test our algorithms in more dynamic conditions we utilized one more data set. The "Weather" data set includes Solar Irradiance, Wind Speed, Wind Peak and Temperature measurements from the station in the University of Washington and for the year 2002 [Deligiannakis et al. 2004], where each file incorporates 523,439 records of measurements. We used the Weather data sets so as to monitor the Variance (Var) and the Signal to Noise Ratio (StN) functions. We also tested our query answering techniques (Section 8) while performing approximate L_2 and cosine (*Cos*) similarity monitoring. We utilized Var, L_2 and *Cos* which have already been used within the geometric monitoring framework [Sharfman et al. 2007a; Burdakis and Deligiannakis 2012]. In addition, the StN function equals the ratio between the mean and the standard deviation ($\frac{\mu}{\sigma}$) in a given window of measurements and can, thus, be applicable to globally quantify the noise present in the measurements.

In each experiment we first measure the number of messages transmitted in the network across different thresholds for a network configuration consisting of 10 sites. We then use the middle case threshold and plot the number of transmitted messages when increasing the scale of the distributed environment (the number of sites). We denote the performance of the original bounding algorithm (Section 3.1) by "Model 0", while "Model 1" refers to the mere application and monitoring of the

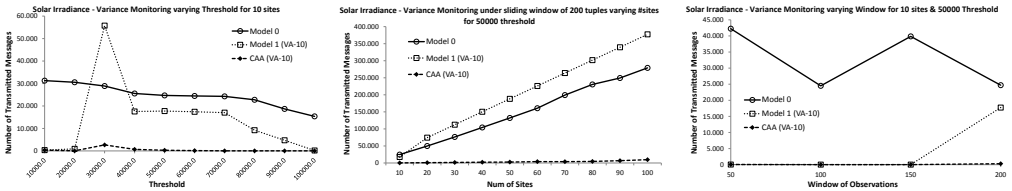


(a) Cash Register Varying Threshold

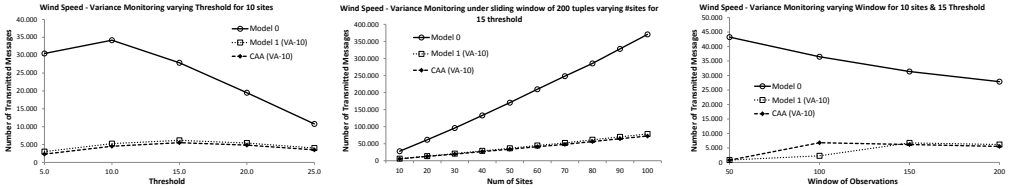


(b) Cash Register Varying Sites

Fig. 8. Corpus Data Set: Performance of our Techniques in the Cash Register Streaming Paradigm



(a) Solar Irradiance, Variance Monitoring



(b) Wind Speed, Variance Monitoring

Fig. 9. Weather Data Set: Performance of our Techniques in the Sliding Window Streaming Paradigm for Variance Monitoring

prediction-based bounding algorithm (Section 4). Eventually, "CAA" shows the performance of the Choosing Amongst Alternatives framework that was introduced in Section 7.2. Moreover, for each of the lines in the graphs, we enclosed the chosen predictor using *LG* to denote the Linear Growth predictor and $VA-W$ so as to declare a Velocity / Acceleration predictor with a window of W measurements [4].

[4]We focus on comparing the performance of our prediction-based geometric monitoring techniques against [Sharfman et al. 2006; 2007b] (Model 0), since we expect our prediction-based methods to give similar benefits when operating over the ellipsoidal bounding regions of [Sharfman et al. 2008] to those seen in our current study using spherical constraints as in [Sharfman et al. 2006; 2007b].

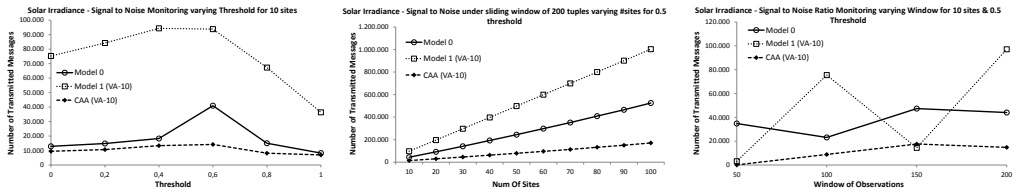
9.1. Corpus Data Set - Cash Register Paradigm

We begin our study by examining the performance of our techniques in the Corpus data set on par with the Cash Register paradigm adoption. Figure 8 depicts the performance of Model 0 and of the CAA approach when using the *LG* and the *VA* predictors. Since almost 6000 documents are received within a period of a month [Sharfman et al. 2008], we choose a $W = 200$ window for the VA predictor which is expected to be roughly the amount of news stories received daily. Each column of the figure corresponds to the case of the terms "Bosnia", "Ipo" and "Febru", respectively.
Sensitivity to Threshold - Chi Square. As shown in the first column of Figure 8, where the χ^2 function for the term "Bosnia" is monitored, Model 0 appears to always be about 2 and 1.85 times worse in terms of the number of transmitted messages when compared to the CAA(LG) and CAA(VA-200) approaches, respectively, for different threshold values (Fig. 8(a)) using 10 sites.
Sensitivity to Threshold - Mutual Information (MI). Moving to the second and third columns of Figure 8(a) we investigate the cases of the "Ipo" as well as "Febru" terms, monitoring the *MI* function across different threshold values for a 10 site configuration (note that *MI* is calculated as a logarithm, therefore the negative threshold values in that axis). In these graphs the peak that occurs at 0.4 and 0 for the "Ipo" and "Febru" involves an accumulation of synchronizations around the average value the *MI* function possesses along the run. We again observe that CAA(LG) performs 1.75-2.1 times better than the Model 0 case for both monitored terms. Despite the fact that CAA(VA-200) is proved slightly worse compared to CAA(LG), it is still able to better amend the peak that occurs in "Febru" monitoring for a 0 threshold.
Sensitivity to Number of Sites. Eventually, switching to Figure 8(b), for the "Bosnia" term (left column) the relative benefits remain almost the same across all network scales. For the "Ipo" term monitoring (middle column) we observe that Model 0 is steadily more than 1.8 times worse than the CAA(LG) choice across different scales. CAA(VA-200) performs worse than the CAA(LG) case for network configurations up to 80 sites. Nonetheless, the introduction of additional sites (along with their respective substreams) in 90, 100 site cases, causes the *MI* function to always lie below the posed zero threshold since the "Ipo" term becomes more rare. This fact is perfectly read by the CAA(VA-200) approach, the transmitted messages of which approach zero. A similar behavior appears early in the third column of Figure 8(b) for the Febru case where the introduction of more than 10 sites causes *MI* to be negative, which is again accurately pinpointed by the the CAA(VA-200) monitoring model reaching savings of 3 orders of magnitude size.

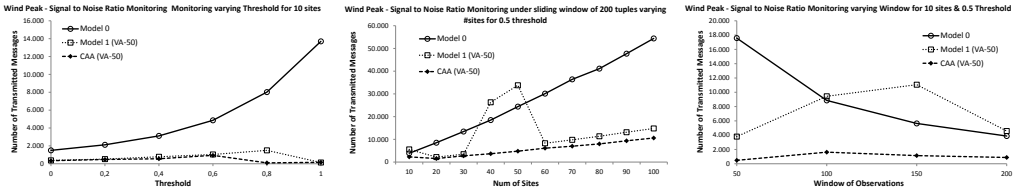
In the previously presented graphs we omitted the lines for the mere application of Model 1 to keep the diagram readable, since Model 1 shows almost identical (actually CAA can occasionally save a few tens of extra messages) behavior with its corresponding CAA applications. CAA possesses the ability to recognize the utility of Model 1 (the mere application of predictors as described in Section 4) in this setting and encompass it throughout its operation. On the other hand, this fact exhibits the ability of Model 1 to provide an efficient solution in environments where v_i values evolve relatively slowly. Nonetheless, as we will shortly present, this is not always true in scenarios where more dynamic updates occur.

9.2. Weather Data - Sliding Window Paradigm

We proceed to the sliding window operation, using the Weather data set and monitoring the Var and StN functions. Please note that in all the graphs presented in the current subsection the Linear Growth predictor is not applicable since it assumes that local vectors (v_i) uniformly evolve by a time dependent factor (Section 3.2), which is obviously unrealistic for the physical measurements in the Weather data and the sliding window application scenario. We thus compare the performance of Model 0, Model 1(VA-W) and CAA(VA-W) cases in our study. For the CAA(VA-W) monitoring model we again choose the window based on natural time units' division. We uniformly utilize a prediction window $W = 10$ which corresponds to the latest minutes of received observations, except for the Wind Peak data where $W = 50$ was chosen to adjust predictions to the expected frequency of the peaks in the wind blasts. For each data set-function pair, the default value of the sliding window



(a) Solar Irradiance, StN Monitoring



(b) Wind Peak, StN Monitoring

Fig. 10. Weather Data Set: Performance of our Techniques in the Sliding Window Streaming Paradigm for StN Monitoring

size, over which the corresponding function is computed, is 200 measurements. However, we also perform a sensitivity analysis on this parameter as well.

Variance Monitoring. Figure 9(a) plots the performance of the techniques in the case of Var monitoring in the Solar Irradiance Data. In the first column of the figure we observe that the cost of Model 0 ranges between 11 and 600 times larger than the cost yielded by CAA(VA-10) monitoring model, while CAA(VA-10) ensures up to 500 times lower cost even when compared with Model 1 across different thresholds. A case of particular interest shows up for a threshold of 30000. There, Model 1 shows a peak in the number of transmitted messages which are higher even when compared to Model 0. This happens due to the existence of specific sites whose drift vectors approach the threshold surface as noted in Figure 3. Obviously, increasing the threshold to 40000 alters the threshold surface and thus hinders the same sites to cause threshold violations. Nonetheless, CAA(VA-10) maintains low transmission cost due to the loosened intersection monitoring capabilities (Section 6.2) that it embodies. We will revisit this issue in the next subsection were we look into the operational details of the CAA monitoring model. In the meantime we note that the same applies for the second column of Figure 9(a) where increasing the scale of the network results in CAA(VA-10) savings that reach a number of 30 times compared to Model 0 and they become even larger when compared to Model 1. Eventually, the third column of the same figure, shows the resilience of our techniques when altering the employed size of observations encapsulated in the sliding window for 10 sites. CAA(VA-10) shows similar behavior when enlarging the window. Model 1 yields more synchronizations for a window of 200 observations since enlarging the window causes the variance values within it to increase and thus some sites approach the posed threshold of 50000. The lack of the alternative mechanisms that are incorporated in CAA leads sites merely utilizing Model 1 to threshold crossings. Finally, Model 0 exhibits high sensitivity to the number of values that local vectors (v_i) are built upon.

Figure 9(b) presents corresponding results for Var monitoring in the Wind Speed data set. Model 1 is slightly worse (almost 1%) in terms of transmitted messages compared to CAA(VA-10) when varying the threshold (first column in the figure) and across different network scales (second column), yet both result in savings ranging between 3 and 13 times compared to the message cost of Model 0. Furthermore, in the third column of Figure 9(b) we observe that both CAA(VA-10) and Model 1 remain resilient to altering the sliding window size ensuring significant benefits when compared to Model 0. Notice that for a window of 100 observations, Model 1 performs better than CAA(VA-10). Recall that CAA resolves ties in the choice of the monitoring mechanism (Section 7)

by picking a random model among those which did not cause a threshold crossing. Thus, when a particular model is always the appropriate choice, the adaptive CAA algorithm may sometimes end up transmitting slightly more messages. The results are similar for the Wind Peak data set which we omit.

Signal to Noise Monitoring. In our next experiment we utilized the same motif for analyzing the performance of our techniques in monitoring the StN function. We begin our discussion with the Solar Irradiance data set in Figure 10(a). CAA(VA-10) performs up to 3 times better than Model 0 when varying the threshold for 10 sites (first column in the figure) and up to 5 times across network configurations of 10-100 sites (second column). Model 1 is again the worst choice as it yields 2-5 times higher cost compared to Model 0 across different thresholds and appears over 2 times worse than Model 0 for different scales. In the third column of Figure 10(a), it is evident that CAA(VA-10) again remains mostly unaffected to different window sizes, while Model 1 exhibits a wavy behavior depending on the accuracy of the employed VA-10 predictor.

We then analyze the performance of the Wind Peak Data in StN monitoring (Figure 10(b)) (the Wind Speed data had similar behavior). Model 1 and CAA possess similar performance across different thresholds with savings ranging between 4 and 85 times compared to the cost of Model 0. The same holds in large part when varying the network scale (middle column in Fig 10(b)) where savings reach a factor of 5. An exception occurs for 40 and 50 site configuration cases. This is another occasion where site predictors lie close to the threshold surface for the given threshold of 0.5 and CAA manages to achieve increased savings due to the intersection monitoring capacity. As more sites are added in the subsequent steps (60-100 site configurations) the predicted estimate's (e^p) position is affected and thus the sites that were previously causing synchronizations (despite their restricted local constraints - balls) were ousted from the threshold surface, stabilizing the cost of Model 1. Eventually, as with the previously examined functions-data set pairs, the CAA monitoring model is not sensitive to changes in the window size (third column of Fig 10(b)) while Model 0 and Model 1 exhibit opposite trends upon enlarging it. Overall, Model 0's cost is 5 to 35 times the transmission cost of CAA, while savings against Model 1 range between 4 to 9 times across different window sizes.

9.3. CAA Operational Insights

We are now providing additional details regarding the choices that CAA makes throughout its operation to investigate the stem of its benefits. Since it is hard to present analytic statistics of alternative models' usage for every single case of the previously discussed graphs, we focus on two situations where Model 1 exhibits possibly unexpected peaks in the number of messages and examine the tools that CAA utilizes to avoid similarly high message exchange.

The first of the aforementioned cases regards the Solar Irradiance under Var monitoring against different thresholds and for 10 sites(left figure of Fig 9(a)). Table IV shows the CAA choices for different thresholds. Intersection1 refers to monitoring the intersection between the original and the predicted convex hull, while Intersection2 refers to monitoring the intersection between the average convex hull and the predicted one. We point out that for threshold ¿10000 (where it exhibits low costs) Model 1 is never employed by CAA. For the threshold 30000 case, Model 0 appears as the most frequent choice but it is only used during the first synchronizations until predictors are stabilized around the threshold surface (if Model 0 was continuously picked, CAA would have had similar cost to Model 0). Afterwards, the loosened intersection framework is chosen which safely leads the monitoring procedure to the decrement of the transmission cost as shown in Fig 9(a).

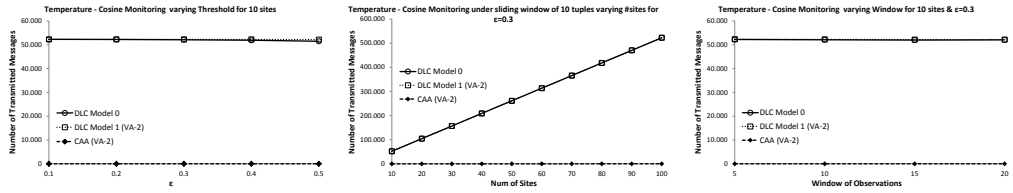
The second case we distinguished during our discussion was the peak that occurs when monitoring the Wind Peak data under the StN function for network configurations of different scale (middle figure of Fig. 10(b)). As Table V shows, for 10 sites the savings CAA provides are mostly attributed to the average and safer model usage, while for 40, 50 and more sites, after a few synchronizations, the single time that Intersection2 is employed by CAA hinders communication for a considerable amount of time.

Table IV. Case study: Solar-Var Vs Threshold Monitoring

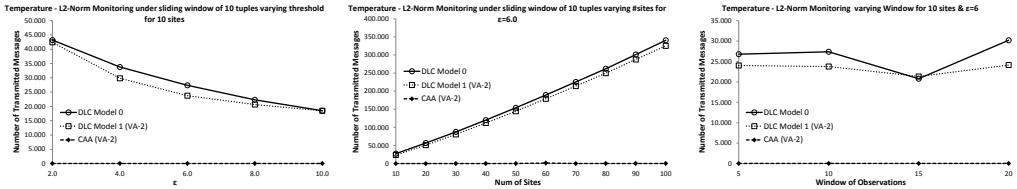
Threshold	Model 0	Model 1	Average	Safer	Intersection 1	Intersection 2
10000	0	15	0	3	0	0
30000	105	0	0	0	4	25
50000	7	0	0	0	5	3
70000	0	0	1	0	0	1
90000	0	0	1	0	0	1

Table V. Case study: Wind Peak-StN Vs # Sites Monitoring

# Sites	Model 0	Model 1	Average	Safer	Intersection 1	Intersection 2
10	35	16	25	35	1	3
40	8	6	12	19	0	1
50	13	10	7	17	0	1
80	12	14	9	14	0	1
90	9	9	12	21	0	1



(a) Temperature, Approximate Cos Monitoring



(b) Temperature, Approximate L_2 Monitoring

Fig. 11. Weather Data Set - Temperature: Approximate Query Answering Performance

9.4. Approximate Query Processing

Finally, in Figure 11 we present an experimental analysis regarding our prediction-based query answering procedures, introduced in Section 8, utilizing temperature measurements in the Weather data set. As in [Burdakis and Deligiannakis 2012], we perform L_2 norm as well as cosine (*Cos*) monitoring and measure the number of transmitted messages for different ε values (please recall that we enforce $|f(e^p(t)) - f(v(t))| \leq \varepsilon$), network scale configurations and sliding window sizes. We use the two most recent readings in the VA predictor i.e., $VA-2$. The adoption of the simple Decomposition to Local Constraints (DLC) approach provides no communication reduction in the case of *Cos* monitoring while allowing relatively small savings which range between 1% to 12% compared to the DLC Model 0 in L_2 approximate tracking. This happens due to the tight approximation requirements i.e., the magnitude of the ε value that are placed while performing approximate *Cos* answers. The latter fact shows that the mere adoption of good predictors does not achieve significant message reduction when the δ parameter which determines the radius of the constructed hypersphere (see Inequality 7 and Fig. 7) receives small values. However, it is important to note that the number of transmitted messages by the DLC approach is still always half the number of messages that the mere adoption of Model 1 and Model 0 (omitted in Figure 11) would require, respectively. This happens because, as noted in Section 8.1.1, the DLC framework does not require

any information to be broadcasted back to the sites. Hence, one way communication suffices during each synchronization process which in turn halves the number of transmitted messages.

On the contrary, the application of CAA on the techniques of Section 8.1.2 (diamonded line approaching the horizontal axis in Figure 11) is capable of providing communication savings from 400 times to 3 orders of magnitude compared to DLC Model 1 which are even higher when compared to DLC Model 0 for both of the monitored functions. Hence, we have validated the claim of Section 8.1.2 regarding the flexibility of the CAA approach to simultaneously tune both the size and the position of the local constraints so as to improve approximate query answering performance.

10. CONCLUSIONS

In this paper, we presented a thorough study regarding prediction models' adoption within the geometric monitoring setting. After identifying the peculiarities exhibited by predictors upon their implementation in the aforementioned environment, we developed a solid theoretic framework composed of sufficient conditions rendering predictors capable of refraining the communication burden. We proposed algorithms incorporating those conditions and expanded on relaxed versions of them along with extensive theoretical analysis on their expected benefits. Our ongoing efforts in this area explore the choice of optimal reference points, as in [Sharfman et al. 2008], that could perhaps enable "looser" conditions of strict containment.

REFERENCES

Anonymous. Paper-Info not Present for Double-Blind Reviewing Purposes.

B. Babcock and C. Olston. 2003. Distributed top-k monitoring. In *SIGMOD*.

S. Burdakis and A. Deligiannakis. 2012. Detecting Outliers in Sensor Networks Using the Geometric Approach. In *ICDE*.

G. Cormode and M. Garofalakis. 2005. Sketching streams through the net: Distributed approximate query tracking. In *VLDB*.

G. Cormode and M. Garofalakis. 2007. Streaming in a connected world: querying and tracking distributed data streams. In *SIGMOD*.

G. Cormode and M. Garofalakis. 2008. Approximate continuous querying over distributed streams. *ACM Transactions on Database Systems* 33, 2 (2008).

G. Cormode, M. Garofalakis, S. Muthukrishnan, and R. Rastogi. 2005. Holistic aggregates in a networked world: distributed tracking of approximate quantiles. In *SIGMOD*.

G. Cormode, S. Muthukrishnan, and K. Yi. 2011. Algorithms for distributed functional monitoring. *ACM Trans. Algorithms* 7 (2011), 21:1–21:20.

G. Cormode, S. Muthukrishnan, and W. Zhuang. 2007. Conquering the Divide: Continuous Clustering of Distributed Data Streams. In *ICDE*.

A. Das, S. Ganguly, M. Garofalakis, and R. Rastogi. 2004. Distributed set-expression cardinality estimation. In *VLDB*.

A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. 2004. Compressing Historical Information in Sensor Networks. In *ACM SIGMOD*.

Minos Garofalakis, Johannes Gehrke, and Rajeev Rastogi. 2002. Querying and mining data streams: you only get one look a tutorial. In *SIGMOD*.

N. Giatrakos, Y. Kotidis, A. Deligiannakis, V. Vassalos, and Y. Theodoridis. 2011. http://dx.doi.org/10.1016/j.is.2011.08.005. In-Network Approximate Computation of Outliers with Quality Guarantees. *Information Systems* (2011. http://dx.doi.org/10.1016/j.is.2011.08.005).

L. Huang, M. Garofalakis, J. Hellerstein, A. Joseph, and N. Taft. 2006. Toward sophisticated detection with distributed triggers. In *MineNet*.

L. Huang, X. Nguyen, M. Garofalakis, and J. M. Hellerstein. 2007. Communication-efficient online detection of network-wide anomalies. In *INFOCOM*.

A. Jain, J. M. Hellestein, S. Ratnasamy, and D. Wetherall. 2004. A wakeup call for internet monitoring systems: The case for distributed triggers. In *HotNets*.

R. Keralapura, G. Cormode, and J. Ramamirtham. 2006. Communication-efficient distributed monitoring of thresholded counts. In *SIGMOD*.

D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. 2004. RCV1: A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research* 5, Apr (2004), 361–397.

Zhenming Liu, Bozidar Radunović, and Milan Vojnović. 2012. Continuous distributed counting for non-monotonic streams. In *PODS*.

S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. 2005. TinyDB: an acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.* 30 (March 2005), 122–173.

C. Olston, J. Jiang, and J. Widom. 2003. Adaptive filters for continuous queries over distributed data streams. In *SIGMOD*.

G. Sagy, D. Keren, I. Sharfman, and A. Schuster. 2010. Distributed threshold querying of general functions by a difference of monotonic representation. *Proc. VLDB Endow.* 4 (2010), 46–57.

I. Sharfman, A. Schuster, and D. Keren. 2006. A Geometric Approach to Monitoring Threshold Functions over Distributed Data Streams. In *SIGMOD*.

I. Sharfman, A. Schuster, and D. Keren. 2007a. Aggregate Threshold Queries in Sensor Networks. In *IPDPS*.

I. Sharfman, A. Schuster, and D. Keren. 2007b. A Geometric Approach to Monitoring Threshold Functions over Distributed Data Streams. *ACM Transactions on Database Systems* 32, 4 (2007).

Izchak Sharfman, Assaf Schuster, and Daniel Keren. 2008. Shape sensitive geometric monitoring. In *PODS*.

K. Yi and Q. Zhang. 2009. Optimal tracking of distributed heavy hitters and quantiles. In *PODS*.

Q. Zhang, J. Liu, and W. Wang. 2008. Approximate Clustering on Distributed Data Streams. In *ICDE*.

Sketch-based Geometric Monitoring of Distributed Stream Queries

Minos Garofalakis
Technical University of Crete
minos@softnet.tuc.gr

Daniel Keren
University of Haifa
dkeren@cs.haifa.ac.il

Vasilis Samoladas
Technical University of Crete
vsam@softnet.tuc.gr

ABSTRACT

Emerging large-scale monitoring applications rely on continuous tracking of complex data-analysis queries over collections of massive, physically-distributed data streams. Thus, in addition to the space- and time-efficiency requirements of conventional stream processing (at each remote monitor site), effective solutions also need to guarantee communication efficiency (over the underlying communication network). The complexity of the monitored query adds to the difficulty of the problem — this is especially true for non-linear queries (e.g., joins), where no obvious solutions exist for distributing the monitor condition across sites. The recently proposed geometric method offers a generic methodology for splitting an arbitrary (non-linear) global threshold-monitoring task into a collection of local site constraints; still, the approach relies on maintaining the complete stream(s) at each site, thus raising serious efficiency concerns for massive data streams. In this paper, we propose novel algorithms for efficiently tracking a broad class of complex aggregate queries in such distributed-streams settings. Our tracking schemes rely on a novel combination of the geometric method with compact sketch summaries of local data streams, and maintain approximate answers with provable error guarantees, while optimizing space and processing costs at each remote site and communication cost across the network. One of our key technical insights for the effective use of the geometric method lies in exploiting a much lower-dimensional space for monitoring the sketch-based estimation query. Due to the complex, highly non-linear nature of these estimates, efficiently monitoring the local geometric constraints poses challenging algorithmic issues for which we propose novel solutions. Experimental results on real-life data streams verify the effectiveness of our approach.

1. INTRODUCTION

Traditional data-management systems are typically built on a *pull-based paradigm*, where users issue one-shot queries to static data sets residing on disk, and the system processes these queries and returns their results. Recent years, however, have witnessed the emergence of a new class of large-scale event monitoring applications, that require the ability to efficiently process continuous, high-volume *streams* of data in real time. Examples include monitoring systems for IP and sensor networks, real-time analysis tools for financial data streams, and event and operations monitoring applications for enterprise clouds and data centers. As both the scale of today's networked systems, and the volumes and rates of the associated data streams continue to increase with no bound in sight, algorithms and tools for effectively analyzing them are becoming an important research mandate.

Large-scale stream processing applications rely on *continuous*, event-driven monitoring, that is, real-time tracking of measurements and events, rather than one-shot answers to sporadic queries. Furthermore, the vast majority of these applications are inherently *distributed*, with several remote monitor sites observing their local, high-speed data streams and exchanging information through a communication network. This distribution of the data naturally implies critical *communication constraints* that typically prohibit centralizing all the streaming data, due to either the huge volume of the data (e.g., in IP-network monitoring, where the massive amounts of collected utilization and traffic information can overwhelm the production IP network [12]), or power and bandwidth restrictions (e.g., in wireless sensornets, where communication is the key determinant of sensor battery life [26]). Finally, an important requirement of large-scale event monitoring is the effective support for tracking complex, *holistic queries* that provide a global view of the data by combining and correlating information across the collection of remote monitor sites. For instance, tracking aggregates over the result of a distributed join (the "workhorse" operator for correlating data from different tables in relational databases) can provide unique, real-time insights into the workings of a large-scale distributed system, including system-wide correlations and potential anomalies [7]. Monitoring the precise value of such holistic queries without continuously centralizing all the data seems hopeless; luckily, when tracking statistical behavior and patters in large scale systems, *approximate answers* (with reasonable approximation error guarantees) are typically sufficient. This often allows algorithms to effectively tradeoff efficiency with approximation quality (e.g., using sketch-based stream approximations [7]).

Given the prohibitive cost of data centralization, it is clear that realizing sophisticated, large-scale distributed data-stream analysis tools must rely on novel algorithmic paradigms for processing local streams of data *in situ* (i.e., locally at the sites where the data is observed). This, of course, implies the need for intelligently decomposing a (possibly complex) global data-analysis and monitoring query into a collection of "safe" local queries that can be tracked independently at each site (without communication), while guaranteeing correctness for the global monitoring operation. This decomposition process can enable truly distributed, event-driven processing of real-time streaming data, using a *push-*

Proceedings of the VLDB Endowment, Vol. 6, No. 10

based paradigm, where sites monitor their local queries and communicate only when some local query constraints are violated [7, 31]. Nevertheless, effectively decomposing a complex, holistic query over the global collections of streams into such local constraints is far from straightforward, especially in the case of *non-linear* queries (e.g., joins) [31].

Prior Work. The bulk of work on data-stream processing has focused on developing space-efficient, one-pass algorithms for performing a wide range of *centralized, one-shot computations* on massive data streams; examples include computing quantiles [21], estimating distinct values [18] and set-expression cardinalities [16], counting frequent elements (i.e., "heavy hitters") [4, 10, 28], approximating large Haar-wavelet coefficients [20], and estimating join sizes and stream norms [1, 2, 15]. As already mentioned, all the above methods work in a centralized, one-shot setting and, therefore, do not consider communication-efficiency issues. Other work has proposed methods that carefully optimize site communication costs for approximating different queries in a distributed setting, including quantiles [22] and heavy hitters [27]; however, the underlying assumption is that the computation is triggered either periodically or in response to a one-shot request. Such techniques are not immediately applicable for *continuous-monitoring*, where the goal is to continuously provide real-time, guaranteed-quality estimates over a distributed collection of streams. Morphing such one-shot solutions to continuous problems entails propagating each change and recomputing the solutions which is communication inefficient, or involves periodic updates and other heuristics that can no longer provide real-time estimation guarantees.

Monitoring distributed data streams has attracted substantial research interest in recent years [6, 29]. Early work has looked at the monitoring of *single values*, and building appropriate models and filters to avoid propagating updates if these are insignificant compared to the value of a simple aggregate (e.g., to the SUM of the distributed values). [30] proposes a scheme based on "adaptive filters" — that is, bounds around the value of distributed variables, which shrink or grow in response to relative stability or variability, while ensuring that the total uncertainty in the bounds is at most a user-specified bound. [23] proposes building a Kalman Filter for individual values, and only propagating an update in a value if it falls more than δ away from the predicted value. The BBQ system [14] builds a dynamic, multi-dimensional probabilistic model of a set of distributed sensor readings to drive acquisitional query processing; this was later extended to the continuous case in the Ken system [5]. A common aspect of all these earlier works is that they typically consider only a small number of monitored values per site, and assume that it is feasible to locally monitor and/or build a model for each such value. In contrast, our problem setup is much more complex, as each resource-limited site monitors a *streaming distribution of a large number of values* and cannot afford to explicitly capture or model each value separately.

Closest in spirit to our work are the results of [3] and [13], as well as our work on tracking distributed quantiles [8] and join aggregates [7]. All these efforts explicitly consider the tradeoff between accuracy and communication for monitoring a class of continuous queries over distributed streams. With the exception of [7], these earlier papers focus solely on a narrow class of distributed-monitoring queries (e.g., top-k values or one-dimensional quantiles), resulting in special-purpose solutions applicable only to the specific form of queries at hand. More recently, [25, 31] have proposed an approach for efficiently monitoring the value of a *general function/query* over distributed data relative to a given threshold. Their solution relies on interesting geometric arguments for breaking up a global threshold condition on a function into "safe" local conditions that can be checked locally at each site. Still, [25, 31] focus on monitoring a *distributed trigger* condition rather than a distributed query result with approximation-error guarantees; perhaps more importantly, they assume that the full state of the stream can be maintained at both the remote sites and the coordinator. [7] considers monitoring the same class of sketch-based query estimates as we do. Their proposed approach is again purpose-built for the specific type of queries; furthermore, as our experimental results show, the effective combination of the generic geometric monitoring method of [25, 31] and sketch-based query estimates (as proposed in this paper) can give significant performance benefits over the approach in [7].[1]

Our Contributions. In this paper, we propose novel algorithmic techniques for efficiently tracking sketch-based approximations for a broad class of complex aggregate queries over massive, distributed data streams. Our tracking protocols are based on a novel combination of the geometric method of Sharfman et al. [25, 31] for monitoring general threshold conditions over distributed streams and AMS sketch estimators for querying massive streaming data [1, 2, 15]. The effective incorporation of sketching techniques significantly expands the scope of the original geometric method, allowing it to efficiently track a broad class of complex queries over massive, high-dimensional distributed data streams with provable error guarantees. More specifically, we focus on the class of stream queries supported by AMS sketching tools, including general inner products (i.e., join aggregates), as well as the special cases of L_2-norms (i.e., self-join sizes) and range aggregates (e.g., for tracking quantiles, histograms, wavelets, and heavy-hitters over the streams) [7]. One of our key technical insights is that, by exploiting properties of AMS sketches, our algorithms can perform highly-efficient geometric monitoring in a *much lower-dimensional space*. Another major technical challenge lies in effectively dealing with the highly non-linear median operator (that is required for estimation over AMS sketches) in the context of geometric function monitoring. We propose novel geometric algorithms for tracking medians computed over AMS sketches of the streams for different types of distributed stream queries of high practical interest. Our experimental study with real-life data sets demonstrates the practical benefits of our approach, showing consistent gains of up to 35% in terms of total communication cost compared to the current state-of-the-art method [7]; furthermore, our techniques demonstrate even more impressive benefits (of over 100%) when focusing on the communication costs of data (i.e., sketch) shipping in the system.

Roadmap. The remainder of this paper is organized as follows. Section 2 discusses background material on distributed streaming, sketches, and the geometric method. In Section 3, we present our novel geometric monitoring schemes for sketch-based approximate query tracking. Section 4 presents the results of our experimental study. Finally, we conclude the paper and discuss future directions in Section 5.

2. PRELIMINARIES AND PROBLEM SETUP

System Architecture. We consider a distributed-computing environment, comprising a collection of k *remote sites* and a designated

[1]Note that [7] also proposes the idea of using *prediction models* for local data streams, which is orthogonal to the work presented in this paper. In fact, the application of prediction models within the geometric monitoring method has already been explored in a recent paper [17].

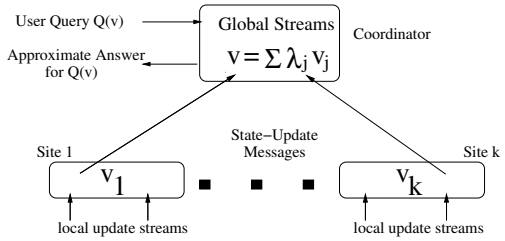


Figure 1: Distributed stream processing architecture.

coordinator site. Streams of data updates arrive continuously at remote sites, while the coordinator site is responsible for generating approximate answers to (possibly, continuous) user queries posed over the *unions* of remotely-observed streams (across all sites). Following earlier work in the area [3, 7, 8, 13, 30], our distributed stream-processing model does not allow direct communication between remote sites; instead, as illustrated in Figure 1, a remote site exchanges messages only with the coordinator, providing it with state information on its (locally-observed) streams. Note that such a hierarchical processing model is, in fact, representative of a large class of applications, including network monitoring where a central Network Operations Center (NOC) is responsible for processing network traffic statistics (e.g., link bandwidth utilization, IP source-destination byte counts) collected at switches, routers, and/or Element Management Systems (EMSs) distributed across the network.

Each remote site $j \in \{1, \ldots, k\}$ observes (possibly, several) local update streams that incrementally render a *local statistics vector* $\boldsymbol{v}_j$ capturing the current local state of the observed stream(s) at site j. As an example, in the case of IP routers monitoring the number of `TCP` and `UDP` packets exchanged between source and destination IP addresses, the local statistics vector $\boldsymbol{v}_j$ has 2×2^{64} entries capturing the up-to-date frequencies for specific (source, destination) pairs observed in `TCP` and `UDP` packets routed through router j. (For instance, the first (last) 2^{64} entries of $\boldsymbol{v}_j$ could be used for `TCP` (respectively, `UDP`) packet frequencies.) All local statistics vectors $\boldsymbol{v}_j$ in our distributed streaming architecture change dynamically over time — when necessary, we make this dependence explicit, using $\boldsymbol{v}_j(t)$ to denote the state of the vector at time t (assuming a consistent notion of "global time" in our distributed system). The unqualified notation $\boldsymbol{v}_j$ typically refers to the *current* state of the local statistics vector.

We define the *global statistics vector* $\boldsymbol{v}$ of our distributed stream(s) as any *weighted average* (i.e., convex combination) of the local statistics vectors $\{\boldsymbol{v}_j\}$; that is, $\boldsymbol{v} = \sum_{j=1}^{k} \lambda_j \boldsymbol{v}_j$, where $\sum_j \lambda_j = 1$ and $\lambda_j \geq 0$ for all j. (Again, to simplify notation, we typically omit the explicit dependence on time when referring to the current global vector.) Our focus is on the problem of effectively answering user queries (or, functions) over the global statistics vector at the coordinator site. Rather than one-time query/function evaluation, we assume a continuous-querying environment which implies that the coordinator needs to *continuously maintain* (or, *track*) the answers to queries as the local update streams $\boldsymbol{v}_j$ evolve at individual remote sites. There are two defining characteristics of our problem setup that raise difficult algorithmic challenges for our query tracking problems:

• *The distributed nature and large volumes of local streaming data* imply important communication and space/time efficiency concerns. Naïve schemes that accurately track query answers by forcing remote sites to ship every remote stream update to the coordinator are clearly impractical, since they can impose an inordinate burden on the underlying communication infrastructure (especially, for high-rate data streams and large numbers of remote sites). Furthermore, the voluminous nature of the local data streams implies that effective streaming tools are needed at the remote sites in order to manage the streaming local statistics vectors in sublinear space/time. A main part of our approach is to adopt the paradigm of continuous tracking of *approximate* query answers at the coordinator site with strong guarantees on the quality of the approximation. This allows our schemes to effectively trade-off space/time/communication efficiency and query-approximation accuracy in a precise, quantitative manner.

• *General, non-linear queries/functions* imply fundamental and difficult challenges for distributed monitoring. For the case of linear functions, a number of approaches have been proposed that rely on the key idea of allocating appropriate *"slacks"* to the remote sites based on their locally-observed function values (e.g., [3, 24, 30]). Unfortunately, it is not difficult to find examples of simple *non-linear* functions on one-dimensional data, where it is basically impossible to make any assumptions about the value of the global function based on the function values observed locally at the sites [31]. This renders conventional slack-allocation schemes inapplicable in our setting.

As a concrete example of complex function tracking, consider the aforementioned global vector $\boldsymbol{v} = \langle \boldsymbol{t}, \boldsymbol{u} \rangle$ of `TCP` and `UDP` packet frequencies observed over a collection of IP routers. where $\boldsymbol{t}$, $\boldsymbol{u}$ are the subvectors of $\boldsymbol{v}$ corresponding to `TCP` and `UDP` traffic, respectively. Tracking the (non-linear) inner-product function $f(\boldsymbol{v}) = \boldsymbol{t} \cdot \boldsymbol{u} = \sum_i \boldsymbol{t}[i]\boldsymbol{u}[i]$ (i.e., the size of the join of the two traffic distributions over (source, destination)) can allow the NOC to effectively monitor the strength of the correlation across the two types of traffic in the underlying set of routers. Clearly, simple slack-allocation techniques [3, 24, 30] cannot be applied here.

AMS Stream Sketches. Techniques based on small-space pseudo-random *sketch* summaries of the data have proved to be very effective tools for dealing with massive, rapid-rate data streams in centralized settings [1, 2, 11, 15, 20]. The key idea in such sketching techniques is to represent a streaming frequency vector $\boldsymbol{v}$ using a much smaller (typically, randomized) *sketch* vector (denoted by $\mathtt{sk}(\boldsymbol{v})$) that (1) can be easily maintained as the updates incrementally rendering $\boldsymbol{v}$ are streaming by, and (2) provide probabilistic guarantees for the quality of the data approximation. The widely used AMS sketch (proposed by Alon, Matias, and Szegedy in their seminal paper [2]) defines i^{th} sketch entry $\mathtt{sk}(\boldsymbol{v})[i]$ as the random variable $\sum_k \boldsymbol{v}[k] \cdot \xi_i[k]$, where $\{\xi_i\}$ is a family of four-wise independent binary random variables uniformly distributed in $\{-1, +1\}$ (with mutually-independent families used across different entries of the sketch). The key here is that, using appropriate pseudo-random hash functions, each such family can be efficiently constructed on-line in small (logarithmic) space [2]. Note that, by construction, each entry of $\mathtt{sk}(\boldsymbol{v})$ is essentially a *randomized linear projection* (i.e., an inner product) of the $\boldsymbol{v}$ vector (using the corresponding ξ family), that can be easily maintained (using a simple counter) over the input update stream. Another important property is the *linearity* of AMS sketches: Given two "parallel" sketches (built using the same ξ families) $\mathtt{sk}(\boldsymbol{v}_1)$ and $\mathtt{sk}(\boldsymbol{v}_2)$, the sketch of the union of the two underlying streams (i.e., the streaming vector $\boldsymbol{v}_1 + \boldsymbol{v}_2$) is simply the component-wise sum of their sketches; that is, $\mathtt{sk}(\boldsymbol{v}_1 + \boldsymbol{v}_2) = \mathtt{sk}(\boldsymbol{v}_1) + \mathtt{sk}(\boldsymbol{v}_2)$. This linearity makes such sketches particularly useful in *distributed* streaming settings [7].

The following theorem summarizes some of the basic estimation properties of AMS sketches for (centralized) stream query processing. (Throughout, the notation $x \in (y \pm z)$ is equivalent to $|x - y| \leq |z|$.) We use $f_{\text{AMS}}()$ to denote the standard *AMS estimator function*, involving both averaging and median-selection opera-

tions over the components of the sketch-vector inner product [1, 2]. Formally, each sketch vector can be conceptually viewed as a two-dimensional $n \times m$ array, where $n = O(\frac{1}{\epsilon^2})$, $m = O(\log(1/\delta))$ and ϵ, $1-\delta$ denote the desired bounds on error and probabilistic confidence (respectively), and the AMS estimator function is defined as:

$$f_{\text{AMS}}(\texttt{sk}(\boldsymbol{v}), \texttt{sk}(\boldsymbol{u})) = \underset{i=1,\ldots,m}{\text{median}}\{\frac{1}{n}\sum_{l=1}^{n} \texttt{sk}(\boldsymbol{v})[l,i] \cdot \texttt{sk}(\boldsymbol{u})[l,i]\}.$$

THEOREM 2.1 ([1, 2]). *Let $\texttt{sk}(\boldsymbol{v})$ and $\texttt{sk}(\boldsymbol{u})$ denote two parallel sketches comprising $O(\frac{1}{\epsilon^2}\log(1/\delta))$ counters, built over the streams $\boldsymbol{v}$ and $\boldsymbol{u}$. Then, with probability at least $1-\delta$, $f_{\text{AMS}}(\texttt{sk}(\boldsymbol{v}), \texttt{sk}(\boldsymbol{u})) \in (\boldsymbol{v}\cdot\boldsymbol{u} \pm \epsilon\|\boldsymbol{v}\|\|\boldsymbol{u}\|)$. The processing time required to maintain each sketch is $O(\frac{1}{\epsilon^2}\log(1/\delta))$ per update.*

Thus, AMS sketch estimators can effectively approximate *inner-product queries* $\boldsymbol{v}\cdot\boldsymbol{u} = \sum_i \boldsymbol{v}[i]\cdot\boldsymbol{u}[i]$ over streaming data vectors and tensors. Such inner products naturally map to *join and multi-join aggregates* when the the vectors/tensors capture the frequency distribution of the underlying join attribute(s) [15]. Furthermore, they can capture several other interesting query classes, including range and quantile queries [19], heavy hitters and top-k queries [4], and approximate histogram and wavelet representations [9, 20, 32]. An interesting special case is that of the (squared) L_2 *norm* (or, *self-join*) query (i.e., $\boldsymbol{u} = \boldsymbol{v}$): Theorem 2.1 implies that the AMS estimator $f_{\text{AMS}}(\texttt{sk}(\boldsymbol{v}), \texttt{sk}(\boldsymbol{v}))$ (or, simply $f_{\text{AMS}}(\texttt{sk}(\boldsymbol{v}))$) is within ϵ relative error of the true squared L_2 norm $\|\boldsymbol{v}\|^2 = \sum_k(\boldsymbol{v}[k])^2$; that is, $f_{\text{AMS}}(\texttt{sk}(\boldsymbol{v})) \in (1\pm\epsilon)\|\boldsymbol{v}\|^2$. To provide ϵ relative-error guarantees for the general inner-product query $\boldsymbol{v}\cdot\boldsymbol{u}$, Theorem 2.1 can be applied with error bound $\epsilon' = \epsilon(\boldsymbol{v}\cdot\boldsymbol{u})/(\|\boldsymbol{v}\|\|\boldsymbol{u}\|)$, giving a total sketching space requirement of $O(\frac{\|\boldsymbol{v}\|^2\|\boldsymbol{u}\|^2}{\epsilon^2(\boldsymbol{v}\cdot\boldsymbol{u})^2}\log(1/\delta))$ counters [1].

A drawback of AMS sketches is that every streaming update must "touch" every component of the sketch vector (to update the corresponding randomized linear projection). This can be problematic for massive, rapid-rate data streams, especially when a tight error guarantee ϵ is required. The *Fast-AMS sketch* [7] solves this problem by guaranteeing *logarithmic-time* (i.e., $O(\log(1/\delta))$) sketch-update costs, while offering the same space/accuracy tradeoff as the basic AMS sketch (through a more careful analysis) [7]. (Our implementation in Section 4 employs the Fast-AMS variant.)

The Geometric Method. Sharfman et al. [31] consider the fundamental problem of *distributed threshold monitoring*; that is, determine whether $f(\boldsymbol{v}) < \tau$ or $f(\boldsymbol{v}) > \tau$, for a given (general) function $f()$ over the global statistics vector and a fixed threshold τ. Their key idea is that, since it is generally impossible to connect the locally-observed values of $f()$ to the global value $f(\boldsymbol{v})$, one can employ geometric arguments to monitor the *domain* (rather than the range) of the monitored function $f()$. More specifically, assume that at any point in time, each site j has informed the coordinator of some prior state of its local vector $\boldsymbol{v}_j^p$; thus, the coordinator has an estimated global vector $\boldsymbol{e} = \boldsymbol{v}^p = \sum_{j=1}^k \lambda_j \boldsymbol{v}_j^p$. Clearly, the updates arriving at sites can cause the local vectors $\boldsymbol{v}_j$ to drift too far from their previously reported values $\boldsymbol{v}_j^p$, possibly leading to a violation of the τ threshold. Let $\Delta\boldsymbol{v}_j = \boldsymbol{v}_j - \boldsymbol{v}_j^p$ denote the local *delta vector* (due to updates) at site j, and let $\boldsymbol{u}_j = \boldsymbol{e} + \Delta\boldsymbol{v}_j$ be the *drift vector* from the previously reported estimate at site j. We can then express the current global statistics vector $\boldsymbol{v}$ in terms of the drift vectors:

$$\boldsymbol{v} = \sum_{j=1}^{k}\lambda_j(\boldsymbol{v}_j^p + \Delta\boldsymbol{v}_j) = \boldsymbol{e} + \sum_{j=1}^{k}\lambda_j\Delta\boldsymbol{v}_j = \sum_{j=1}^{k}\lambda_j(\boldsymbol{e}+\Delta\boldsymbol{v}_j).$$

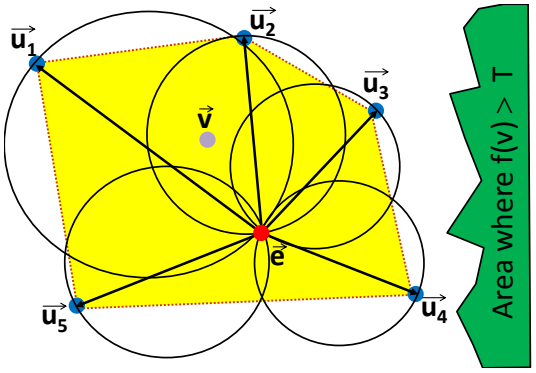


Figure 2: Estimate vector $\boldsymbol{e}$, delta vectors $\Delta\boldsymbol{v}_j$ (arrows out of $\boldsymbol{e}$), convex hull enclosing the current global vector $\boldsymbol{v}$ (dotted outline), and bounding balls $B(\boldsymbol{e}+\frac{1}{2}\Delta\boldsymbol{v}_j, \frac{1}{2}\|\Delta\boldsymbol{v}_j\|)$.

That is, the current global vector is a convex combination of drift vectors and, thus, guaranteed to lie somewhere within the convex hull of the delta vectors around $\boldsymbol{e}$. Figure 2 depicts an example in $d = 2$ dimensions. The current value of the global statistics vector lies somewhere within the shaded convex-hull region; thus, as long as the convex hull does not overlap the inadmissible region (i.e., the region $\{\boldsymbol{v} \in \mathbb{R}^2 : f(\boldsymbol{v}) > \tau\}$ in Figure 2), we can guarantee that the threshold has not been violated (i.e., $f(\boldsymbol{v}) \leq \tau$)).

The problem, of course, is that the $\Delta\boldsymbol{v}_j$'s are spread across the sites and, thus, the above condition cannot be checked locally. To transform the global condition into a local constraint, we place a d-dimensional *bounding ball* $B(\boldsymbol{c}, r)$ around each local delta vector, of radius $r = \frac{1}{2}\|\Delta\boldsymbol{v}_j\|$ and centered at $\boldsymbol{c} = \boldsymbol{e} + \frac{1}{2}\Delta\boldsymbol{v}_j$ (see Figure 2). It can be shown that (in any dimensionality d) the union of all these balls completely covers the convex hull of the drift vectors [31]. This observation effectively reduces the problem of monitoring the global statistics vector to the local problem of each remote site monitoring the ball around its local delta vector.

More specifically, given the monitored function $f()$ and threshold τ, we can partition the d-dimensional space into two sets $V = \{\boldsymbol{v} : f(\boldsymbol{v}) > \tau\}$ and $\overline{V} = \{\boldsymbol{v} : f(\boldsymbol{v}) \leq \tau\}$. (Note that these sets can be arbitrarily complex, e.g., they may comprise multiple disjoint regions of $\mathbb{R}^d$.) The basic protocol is now quite simple: Each site monitors its delta vector $\Delta\boldsymbol{v}_j$ and, with each update, checks whether its bounding ball $B(\boldsymbol{e}+\frac{1}{2}\Delta\boldsymbol{v}_j, \frac{1}{2}\|\Delta\boldsymbol{v}_j\|)$ is *monochromatic*, i.e., all points in the ball lie within the same region (V or $\overline{V}$). If this is not the case, we have a *local threshold violation*, and the site communicates its local $\Delta\boldsymbol{v}_j$ to the coordinator. The coordinator then initiates a *synchronization process* that typically tries to resolve the local violation by communicating with only a subset of the sites in order to "balance out" the violating $\Delta\boldsymbol{v}_j$ and ensure the monochromicity of all local bounding balls [31]. In the worst case, the delta vectors from all k sites are collected, leading to an accurate estimate of the current global statistics vector, which is by definition monochromatic (since all bounding balls have 0 radius).

In more recent work, Sharfman et al. [25] demonstrate that their geometric monitoring method can employ properties of the function and the data to guide the choice of a global *reference point* and local *bounding ellipsoids* for defining the local constraints. Furthermore, they show that the local bounding balls/ellipsoids defined by the geometric method are actually special cases of a more general theory of *Safe Zones (SZs)*, which can be broadly defined as *convex subsets of the admissible region* of a threshold query. It is not diffi-

cult to see that, as long as the local drift vectors stay within such a SZ, the global vector is guaranteed (by convexity) to be within the admissible region of the query [25].

3. SKETCH-BASED APPROXIMATE GEOMETRIC MONITORING

In this section, we develop our approach for geometric monitoring of non-linear, inner-product queries using AMS sketches. The sketching idea offers an effective streaming dimensionality-reduction tool that significantly expands the scope of the original geometric method [31], allowing it to handle massive, high-dimensional distributed data streams in an efficient manner with approximation-quality guarantees. The key technical observation is that, by exploiting properties of the AMS estimator function, geometric monitoring can now take place in a *much lower-dimensional space*, allowing for communication-efficient monitoring. Effectively dealing with the highly non-linear median operator in the AMS estimator also mandates novel algorithmic solutions. We start by showing how our approximate function monitoring problem can be transformed into low-dimensional threshold crossing queries for the geometric method. To simplify notation, in the remainder of this section, we use $\tilde{\boldsymbol{v}}_i$ to denote the AMS sketch at remote sites and let $\tilde{\boldsymbol{v}} = \sum_j \tilde{\boldsymbol{v}}_j$ denote the global AMS sketch of the entire distributed stream; similarly, $\tilde{\boldsymbol{v}}_j^p$, $\tilde{\boldsymbol{v}}^p = \sum_i \tilde{\boldsymbol{v}}_j^p$ denote the local/global sketch values last communicated to the coordinator.

3.1 From Threshold Crossing to Approximate Function Monitoring

Consider the task of monitoring (at the coordinator) the value of a function $f()$ over the (full) global statistics vector $\boldsymbol{v}$ to within θ relative error. (Our discussion here focuses on relative error – the case of monitoring to within bounded *absolute* error can be handled in a similar manner, e.g., using the absolute to relative error transformation outlined under Theorem 2.1.) Since the coordinator only holds the estimated value of the global statistics vector $\boldsymbol{v}^p$ based on the most recent site updates, our monitoring protocol would have to guarantee that the estimated function value carries at most θ relative error compared to the up-to-date value $f(\boldsymbol{v}) = f(\boldsymbol{v}(t))$, that is $f(\boldsymbol{v}^p) \in (1 \pm \theta) f(\boldsymbol{v})$, which is obviously equivalent to monitoring two threshold queries on $f(\boldsymbol{v})$:

$$f(\boldsymbol{v}) \geq \frac{f(\boldsymbol{v}^p)}{1+\theta} \quad \text{and} \quad f(\boldsymbol{v}) \leq \frac{f(\boldsymbol{v}^p)}{1-\theta}.$$

Now, since we assume that the remote sites only maintain *AMS sketches* of their local vectors, all we have at our disposal are the sketched versions of the $\boldsymbol{v}$ and $\boldsymbol{v}^p$ vectors (denoted by $\tilde{\boldsymbol{v}}$ and $\tilde{\boldsymbol{v}}^p$, respectively), and the corresponding function values are approximated through the AMS estimator function $f_{\text{AMS}}()$ (Theorem 2.1). This, of course, implies a *sketching error* ϵ in the function values which can be bounded with the help of Theorem 2.1 so that $f_{\text{AMS}}(\tilde{\boldsymbol{v}}) \in (1 \pm \epsilon) f(\boldsymbol{v})$ with high probability (whp). Since our end goal is to guarantee that the sketch-based estimate available at the coordinator $f_{\text{AMS}}(\tilde{\boldsymbol{v}}^p)$ is within θ relative error, the above threshold monitoring conditions become:

$$f(\boldsymbol{v}) \geq \frac{f_{\text{AMS}}(\tilde{\boldsymbol{v}}^p)}{1+\theta} \quad \text{and} \quad f(\boldsymbol{v}) \leq \frac{f_{\text{AMS}}(\tilde{\boldsymbol{v}}^p)}{1-\theta},$$

and, since $f_{\text{AMS}}(\tilde{\boldsymbol{v}}) \in (1 \pm \epsilon) f(\boldsymbol{v})$, it is not difficult to see that these two conditions are satisfied (whp) as long as:

$$f_{\text{AMS}}(\tilde{\boldsymbol{v}}) \geq \frac{f_{\text{AMS}}(\tilde{\boldsymbol{v}}^p)(1+\epsilon)}{1+\theta} \quad \text{and} \quad f_{\text{AMS}}(\tilde{\boldsymbol{v}}) \leq \frac{f_{\text{AMS}}(\tilde{\boldsymbol{v}}^p)(1-\epsilon)}{1-\theta}. \tag{1}$$

These are exactly the threshold conditions that our approximate function monitoring protocols will need to track. Note that $f_{\text{AMS}}(\tilde{\boldsymbol{v}}^p)$ in the above expression is a constant (based on the latest communication of the coordinator with the remote sites). When either of the above conditions is violated, some (possibly all) remote sites must *flush* their current local stream estimates to the coordinator, updating $\tilde{\boldsymbol{v}}^p$ so that the difference between $f_{\text{AMS}}(\tilde{\boldsymbol{v}})$ and $f_{\text{AMS}}(\tilde{\boldsymbol{v}}^p)$ is again small. Also, observe that the condition $\frac{1+\epsilon}{1+\theta} \leq \frac{1-\epsilon}{1-\theta}$ always holds as long as $\theta \geq \epsilon$, which is obviously the case (the overall error guarantee cannot be tighter than the incurred sketching error).

As discussed earlier, for remote site j, $\tilde{\boldsymbol{v}}_j$ denotes the sketch of local stream updates, and $\tilde{\boldsymbol{v}}_j^p$ the sketch last flushed to the coordinator. Exploiting the linearity of AMS sketches, remote site j maintains $\Delta\tilde{\boldsymbol{v}}_j = \tilde{\boldsymbol{v}}_j - \tilde{\boldsymbol{v}}_j^p$, corresponding to stream updates since the last flush. At the time of the next flush, the remote site simply transfers $\Delta\tilde{\boldsymbol{v}}_j$ to the coordinator and resets its local delta sketch to zero. (If the stream updates sketched in $\Delta\tilde{\boldsymbol{v}}_j$ are few, in order to reduce communication cost, the remote site may send the updates verbatim to the coordinator.)

3.2 Applying the Geometric Method: Overview

Having reduced approximate distributed stream monitoring to appropriate threshold-crossing conditions (Eqn. (1)), we now turn our attention to the issue of effectively applying the geometric method to the problem at hand. A direct application would take the distributed stream sketch $\tilde{\boldsymbol{v}}$ as the global statistics vector, scaling each local sketch $\tilde{\boldsymbol{v}}_j$ by the number of sites k to obtain the local statistics vectors (in order to satisfy the convex combination requirement of the geometric method); then, the geometric method could be employed to monitor the two threshold conditions on $f_{\text{AMS}}(\tilde{\boldsymbol{v}})$ in the $(n \times m)$-dimensional sketching space (Section 2).

Unfortunately, such a direct application of the geometric method turns out to perform poorly in practice, giving high communication overheads. The problem here is that, even though sketch vectors are a compressed, $(n \times m)$-dimensional representation of the full stream, they can still get fairly large, especially when tight error bounds are required. Thus, when a local threshold violation occurs at a remote site, it triggers a balancing process that requires some of the sites to transmit their local statistics (i.e., sketches) to the coordinator, imposing high communication overheads.

To address this issue, we develop a novel technique that allows us to track the threshold conditions on $f_{\text{AMS}}(\tilde{\boldsymbol{v}})$ through geometric monitoring in a much lower-dimensional space. More specifically, consider a sketch $\boldsymbol{x}$ as a two-dimensional $n \times m$ array, and let $\boldsymbol{x}[i]$ $(i = 1, \ldots m)$ denote the n-vector corresponding to the i^{th} column of the sketch matrix. We define the local statistics vector for remote site j as the m-*dimensional error vector* $\boldsymbol{d}_j$, where

$$\boldsymbol{d}_j[i] \;=\; \|\Delta\tilde{\boldsymbol{v}}_j[i]\| \;=\; \|\tilde{\boldsymbol{v}}_j[i] - \tilde{\boldsymbol{v}}_j^p[i]\|,$$

for $i = 1, \ldots, m$, and the global statistics vector as the (m-dimensional) average error vector $\boldsymbol{d} = \frac{1}{k}\sum_{j=1}^{k} \boldsymbol{d}_j$. In what follows, we show how to construct functions $F_u()$ and $F_l()$ of $\boldsymbol{d}$ that provide lower and upper bounds on $f_{\text{AMS}}(\tilde{\boldsymbol{v}})$; that is,

$$F_l(\boldsymbol{d}) \;\leq\; f_{\text{AMS}}(\tilde{\boldsymbol{v}}) \;\leq\; F_u(\boldsymbol{d}).$$

We can then monitor the threshold-crossing conditions on $f_{\text{AMS}}(\tilde{\boldsymbol{v}})$ (Eqn. (1)) using the geometric method for $F_u(\boldsymbol{d})$ and $F_l(\boldsymbol{d})$ in the m-dimensional space of error vectors $\boldsymbol{d}$. It is important to note that this optimization implies huge communication savings: Sketch matrices are typically very "thin", i.e., $n >> m$, since n depends quadratically on the sketching error ϵ, whereas m depends only logarithmically on the desired confidence δ [2, 1, 9, 15].

Another major technical challenge that arises is how to effectively test the monochromicity of bounding balls in the resulting lower-dimensional space with respect to threshold conditions involving the highly non-linear median operator present in the AMS estimator (as well as the upper/lower bound functions $F_u()$ and $F_l()$). Our techniques and analyses make use of three well-known properties of the median operator:

Monotonicity: If $\boldsymbol{x}[i] \leq \boldsymbol{y}[i]$ for all i, then $\text{median}_i\{\boldsymbol{x}[i]\} \leq \text{median}_i\{\boldsymbol{y}[i]\}$.

Distributivity: For any monotone function $f()$, $\text{median}_i\{f(\boldsymbol{x}[i])\} = f(\text{median}_i\{\boldsymbol{x}[i]\})$.

Homogeneity: $\forall \lambda \in \mathbb{R}$, $\text{median}_i\{\lambda \boldsymbol{x}[i]\} = \lambda\, \text{median}_i\{\boldsymbol{x}[i]\}$.

We propose a number of novel algorithmic techniques to address the aforementioned technical challenges for three different types of distributed stream queries of high practical interest. We start with the easier cases of L_2-norm (i.e., self-join) and range queries, and then extend our approach to the case of general inner-product (i.e., binary-join) queries.

3.3 Monitoring Self-Joins

In the case of (approximate) self-join/L_2-norm queries, our goal is to track an estimate of the (squared) norm of a frequency vector using AMS sketches. Thus, we need to monitor the values of the AMS estimator function

$$f_{\text{AMS}}(\tilde{\boldsymbol{v}}) = \underset{i=1,\ldots,m}{\text{median}}\{\frac{1}{n}\sum_{l=1}^{n}(\tilde{\boldsymbol{v}}[l,i])^2\} = \underset{i=1,\ldots,m}{\text{median}}\{\frac{1}{n}\|\tilde{\boldsymbol{v}}[i]\|^2\} \tag{2}$$

where $\tilde{\boldsymbol{v}}$ is an $n \times m$-sized AMS sketch and $\tilde{\boldsymbol{v}}[i]$ is the i^{th}-th column of the sketch. Using the distributivity of the median operator, the threshold-crossing conditions in Eqn. (1) become:

$$\sqrt{n\frac{1+\epsilon}{1+\theta}f_{\text{AMS}}(\tilde{\boldsymbol{v}}^p)} \leq \underset{i=1,\ldots,m}{\text{median}}\{\|\tilde{\boldsymbol{v}}[i]\|\} \leq \sqrt{n\frac{1-\epsilon}{1-\theta}f_{\text{AMS}}(\tilde{\boldsymbol{v}}^p)}.$$

We now develop "safe" threshold conditions over $\mathbb{R}^m$ for the above monitoring problem using upper/lower bound functions defined over the m-dimensional error vector $\boldsymbol{d}$. By definition, at site j, $\boldsymbol{d}_j[i] = \|\tilde{\boldsymbol{v}}_j[i] - \tilde{\boldsymbol{v}}_j^p[i]\|$; thus, applying the the triangle inequality, we have

$$\|\tilde{\boldsymbol{v}}[i] - \tilde{\boldsymbol{v}}^p[i]\| \leq \sum_{j=1}^{k}\|\tilde{\boldsymbol{v}}_j[i] - \tilde{\boldsymbol{v}}_j^p[i]\| = \sum_{j=1}^{k}\boldsymbol{d}_j[i] = k\boldsymbol{d}[i], \tag{3}$$

or, equivalently,

$$\|\tilde{\boldsymbol{v}}^p[i]\| - k\boldsymbol{d}[i] \leq \|\tilde{\boldsymbol{v}}[i]\| \leq \|\tilde{\boldsymbol{v}}^p[i]\| + k\boldsymbol{d}[i].$$

Then, by monotonicity of the median, it is sufficient to monitor the following threshold conditions over $\boldsymbol{d} \in \mathbb{R}^m$:

$$F_u(\boldsymbol{d}) = \underset{i}{\text{median}}\{\|\tilde{\boldsymbol{v}}^p[i]\| + k\boldsymbol{d}[i]\} \leq \sqrt{n\frac{1-\epsilon}{1-\theta}f_{\text{AMS}}(\tilde{\boldsymbol{v}}^p)}$$

$$F_l(\boldsymbol{d}) = \underset{i}{\text{median}}\{\|\tilde{\boldsymbol{v}}^p[i]\| - k\boldsymbol{d}[i]\} \geq \sqrt{n\frac{1+\epsilon}{1+\theta}f_{\text{AMS}}(\tilde{\boldsymbol{v}}^p)}.$$

Geometric Monitoring for the Median. By dividing both sides of the above threshold conditions over $\mathbb{R}^m$ by $\pm k$ and by virtue of the homogeneity of the median, both conditions take the general form

$$F(\boldsymbol{d}) = \underset{i=1,\ldots,m}{\text{median}}\{\boldsymbol{a}[i] + \boldsymbol{d}[i]\} \leq \zeta,$$

where $\boldsymbol{a}$ is a constant m-dimensional vector and $\zeta \in \mathbb{R}$.

Algorithm 1: Computing the distance of a vector to the region defined by a median threshold.

Data: $\boldsymbol{c} = [\boldsymbol{c}[1], \ldots, \boldsymbol{c}[m]]$, $\boldsymbol{a} = [\boldsymbol{a}[1], \ldots, \boldsymbol{a}[m]]$: m-dimensional vectors; ζ: real.
Result: The distance of $\boldsymbol{c}$ to the region $\{\boldsymbol{x} \in \mathbb{R}^m | \text{median}_i\{\boldsymbol{a}[i] + \boldsymbol{x}[i]\} \geq \zeta\}$.

```
begin
    let z = a + c
    Sort(z, ascending)
    r = 0
    for i ⟵ ⌊(m+1)/2⌋ to m do
        if z[i] < ζ then
            r += (ζ − z[i])^2
            z[i] = ζ
    return ρ_j = √r
end
```

To monitor such conditions using the geometric method, we must be able, given a bounding ball $B(\boldsymbol{c}, \rho)$ in $\mathbb{R}^m$, to efficiently decide whether the ball is monochromatic; that is, whether $\text{median}_i\{\boldsymbol{a}[i] + \boldsymbol{x}[i]) \leq \zeta$, for all $\boldsymbol{x} \in B(\boldsymbol{c}, \rho)$, This can be done by determining the Euclidean distance ρ_ζ of the ball center $\boldsymbol{c}$ from the closest point in the inadmissible region $Z = \{\boldsymbol{x} \in \mathbb{R}^m | F(\boldsymbol{x}) \geq \zeta\}$; then, the ball $B(\boldsymbol{c}, \rho)$ is monochromatic if and only if $\rho \leq \rho_\zeta$.

We now show how to efficiently compute this distance to the inadmissible region. Clearly, if $\text{median}_i\{\boldsymbol{a}[i] + \boldsymbol{c}[i]\} \geq \zeta$, then $\rho_\zeta = 0$. Else, we can employ a greedy algorithm to find a point $\boldsymbol{z}$ on the boundary of the inadmissible region Z (with $\text{median}_i\{\boldsymbol{a}[i] + \boldsymbol{z}[i]\}) = \zeta$), such that no other point in Z is closer to the ball center $\boldsymbol{c}$ (note that this point $\boldsymbol{z}$ is not necessarily unique). Algorithm 1 constructs such a boundary point $\boldsymbol{z}$ in a greedy manner: Starting with $\boldsymbol{z} = \boldsymbol{a} + \boldsymbol{c}$, it takes all coordinates from rank $\lfloor \frac{m+1}{2} \rfloor$ to rank m that are $< \zeta$ and sets them equal to ζ in order to reach the boundary; then, it returns the distance $\rho_\zeta = \|\boldsymbol{c} - \boldsymbol{z}\|$. The following theorem summarizes our analysis (due to space constraints, the proof is deferred to the full paper).

THEOREM 3.1. *Algorithm 1 correctly computes the minimum Euclidean distance of point $\boldsymbol{c} \in \mathbb{R}^m$ from the inadmissible region $Z = \{\boldsymbol{x} \in \mathbb{R}^m | \text{median}_i\{\boldsymbol{a}[i] + \boldsymbol{x}[i]\} \geq \zeta\}$ in time $O(m \log m)$.*

3.4 Monitoring Range Aggregates

We now turn our attention to a different special type of inner-product queries, namely the inner product of a distributed data stream with a *constant* vector $\boldsymbol{b}$. An important special case here is that of *range aggregates*, in which the constant vector $\boldsymbol{b}$ simply contains non-zero values for a subset S of values in the joint data distribution in the streaming vector $\boldsymbol{v}$, and zero everywhere else; thus, $\boldsymbol{b} \cdot \boldsymbol{v} = \sum_{i \in S}\boldsymbol{b}[i]\boldsymbol{v}[i]$, i.e., the distribution aggregate (e.g., the number of tuples) in range S. These aggregates can, of course, be estimated using an AMS sketch estimator $f_{\text{AMS}}(\tilde{\boldsymbol{v}}, \tilde{\boldsymbol{b}})$ (where $\tilde{\boldsymbol{b}}$ is the constant sketch vector for $\boldsymbol{b}$), with the quality guarantees outlined in Theorem 2.1. Such approximate range aggregates over AMS sketches have been utilized in several important streaming applications, including the construction of effective quantile, histogram, and wavelet summaries over streaming data [9, 19, 20, 32]. For instance, in the case of wavelets, we are interested in estimat-

ing large wavelet coefficients, which are inner product of the data distribution with constant wavelet-basis vectors [9].[2]

Within our framework, we are asked to monitor the estimator

$$f_{\text{AMS}}(\tilde{\boldsymbol{v}}, \tilde{\boldsymbol{b}}) = \underset{i=1,\ldots,m}{\text{median}}\{\frac{1}{n}\tilde{\boldsymbol{b}}[i]\tilde{\boldsymbol{v}}[i]\}.$$

The global statistic again consists of a single sketch (since $\boldsymbol{b}$ is constant). Thus, we can start from Eqn. (3), and we obtain

$$\left|\boldsymbol{b}[i](\tilde{\boldsymbol{v}}[i] - \tilde{\boldsymbol{v}}^p[i])\right| \;\leq\; \|\boldsymbol{b}[i]\|\|\tilde{\boldsymbol{v}}[i] - \tilde{\boldsymbol{v}}^p[i]\| \;\leq\; k\boldsymbol{d}[i]\|\boldsymbol{b}[i]\|,$$

which yields the two threshold conditions:

$$\underset{i}{\text{median}}\{\boldsymbol{b}[i]\tilde{\boldsymbol{v}}^p[i] + k\boldsymbol{d}[i]\|\boldsymbol{b}[i]\|\} \;\leq\; n\frac{1-\epsilon}{1-\theta}f_{\text{AMS}}(\tilde{\boldsymbol{v}}^p, \boldsymbol{b}) \quad (4)$$

$$\underset{i}{\text{median}}\{\boldsymbol{b}[i]\tilde{\boldsymbol{v}}^p[i] - k\boldsymbol{d}[i]\|\boldsymbol{b}[i]\|\} \;\geq\; n\frac{1+\epsilon}{1+\theta}f_{\text{AMS}}(\tilde{\boldsymbol{v}}^p, \boldsymbol{b}). \quad (5)$$

Geometric Monitoring for the Median of Linear Forms. By dividing both sides by $\pm k$ and by virtue of the homogeneity of the median, both conditions take the form

$$F(\boldsymbol{d}) = \underset{i}{\text{median}}\{\boldsymbol{a}[i] + \boldsymbol{b}[i]\boldsymbol{d}[i]\} \;\leq\; \zeta$$

for $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^m$, where $\boldsymbol{b}$ has nonnegative components and $\zeta \in \mathbb{R}$.

The monochromicity question for ball $B(\boldsymbol{c}, \rho)$ can be addressed in a spirit similar to that of Section 3.3. One small complication arises by the fact that, in this case, ρ_ζ may be undefined! This may occur only if some entries in $\boldsymbol{b}$ are zero, so that the inadmissible region $Z = \{\boldsymbol{x} \in \mathbb{R}^m \mid F(\boldsymbol{x}) \geq \zeta\}$ is empty. To handle this complication smoothly, we apply a standard algebraic perturbation trick; we assume that, when $\boldsymbol{b}[i] = 0$ and $\boldsymbol{a}[i] < \zeta$, then $\boldsymbol{a}[i] + \boldsymbol{b}[i](+\infty) \geq \zeta$. Thus, region Z is never empty, although some of its elements may have infinite coordinates.

As earlier, we wish to minimize $\rho_\zeta^2 = \sum_{i=1}^m (\boldsymbol{x}[i] - \boldsymbol{c}[i])^2$, where $\boldsymbol{x} \in Z$. For each $i = 1, ..., m$, let r_i^2 denote the minimum of $(\boldsymbol{x}[i] - \boldsymbol{c}[i])^2$, such that $\boldsymbol{a}[i] + \boldsymbol{b}[i]\boldsymbol{x}[i] \geq \zeta$. It is easy to derive that,

$$r_i^2 = \begin{cases} 0 & \text{if } \boldsymbol{a}[i] + \boldsymbol{b}[i]\boldsymbol{c}[i] \geq \zeta \\ +\infty & \text{if } \boldsymbol{a}[i] < \zeta \text{ and } \boldsymbol{b}[i] = 0 \\ (\frac{\zeta - \boldsymbol{a}[i]}{\boldsymbol{b}[i]} - \boldsymbol{c}[i])^2 & \text{if } \boldsymbol{a}[i] + \boldsymbol{b}[i]\boldsymbol{c}[i] < \zeta \text{ and } \boldsymbol{b}[i] \neq 0 \end{cases}$$

Then, ρ_ζ^2 is equal to the sum of the $(m+1)/2$ smallest r_i^2s (treating ties arbitrarily). Again, the ball is monochromatic if and only if $\rho \leq \rho_\zeta$.

3.5 Monitoring General Inner Products

We now turn our attention to a more complicated monitoring problem, where the global statistic comprises of the concatenation of two sketches $\langle \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{u}} \rangle$ corresponding to two distributed streams, and the monitored function is the sketch estimate of the inner product of the sketched vectors, corresponding to the size of the inner product (i.e., join):

$$\begin{aligned} f_{\text{AMS}}(\tilde{\boldsymbol{v}}, \tilde{\boldsymbol{u}}) &= \underset{i=1,\ldots,m}{\text{median}}\{\frac{1}{n}\sum_{l=1}^{n}\tilde{\boldsymbol{v}}[l,i]\tilde{\boldsymbol{u}}[l,i]\} \\ &= \underset{i=1,\ldots,m}{\text{median}}\{\frac{1}{n}\tilde{\boldsymbol{v}}[i]\tilde{\boldsymbol{u}}[i]\} \end{aligned}$$

In addition, error vectors are also concatenated, denoted as $\langle \boldsymbol{d}_v, \boldsymbol{d}_u \rangle$.

[2]Note that sketching is employed here since the range queries of interest are *not fixed* (i.e., can vary over time), and a search over the sketch summary is needed to discover the ranges of interest as the stream distribution changes. In simpler scenarios where the range aggregate of interest is fixed, slack-allocation techniques for tracking linear aggregates can be used (e.g., [24]).

We now develop bounds for the monitored function using the error vectors. From Eqn. (3), we can write

$$\tilde{\boldsymbol{v}}[i] = \tilde{\boldsymbol{v}}^p[i] + k\boldsymbol{d}_v[i]\boldsymbol{q}_{v,i} \quad \text{and} \quad \tilde{\boldsymbol{u}}[i] = \tilde{\boldsymbol{u}}^p[i] + k\boldsymbol{d}_u[i]\boldsymbol{q}_{u,i},$$

where $\boldsymbol{q}_{v,i}$ and $\boldsymbol{q}_{u,i}$ are (unknown) vectors of length at most 1. Thus,

$$\begin{aligned} \tilde{\boldsymbol{v}}[i]\tilde{\boldsymbol{u}}[i] = \tilde{\boldsymbol{v}}^p[i]\tilde{\boldsymbol{u}}^p[i] + k\boldsymbol{d}_v[i]\boldsymbol{q}_{v,i}\tilde{\boldsymbol{u}}^p[i] + k\boldsymbol{d}_u[i]\boldsymbol{q}_{u,i}\tilde{\boldsymbol{v}}^p[i] \\ + k^2\boldsymbol{d}_v[i]\boldsymbol{d}_u[i]\boldsymbol{q}_{v,i}\boldsymbol{q}_{u,i} \end{aligned}$$

Exact maximization/minimization of the above condition is possible but yields formulas that are too unwieldy. We provide slightly weaker upper and lower bounds by treating each term in the above sum separately. Then, applying median monotonicity, we get the following conditions:

$$\begin{aligned} \underset{i}{\text{median}}\{\tilde{\boldsymbol{v}}^p[i]\tilde{\boldsymbol{u}}^p[i] + k\boldsymbol{d}_v[i]\|\tilde{\boldsymbol{u}}^p[i]\| + k\boldsymbol{d}_u[i]\|\tilde{\boldsymbol{v}}^p[i]\| \\ + k^2\boldsymbol{d}_v[i]\boldsymbol{d}_u[i]\} \;\leq\; n\frac{1-\epsilon}{1-\theta}f_{\text{AMS}}(\tilde{\boldsymbol{v}}^p, \tilde{\boldsymbol{u}}^p) \\ \underset{i}{\text{median}}\{\tilde{\boldsymbol{v}}^p[i]\tilde{\boldsymbol{u}}^p[i] - k\boldsymbol{d}_v[i]\|\tilde{\boldsymbol{u}}^p[i]\| - k\boldsymbol{d}_u[i]\|\tilde{\boldsymbol{v}}^p[i]\| \\ - k^2\boldsymbol{d}_v[i]\boldsymbol{d}_u[i]\} \;\geq\; n\frac{1+\epsilon}{1+\theta}f_{\text{AMS}}(\tilde{\boldsymbol{v}}^p, \tilde{\boldsymbol{u}}^p) \end{aligned}$$

Geometric Monitoring for the Median of Bilinear Forms. By dividing both sides by $\pm k^2$ and by virtue of the homogeneity of the median, both conditions take the form:

$$F(\boldsymbol{x}, \boldsymbol{y}) = \underset{i}{\text{median}}\{\boldsymbol{x}[i]\boldsymbol{y}[i] + \boldsymbol{a}[i]\boldsymbol{x}[i] + \boldsymbol{b}[i]\boldsymbol{y}[i] + \boldsymbol{g}[i]\} \leq \zeta,$$

with variables $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^m$, and constants $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{g} \in \mathbb{R}^m$, where $\boldsymbol{a}, \boldsymbol{b}$ have nonnegative components, and $\zeta \in \mathbb{R}$.

In order to apply the geometric method, we need to determine the monochromicity of balls of error vectors (in the combined $2m-$dimensional space. In other words, we need to determine whether a ball defined by $(\boldsymbol{x} - \boldsymbol{c})^2 + (\boldsymbol{y} - \boldsymbol{c}')^2 \leq \rho^2$ (where $\boldsymbol{c}, \boldsymbol{c}'$ are m-vectors) intersects the interior of the inadmissible region $Z = \{\langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathbb{R}^{2m} \mid F(\boldsymbol{x}, \boldsymbol{y}) \geq \zeta\}$. For this problem, essentially the same reasoning applied to the range query case leads us to a bound-of-ball-radius solution.

Given a ball $B(\langle \boldsymbol{c}, \boldsymbol{c}' \rangle, \rho)$, let some vector $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \in Z$ be a nearest neighbor of $\langle \boldsymbol{c}, \boldsymbol{c}' \rangle$. Then,

$$\rho_\zeta^2 = (\boldsymbol{x} - \boldsymbol{c})^2 + (\boldsymbol{y} - \boldsymbol{c}')^2 = \sum_{i=1}^{m}(\boldsymbol{x}[i] - \boldsymbol{c}[i])^2 + (\boldsymbol{y}[i] - \boldsymbol{c}'[i])^2 \quad (6)$$

We compute the (squared) distance ρ_ζ^2 of the center $\langle \boldsymbol{c}, \boldsymbol{c}' \rangle$ to region Z by computing, for each component $i = 1, \ldots, m$ of the median operator in $F()$, a squared coefficient, r_i^2. Each r_i^2 corresponds to a term in the sum of Eqn. (6). Then, the (squared) distance of center $\langle \boldsymbol{c}, \boldsymbol{c}' \rangle$ to the boundary of the inadmissible region Z is obtained by summing the $\lfloor (m+1)/2 \rfloor$ smallest r_i^2.

We now turn to the computation of r_i^2. To keep notation clean, we drop the i-index from the variables: Let c and c' be the i-th coordinates of $\boldsymbol{c}$ and $\boldsymbol{c}'$, respectively (and, similarly, for a, b, g, x, y). If $cc' + ac + bc' + g \geq \zeta$, then $r^2 = 0$. Else, we need to compute x and y which minimize $(x-c)^2 + (y-c')^2$ but set the corresponding component of the median to ζ, that is,

$$r^2 = \inf\{(x - c)^2 + (y - c')^2 \mid xy + ax + by + g = \zeta\}$$

To simplify the problem, first rewrite

$$xy + ax + by + g = (x + b)(y + a) + g - ab$$

By substituting $p = x + b$ and $q = y + a$, we have:

$$r^2 = \inf\{(p-\alpha)^2 + (q-\beta)^2 \,|\, pq = \tau\},$$

where $\alpha = c + b$, $\beta = c' + a$ and $\tau = \zeta + ab - g$.

Now, if $\tau = 0$, then $r^2 = \min(\alpha, \beta)$; else, substitute $q = \tau/p$ to obtain $(p-\alpha)^2 + (\frac{\tau}{p} - \beta)^2$. Taking the derivative equal to 0 reduces to the quartic equation

$$f(p) = p^4 - \alpha p^3 + \tau\beta p - \tau^2 = 0.$$

One of its real roots yields the smallest r^2 (real roots exist since $f(0) < 0$).

3.6 Synchronization Policies for Remote Sites

When the coordinator determines from the geometric method that there is a global violation in the monitoring (that is, the global error vector $\boldsymbol{d}$ is no longer within the admissible region), the coordinator signals some remote sites to flush their updates $\Delta\tilde{\boldsymbol{v}}_j$. While several flushing policies are possible, we describe two alternatives.

Eager Synchronization. This is a simple policy, where all remote sites synchronize concurrently. Each remote site j transmits its current stream updates $\Delta\tilde{\boldsymbol{v}}_j$. After the end of this process, the system reaches a state where, for all j, $\tilde{\boldsymbol{v}}_j = \tilde{\boldsymbol{v}}_j^p$ and thus $\boldsymbol{d}_j = \boldsymbol{d} = 0$. Then, new bounds for the error are computed and broadcast to all remote sites and stream processing begins anew.

Lazy Synchronization. In eager syncronization, even sites whose local updates are few are forced to synchronize. This may be wasteful, and unnecessary; these sites are probably not contributing to the error significantly. A lazy approach would be for the coordinator to syncronize a minimum number of sites, necessary to restore global bounds. Remote sites are ranked in (descending) order of the number of unflushed updates (other choices, such as the distance of $\boldsymbol{d}_j$ to the inadmissible zone, are possible, but our experiments indicated that they do not perform as well). Then, sites are asked to flush sequentially, until, after some flush, the global error $\boldsymbol{d}$ is again restored within the (updated) bounds.

4. EXPERIMENTAL STUDY

In this section, we discuss the empirical evaluation of our techniques using real-life data sets. We start by discussing our testbed and methodology.

Data Sets and Techniques. We use the same real-life data sets as [7] for our experiments. The first data set, **WCup**[3], was drawn from the Internet Traffic Archive and contains HTTP requests sent to the servers hosting the World Cup 1998 web site (totaling approximately 1.35 billion requests over a three-month period). The second data set, **Cdad**[4], comprises SNMP network usage data obtained from CRAWDAD (the Community Resource for Archiving Wireless Data at Dartmouth). It consists of measurements of total network traffic every five minutes over a four month period at a large number of access points (approximately 200) inside a corporate research center (IBM Watson). We tracked the distribution of the `size` attribute from **WCup** and the `shortRet` attribute from **Cdad**, since both these attributes take a very large number of values thus making streaming estimation challenging.

From each data set, we construct a distributed stream for a number of remote sites, by hashing the site field from the data set to the desired number of remote sites in each experiment (**WCup** relates to 26 sites and **Cdad** to 27). Thus, skew in the datasets also appears in our streams. We focus primarily on self-join queries over these streams, as these queries are not parameterized and their sketching error is predictable.

We experimented with our sketch-based geometric monitoring schemes using both the eager and the lazy synchronization policy (denoted by **GM-lazy** and **GM-eager**, respectively). To demonstrate their effectiveness, our methods are contrasted against the sketch-based monitoring technique of [7] (denoted by **CG**). In a nutshell, **CG** is a purely "push-based" monitoring protocol: Each site j continuously tracks the value of its relative delta sketch vector norm $\frac{\|\tilde{\boldsymbol{v}}_j - \tilde{\boldsymbol{v}}_j^p\|}{\|\tilde{\boldsymbol{v}}_j\|}$ checking that it is below an upper bound that depends on ϵ, θ, and the number of sites (determined by the analysis in [7]). When that upper bound is violated, the site simply sends the coordinator its local delta sketch vector (or, the local updates themselves, if smaller), resetting its delta to zero, and resumes its local tracking.

All three methods were implemented using the Fast-AMS sketching technique [7]. Furthermore, since our **GM** schemes are *static* (i.e., do not try to predict the evolution of local/global statistics vectors), we compared them against the static variant of the **CG** technique [7]. As mentioned earlier, the idea of using dynamic *prediction models* (as suggested in [7]) is essentially orthogonal to the ideas in this paper, and prediction models have recently been shown to significantly improve the performance of geometric monitoring as well [17]. We defer the comparison of the dynamic, prediction-based variants of the schemes to the full version of this paper.

Metrics.

Our main focus is on the *communication cost* incurred by our method. We distinguish two parts in the total communication traffic. The first part, *data communication*, comprises messages transmitted from remote sites to the coordinator, when remote sites flush their sketched updates $\Delta\tilde{\boldsymbol{v}}_j$ (or, the list of update records verbatim, if smaller). The second part comprises the *monitoring overhead* of the geometric method, for tracking the global error vector $\boldsymbol{d}$. Studying data communication in isolation, provides a better contrast to the method of [7], since this is the only type of communication performed by that method. In their technique, flushes happen by each remote site when a purely local condition is violated. Our technique, in its effort to delay flushes (improving the effectiveness of sketching) by balancing local errors, incurs additional monitoring overhead. Naturally, we are interested in the cost of this overhead, relative to the gains in data communication costs.

In addition, the separation of these traffic costs makes sense because, in principle, it is possible that the coordinator for the protocols of the geometric method is not co-located in the same machine with the site which collects the global stream updates. For example, if communication channels among remote sites are good, the role of coordinator may be assigned to one of the sites. Furthermore, the traffic patterns of these two types of communication differ significantly: Data communication traffic consists of large messages, travelling from remote sites to the collection site only. Geometric monitoring traffic, on the other hand, consists of small messages, which can easily fit in a single UDP datagram. Upstream traffic (from remote sites to coordinator) is almost equal in volume to downstream traffic (from coordinator to sites). Moreover, downstream traffic consists of identical messages to all sites, and thus it can be implemented by multicast channels. For these reasons, distinguishing between these types of communication can highlight the suitability of each technique in different distributed settings.

Another interesting metric is the *scalability* of our monitoring schemes as the the number of sites collecting the distributed stream becomes larger. In the technique of [7], each site's local condition

[3] `http://ita.ee.lbl.gov/html/contrib/WorldCup.html`

[4] `http://crawdad.cs.dartmouth.edu/meta.php?name=ibm/watson#N100AD`

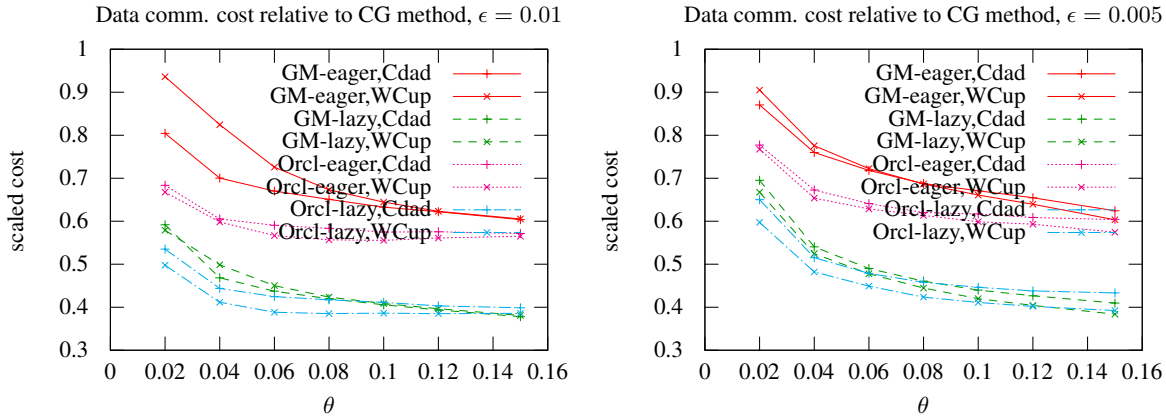


Figure 3: Self-join, data communication costs, as a fraction of the cost of **CG**.

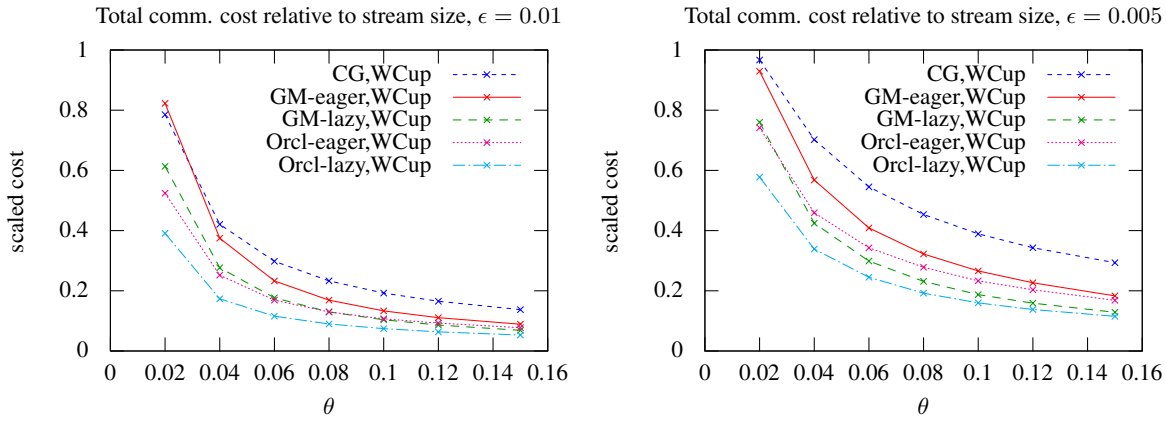


Figure 4: Self-join, total communication cost, as a fraction of stream size (**WCup** dataset).

becomes stricter, causing more frequent flushes—thus, less opportunity for communication reduction via sketching. For our technique, data communication is not increased with more sites, in fact it decreases slightly, as with more nodes there is greater opportunity for balancing. Unfortunately, the overhead of the geometric method increases significantly with the number of sites, rendering our techniques non-scalable to many sites. Although a proper study of scalability is outside the scope of this paper, we present some scalability results, in order to motivate further research.

Results: Communication Cost. The results presented measure the communication cost incurred by our methods. In order to contrast better with the techniques of [7], we do not present absolute cost, by rather the cost scaled relative to the cost of the **CG** method (which has scaled cost 1).

In these experiments, the number of remote sites is 4. We used two different sketch sizes. In both, $\delta = 1/2^7$ (thus, sketches had 7 columns each). The first sketch is built for sketching error $\epsilon = 0.01$ and the second, larger sketch is built for $\epsilon = 0.005$.

Fig. 3 depicts the (relative) data communication cost for our methods, as a function of the total monitoring error θ. It can be seen that both variants of our technique improve significantly upon the cost of the previous technique, with the lazy variant performing much better than the eager one.

A natural question that arises is how much further one can reduce data communication in this framework. To quantify the potential improvement, Fig. 3 depicts also the costs of an *unrealistic oracle-based scheme*, in which data is collected from sites (using either the eager or the lazy flushing policy) *only when a global violation occurs*. As can be seen, the cost of our lazy strategy is quite close to that of the lazy oracle-based one, leaving very little room for improvement. The costs of our eager strategy, while not as close to the eager oracle-based one, are still near. These results validate our claim that monitoring in a lower-dimensional space via the error vector $\boldsymbol{d}$ provides an excellent compromise between monitoring accuracy and monitoring cost.

The total communication costs are depicted in Fig. 4, as a function of stream size. For the technique of [7], this cost is equal to that depicted in Fig. 3, whereas our techniques incur additional cost related to error monitoring. Still, as shown in , this additional cost is well-worth. Our techniques still outperform that of [7], sometimes by up to 35%. Note that, to keep to plot clear, we only show the graphs for the **WCup** dataset. The graphs for the **Cdad** dataset almost coincide.

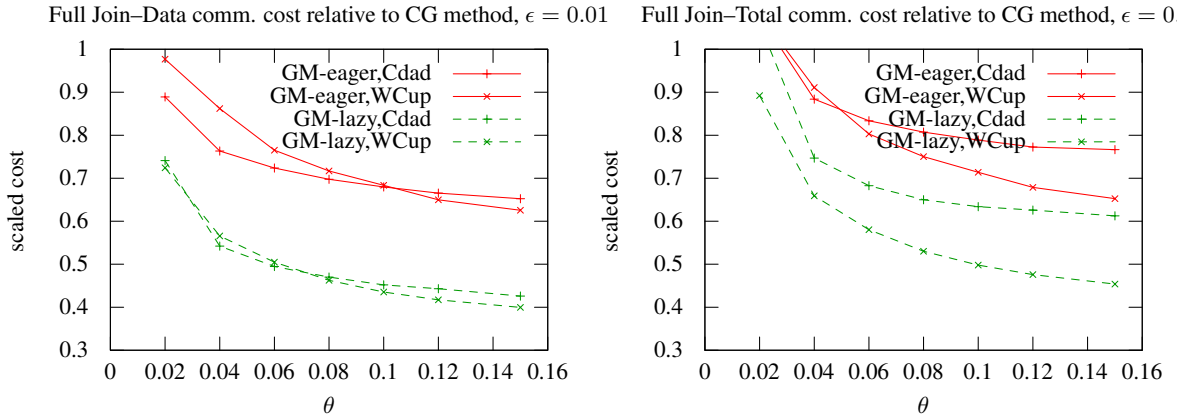


Figure 5: Full-join, data and total communication costs, as a fraction of the cost of **CG**, for sketching error $\epsilon = 0.01$.

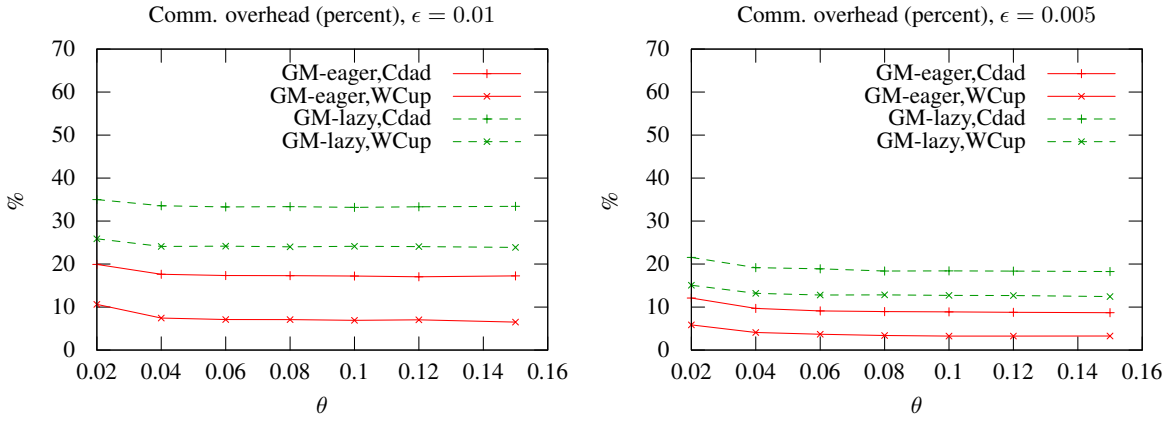


Figure 6: Self-join monitoring overhead as a percent of total cost.

In Fig. 6, the overhead incurred by geometric monitoring is shown as a function of θ. Although the overhead remains relatively constant as θ increases, it is higher for $\epsilon = 0.01$ (where sketches are smaller, imposing less per-flush cost). Interestingly, the lazy method has significantly higher overhead (as a percent) over the eager method. This is due to two factors; first, because the eager method exhibits higher data communication, and second, because intuitively, the lazy method performs more rebalancing, as it delays flushing some sites. Still, the additional overhead is justified for the lazy method, because the savings in data communication are greater.

Other types of monitored queries behave similarly to the self-join query. Due to space restrictions we only present results for our most general full-join query. Fig. 5 presents data and total communication costs for monitoring the join of two streams with sketching error $\epsilon = 0.01$. From each data set we created two streams by splitting the records (**WCup** dataset was split on the `clientID` attribute and **Cdad** was split on the `site` attribute). The join attributes were the same ones tracked in the self-join experiments (`size` and `shortRet` respectively). The same broad effect, of much reduced data communication over **CG** with modest monitoring overhead is observed in this case as well.

4.1 Effect of sketch size

Sketching accuracy ϵ affects communication cost more significantly than the probability bound δ. As sketch size grows with $\log(1/\delta)$, reasonable values of δ (say, from 2^{-11} to 2^{-7} will only affect the sketch size modestly.

Sketching accuracy affects sketch size more strongly, as it increases with $O(1/\epsilon^2)$. When overall accuracy θ is kept constant, the increased accuracy of larger sketches implies that the global sketch will need to be updated less frequently, incurring fewer, albeit larger messages. This implies a trade-off between number of messages and message size.

We now study the trade-off between sketching accuracy (and sketch size) and monitoring accuracy. Fig. 7 depicts data communication cost for overall accuracy $\theta = 0.04$, as the ratio ϵ/θ varies from 0.1 to 0.95. Note that, this cost is normalized (to the total size of all stream updates), and not relative to CG (which is actually also shown in the graph).

The analysis of [7] indicates that their technique performs best for $\epsilon \approx \theta/2$. This is exhibited by our experiments, for our techniques also.

However, the benefit of our technique over that of [7] should be greater when ϵ is smaller than $\theta/2$, that is, when sketches are

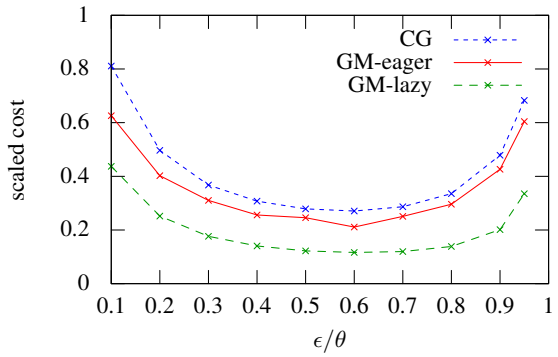


Figure 7: Data comm. cost relative to stream size, over ϵ/θ, for $\theta = 0.04$ (**WCup** dataset).

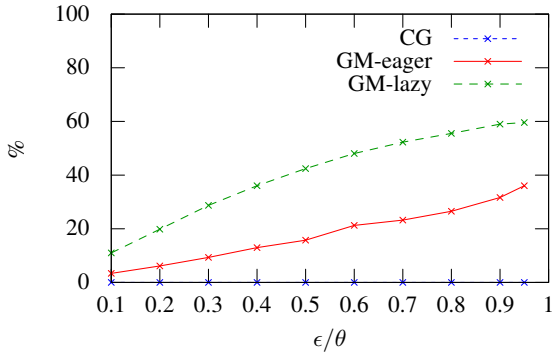


Figure 8: Comm. overhead (percent of total comm.), over ϵ/θ, for $\theta = 0.04$ (**WCup** dataset).

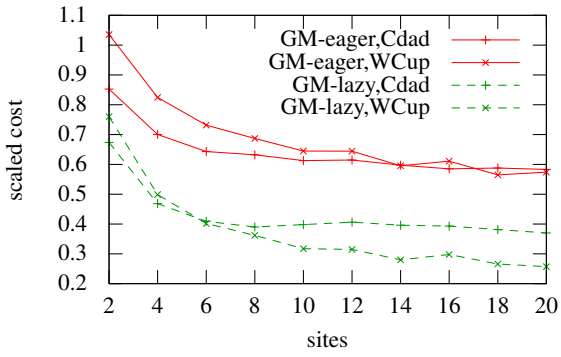


Figure 9: Data comm. cost over number of sites, as a fraction of the cost of **CG**, for $\epsilon = 0.01$ and $\theta = 0.04$.

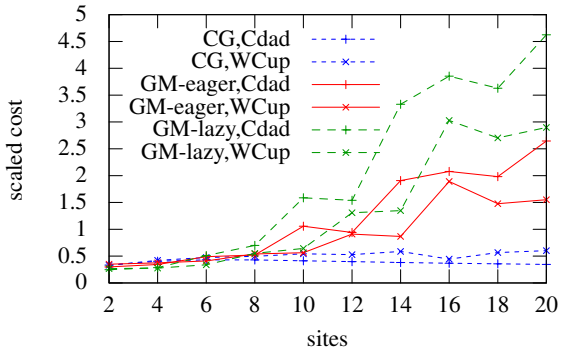


Figure 10: Total comm. over number of sites, as a fraction of stream size, for $\epsilon = 0.01$ and $\theta = 0.04$.

more accurate and larger, because balancing the global error can be done more effectively (and cheaply) when monitoring accuracy is relaxed. Indeed, Fig. 8 shows that the overhead grows significantly as θ approaches ϵ. Again, note that the lazy technique has higher overhead than the eager technique (CG has overhead 0 in this plot).

In practice, it may be desirable that applications utilize sketches of small ϵ, relying on relaxed monitoring precision in order to decrease communication. This is because ϵ cannot be adjusted on-the-fly, once a stream has started to be sketched, whereas adjusting θ can be done on-line very easily.

4.2 Scalability

To measure the behavior of our techniques as the number of sites grows, we conducted experiments where the number of sites monitoring a stream increases, keeping other parameters constant.

Fig. 9 depicts data communication cost (relative to the CG method). As expected, the advantage of our techniques in this aspect of the cost is maintained over the CG method—in fact, there is slow improvement. In fact, the data cost of the lazy synchronization over CG for WorldCup on streams of 20 sites is 4 times less.

Unfortunately, this does not render our techniques scalable, because as the number of sites grows, communication overhead becomes dominant. Fig. 10 depicts the total y(normalized) communication cost. It can be seen that the CG method maintains a 50% saving over the cost of the naive method. In our technique however, communication overhead dominates, to the extent that the total cost becomes 2–4.5 times higher than the cost of the naive method! Notice that the lazy method exhibits again about twice the overhead of the eager method.

The source of the problem seems to be in protocols of the geometric method itself (which we have not adapted in any way in this paper). As the number of sites increases, opportunities for rebalancing among sites increase commensurably. The protocols of the geometric method exhaust every opportunity to ensure that a global violation (triggering flushes) does not occur, without regard for the cost incurred by this rebalancing.

These results indicate that our techniques are only applicable beneficially to applications with a modest number of sites (up to 7). They also indicate an important direction for further research, namely, attempt to capture (at least some of) the benefit in data communication with a scalable rebalancing approach.

5. CONCLUSIONS

The problem addressed in this paper is monitoring of massive, distributed streaming data. The recently proposed geometric method has been combined with AMS sketches towards reducing the communication cost of tracking complex aggregate queries over distributed streams with strict error bounds.

To reduce communication cost, we utilized AMS sketches, similarly to previous work, but in a novel way; we developed a novel

geometric method of dynamic balancing of error between remote sites, improving summarization at the sites before stream data has to be transferred over the network. We showed how to treat three fundamental types of aggregate queries: self-join, range and 2-way join between streams. Finally, we presented extensive empirical results to validate our performance claims for our techniques and demonstrate their practical viability.

Extensions and Future Work. In this paper, we applied the standard geometric method, as it appears in the literature. The techniques we developed exhibit much improved performance compared to previous techniques (particularly that of [7]) but fail to scale performance-wise when the number of remote sites increases. A fruitful problem of future research will be to enhance the standard geometric method, adapting it to the particularities of sketch-based monitoring, in order to improve scalability. Another promising direction for extension is the adoption of dynamic predictive error models. This idea has been shown in [7] to be beneficial to data communication and may also prove useful in reducing the overhead of the geometric method. We also intend to combine our techniques with other types of sketches from the literature and extend their applicability to new types of queries.

Acknowledgments. This work was partially supported by the European Commission under ICT-FP7- LIFT-255951 (Local Inference in Massively Distributed Systems).

6. REFERENCES

[1] N. Alon, P. B. Gibbons, Y. Matias, and M. Szegedy. "Tracking Join and Self-Join Sizes in Limited Storage". In *ACM PODS*, 1999.

[2] N. Alon, Y. Matias, and M. Szegedy. "The Space Complexity of Approximating the Frequency Moments". In *ACM STOC*, 1996.

[3] B. Babcock and C. Olston. "Distributed Top-K Monitoring". In *ACM SIGMOD*, 2003.

[4] M. Charikar, K. Chen, and M. Farach-Colton. "Finding Frequent Items in Data Streams". In *ICALP*, 2002.

[5] D. Chu, A. Deshpande, J. M. Hellerstein, and W. Hong. "Approximate Data Collection in Sensor Networks using Probabilistic Models". In *IEEE ICDE*, 2006.

[6] G. Cormode and M. Garofalakis. Streaming in a connected world: querying and tracking distributed data streams. In *ACM SIGMOD*, 2007.

[7] G. Cormode and M. Garofalakis. "Approximate Continuous Querying of Distributed Streams". *ACM TODS*, 33(2), 2008.

[8] G. Cormode, M. Garofalakis, S. Muthukrishnan, and R. Rastogi. "Holistic Aggregates in a Networked World: Distributed Tracking of Approximate Quantiles". In *ACM SIGMOD*, 2005.

[9] G. Cormode, M. Garofalakis, and D. Sacharidis. "Fast Approximate Wavelet Tracking on Streams". In *EDBT*, 2006.

[10] G. Cormode and S. Muthukrishnan. "What's Hot and What's Not: Tracking Most Frequent Items Dynamically". In *ACM PODS*, 2003.

[11] G. Cormode and S. Muthukrishnan. "An improved data stream summary: The count-min sketch and its applications". In *Jrnl. of Algorithms*, 55(1), 2005.

[12] C. Cranor, T. Johnson, O. Spatscheck, and V. Shkapenyuk. "Gigascope: A Stream Database for Network Applications". In *ACM SIGMOD*, 2003.

[13] A. Das, S. Ganguly, M. Garofalakis, and R. Rastogi. "Distributed Set-Expression Cardinality Estimation". In *VLDB*, 2004.

[14] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong. "Model-Driven Data Acquisition in Sensor Networks". In *VLDB*, 2004.

[15] A. Dobra, M. Garofalakis, J. Gehrke, and R. Rastogi. "Processing Complex Aggregate Queries over Data Streams". In *ACM SIGMOD*, 2002.

[16] S. Ganguly, M. Garofalakis, and R. Rastogi. "Processing Set Expressions over Continuous Update Streams". In *ACM SIGMOD*, 2003.

[17] N. Giatrakos, A. Deligiannakis, M. Garofalakis, I. Sharfman, and A. Schuster. "Prediction-based Geometric Monitoring over Distributed Data Streams". In *ACM SIGMOD*, 2012.

[18] P. B. Gibbons. "Distinct Sampling for Highly-Accurate Answers to Distinct Values Queries and Event Reports". In *VLDB*, 2001.

[19] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss. "How to Summarize the Universe: Dynamic Maintenance of Quantiles". In *VLDB*, 2002.

[20] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss. "One-pass wavelet decomposition of data streams". *IEEE TKDE*, 15(3), 2003.

[21] M. B. Greenwald and S. Khanna. "Space-Efficient Online Computation of Quantile Summaries". In *ACM SIGMOD*, 2001.

[22] M. B. Greenwald and S. Khanna. "Power-Conserving Computation of Order-Statistics over Sensor Networks". In *ACM PODS*, 2004.

[23] A. Jain, E. Y. Chang, and Y.-F. Wang. "Adaptive stream resource management using Kalman Filters". In *ACM SIGMOD*, 2004.

[24] R. Keralapura, G. Cormode, and J. Ramamirtham. "Communication-efficient distributed monitoring of thresholded counts". In *ACM SIGMOD*, 2006.

[25] D. Keren, I. Sharfman, A. Schuster, and A. Livne. "Shape-Sensitive Geometric Monitoring". *IEEE TKDE*, 24(8), 2012.

[26] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. "The Design of an Acquisitional Query Processor for Sensor Networks". In *ACM SIGMOD*, 2003.

[27] A. Manjhi, V. Shkapenyuk, K. Dhamdhere, and C. Olston. "Finding (Recently) Frequent Items in Distributed Data Streams". In *IEEE ICDE*, 2005.

[28] G. S. Manku and R. Motwani. "Approximate Frequency Counts over Data Streams". In *VLDB*, 2002.

[29] NII Shonan Workshop on Large-Scale Distributed Computation, Shonan Village, Japan, January 2012. `http://www.nii.ac.jp/shonan/seminar011/`.

[30] C. Olston, J. Jiang, and J. Widom. "Adaptive Filters for Continuous Queries over Distributed Data Streams". In *ACM SIGMOD*, 2003.

[31] I. Sharfman, A. Schuster, and D. Keren. "A geometric approach to monitoring threshold functions over distributed data streams". In *ACM SIGMOD*, 2006.

[32] N. Thaper, S. Guha, P. Indyk, and N. Koudas. "Dynamic Multidimensional Histograms". In *ACM SIGMOD*, 2002.

Continuous Fragmented Skylines over Distributed Streams

Odysseas Papapetrou and Minos Garofalakis

Technical University of Crete
{papapetrou, minos}@softnet.tuc.gr

***Abstract*—Distributed skyline computation is important for a wide range of application domains, from distributed and web-based systems to ISP-network monitoring and distributed databases. The problem is particularly challenging in dynamic distributed settings, where the goal is to efficiently monitor a continuous skyline query over a collection of distributed streams. All existing work relies on the assumption of a single point of reference for object attributes/dimensions, i.e., objects may be vertically or horizontally partitioned, but the accurate value of each dimension for each object is always maintained by a single site. This assumption is unrealistic for several distributed monitoring applications, where object information is *fragmented* over a set of distributed streams (each monitored by a different site) and needs to be aggregated (e.g., averaged) across several sites. Furthermore, it is frequently useful to define skyline dimensions through complex functions over the aggregated objects, which raises further challenges for dealing with object fragmentation. In this paper, we present the first known distributed approach for *continuous fragmented skylines*, namely distributed monitoring of skylines over *complex functions* of *fragmented* multi-dimensional objects. We also propose several optimizations, including a new technique based on random-walk models for adaptively determining the most efficient monitoring strategy for each object. A thorough experimental study with synthetic and real-life data sets verifies the effectiveness of our approach, demonstrating order-of-magnitude improvements in communication costs compared to the only available centralized solution.**

I. Introduction

Since the introduction of the skyline operator [1], the problem of efficiently constructing skylines in distributed environments (such as, client-server and P2P architectures) has been widely studied (see [2] for a survey). The bulk of this work has typically focused on *one-shot* skyline computation, proposing CPU- and communication-efficient strategies for one-time computation of the set of skyline (i.e., dominating, or, Pareto-optimal) objects across *static*, distributed multi-dimensional object collections. Such one-shot techniques over static data are inadequate for new, rapidly-emerging classes of large-scale event monitoring applications, which need to effectively manage, query, and analyze large collections of *distributed data streams*. Prototypical examples include ISP network-monitoring systems (where usage information from a multitude of monitoring points must be tracked and correlated in order to quickly react to hot spots, floods, failures, and attacks), and wireless sensor networks (where multiple remote sensor measurements must be monitored and analyzed for trends, patterns, intrusions, or other adverse events). Querying in such systems is naturally distributed (i.e., over a collection of remote sites), and also *continuous*, that is, we require real-time monitoring of query answers and events, not merely one-shot responses to sporadic queries.

router	target IP	#packets	vol.
1	121.11.*.*	134	1226
1	110.1.*.*	60	72
2	121.11.*.*	180	1280
2	110.1.*.*	80	100
3	121.11.*.*	160	1301
4	201.7.*.*	627	4874
...	...	...	...

target IP	#packets	vol.
121.11.*.*	158	1269
110.1.*.*	70	86
201.7.*.*	627	4874
117.3.*.*	884	982
...	...	...

Aggregation (average)

target IP	#packets	Var(vol.)	skyline
121.11.*.*	158	1497	YES
110.1.*.*	70	392	NO
201.7.*.*	627	0	NO
117.3.*.*	884	1208	YES
...	...	...	...

Skyline space
Dimensions: #packets, Var(vol.)

Fig. 1. Monitoring an ISP network: (a) the raw-distributed data, (b) the aggregated data, (c) the skyline space.

The problem of continuous skyline maintenance in such dynamic distributed settings has also been addressed in recent work [3]. Still, that work, as well as all existing work in distributed skyline processing assumes *horizontal* or *vertical partitioning* of the data, i.e, each site maintains a subset of the complete object vectors, e.g., [4], or a subset of the dimensions of all objects [5], [6]. As such, all previous algorithms rely on the fundamental assumption that there exists a single site in the network maintaining the accurate value for each object's dimension. This configuration enables each site to independently apply local, arbitrarily complex, filtering techniques on the observed updates, drastically reducing the network resources. This assumption, however, is unrealistic for a number of real-world, distributed monitoring applications, where the vector corresponding to each object is determined *by aggregating* (e.g., averaging) partial vector values fragmented over many sites.

To make matters worse, the skyline dimensions may be defined through (possibly) *complex, non-linear functions* over the aggregated object vectors. For example, an ISP might be interested in monitoring the skyline of the aggregate packet volume and the (non-linear) variance of the packet sizes routed to each subnet through each of the edge routers. Such complex *functional* skyline queries are, of course, particularly challenging in the case of fragmented objects: each site only has its partial view of the object vector values, and, for non-linear functions like variance, it is *impossible* to estimate the value of the function on the global object vector from the function values computed locally [7].

Example 1. Consider the problem of monitoring the network of a large ISP. A typical configuration involves installing monitoring code at the edge routers of the ISP to collect workload statistics over sliding windows for a set of IP addresses served by the ISP. Skyline queries on the data *aggregated* over all edge routers are powerful tools for network administrators, for instance, to quickly identify problematic IP addresses or interesting network events. For example, the skyline of the average (over all routers) number of packets and transfer volume, per target IP (data shown in Fig. 1(b)), helps an administrator to quickly focus on the IPs under attack. The skyline dimensions can even be defined through

complex, non-linear functions on the aggregated data, such as the variance on the workload per IP, collected by the edge routers (Fig. 1(c)) – a key indicator for sites under a DoS attack. Even though the industry standard in routers enables local statistics maintenance, aggregation of the data in order to maintain the skyline space is a challenging task, due to the sheer volume and volatility of the traffic update streams. The problem is only aggravated by the usage of non-linear functions for the definition of the skyline dimensions (e.g., variance), in which case a router observing a local update cannot even predict the direction of the change at the skyline space, e.g., a sudden drop in the transfer volume at one edge router may actually cause an increase of the variance. This calls for a distributed solution for skyline maintenance, where each edge-router monitor can react only to its local updates that potentially invalidate the existing skyline, notifying the central monitor for further analysis.■

Prior Work. Since the proposal of the skyline operator [1], several aspects of skyline computation have been explored, such as, continuous skylines, e.g., [8], [9], [10], functional (or, *dynamic*) skylines [9], subspace skylines [11], and skylines over distributed and P2P networks [2]. Our contribution lies on the intersection of the areas of distributed, functional and continuous skyline queries, with a novel data fragmentation model.

Algorithms for efficiently constructing skylines in P2P and distributed networks have been widely considered in the recent years (see [2] for a recent survey). These algorithms typically rely on three key ideas to reduce the network communication between participants: (1) *Additivity of the Skyline Operator:* The skyline over all remote sites is always a subset of the union of the local skylines computed at each site, e.g., [4]; (2) *Point Filtering:* Representative points, belonging to one or more sites' local skylines, can help other sites effectively reduce their local skylines [5], [6]; and, (3) *Site Filtering:* Compact local site summaries can be used to target neighboring sites that can potentially contribute skyline points [12]. However, at the core of these approaches is the requirement that the value of each dimension for each object is always maintained by a single site, i.e., vertical or horizontal data partitioning, but not fragmentation. Even though both vertical and horizontal data partitioning models hold significant interest for real-life applications (and, in fact, they can also be handled by our work), they are out of the focus of this work. Instead, our contribution is optimized for the case of fragmented data objects, as this arises frequently in a wide range of network-based applications. Furthermore, we focus on continuous skyline queries, and not on one-shot queries.

Perhaps most similar to ours is the work of Zhang et al. for distributed continuous skyline monitoring [3], which relies on installing filters at remote sites to control the updates that need to be sent to the coordinator. The functionality of filters is similar to the one of threshold-crossing queries used in this work. In fact, in the simple case where data is partitioned but not fragmented, and no functions are used for producing the skyline space, our algorithms (without the adaptivity extension) and the one of [3] produce similar types of local constraints, yet, each one following different optimization strategies. Notice however that [3] supports neither fragmented data nor functional skylines, the combination of which is the main focus of this work. Still, some of the ideas of [3], i.e., near-optimal derivation of filters, as well as the sampling-based extension that trades accuracy for performance, can potentially be adapted for the case of fragmented functional skylines, and will be considered in our future work.

Our Contributions. All previous distributed skyline techniques assume either horizontal or vertical partitioning of the database at the sites, which implies that the accurate value of each dimension for each object is known by one of the sites at any time. In this work, we consider the *fundamentally different problem of continuous fragmented skyline queries*, where: (a) each dimension for each object is fragmented over a number of sites, i.e., the actual values of each object are computed by the aggregation (e.g., averaging) of all object's vectors across all sites, and, (b) the skyline space can be further defined through complex arbitrary functions, parameterized by the aggregate values of the objects. Our contributions are summarized as follows:

• We formally define the continuous fragmented skyline problem, and outline the key underlying challenges.

• We present the first known algorithms for efficient processing of continuous fragmented skyline queries, with dimensions defined through possibly complex arbitrary functions over the aggregate vectors. The two algorithms (termed PIVOT and DIRECT) employ different methodologies for *decomposing* the problem to a select set of distributed threshold-crossing queries that are guaranteed to fire when a change in the skyline occurs. These queries can then be monitored efficiently using ideas from the geometric method [13], [7].

• We propose several optimizations that significantly improve the communication efficiency of our fragmented skyline monitoring algorithms. These include techniques for effectively reducing the queries, which can result to substantial network cost reduction, as well as a technique based on random-walk models for adaptively determining the most efficient monitoring strategy for different objects in the system.

• We present a thorough experimental study of our algorithms over both synthetic and real-life data sets. Our experimental results demonstrate substantial performance benefits compared to the (only alternative) centralized solution, which *often exceed two orders of magnitude*.

II. PROBLEM FORMULATION

System Model. We consider a distributed computing environment, comprising a collection of N *remote processing sites* $\mathcal{P} = \{p_1, p_2, \ldots, p_N\}$ and a designated *coordinator site*. Remote sites receive continuous streams of data updates for a collection of n multi-dimensional objects $\mathcal{O} = \{o_1, o_2, \ldots, o_n\}$ that reside in the system (possibly fragmented across multiple sites), while the coordinator is responsible for maintaining answers to continuous user queries posed over the union of remotely-observed streams (across all sites). The (sub)set of sites monitoring object o_j is denoted by $\mathcal{P}(o_j) \subseteq \mathcal{P}$, while $\mathcal{O}(p_i)$ denotes the (sub)set of objects monitored by site p_i. Following earlier work in the area, e.g., [14], [15], [16], our distributed stream-processing model does not allow direct communication between remote sites; instead, a remote site exchanges messages only with the coordinator, providing it with state information on its (locally-observed) streams. Note that such a hierarchical processing model is, in fact, representative of several application domains, including ISP network monitoring and sensor networks.

At time t, the local state of each object o_j at site

p_i is captured by a dynamic d-dimensional *local statistics vector* $\vec{v}(o_j, p_i, t)$. The global state of o_j is defined as the average of o_j's local statistics vectors across all sites in $\mathcal{P}(o_j)$, i.e., the *global statistics vector* $\vec{v}(o_j, t) = \frac{1}{|\mathcal{P}(o_j)|} \sum_{p_i \in \mathcal{P}(o_j)} \vec{v}(o_j, p_i, t)$.[1] (To simplify notation, we omit the explicit dependence on time when referring to the current value of local/global vectors.)

Problem Statement. Our goal is to define effective protocols for continuously monitoring distributed skylines over complex functions of fragmented multi-dimensional objects. More formally, assume that the dimensions of our skyline space are defined through a d'-dimensional *function vector* $\boldsymbol{f} : \mathbb{R}^d \rightarrow \mathbb{R}^{d'}$, where each dimension $\boldsymbol{f}[k](\vec{v}(\cdot))$ is a possibly complex, non-linear, arbitrary function over the original d-dimensional global statistics vectors of our objects. We define the notion of *functional dominance* (or, $\boldsymbol{f}$*-dominance*) over fragmented data objects as follows. (Wlog., the definition assumes that lower values are preferred for the skyline.)

Definition 1 ($\boldsymbol{f}$-dominance)**.** *Let $\vec{v}(o_i)$, $\vec{v}(o_j)$ denote the global statistics vectors of objects o_i and o_j. We say that o_i $\boldsymbol{f}$-*dominates *o_j (denoted as $o_i \prec_{\boldsymbol{f}} o_j$) if and only if $\boldsymbol{f}[k](\vec{v}(o_i)) \leq \boldsymbol{f}[k](\vec{v}(o_j))$ for all $k = 1, \ldots, d'$, and $\exists k \in \{1, \ldots, d'\}$ such that $\boldsymbol{f}[k](\vec{v}(o_i)) < \boldsymbol{f}[k](\vec{v}(o_j))$.*

The $\boldsymbol{f}$*-skyline* of the set of objects $\mathcal{O} = \{o_1, \ldots, o_n\}$ fragmented over the remote sites $\mathcal{P}$ is then simply defined as the subset of objects in $\mathcal{O}$ that are not $\boldsymbol{f}$-dominated by any other object in $\mathcal{O}$. That is, $o \in \mathcal{O}$ belongs in the $\boldsymbol{f}$-skyline if and only if $\nexists o' \in \mathcal{O}$ such that $o' \prec_{\boldsymbol{f}} o$.

We address the challenging task of continuously maintaining the $\boldsymbol{f}$-skyline over a large collection of fragmented multi-dimensional objects $\mathcal{O}$ that are dynamically updated across multiple remote sites $\mathcal{P}$. Our protocols aim to *minimize communication* across remote sites and the coordinator — a critical requirement in large-scale monitoring systems, owing to either network-capacity restrictions (e.g., in ISP monitoring, where the volumes of collected utilization and traffic data can be huge [17]), or power and bandwidth restrictions (e.g., in wireless sensor networks, where communication overhead is the key factor in determining sensor battery life [18]). It is important to note that the centralized solution that ships all updates to a coordinator can easily introduce network, computation, and power bottlenecks, overwhelming the underlying network infrastructure. Similarly, simplistic solutions based on batch or periodic updates to the coordinator can either cause large amounts of unnecessary network traffic (with no real change in the skyline) or fail to react to important transitions in a timely manner (when the update period is large). Most importantly, such techniques cannot offer useful guarantees on the quality of the skyline between updates. Instead, our proposed algorithms are *reactive* (based on the observed stream of object updates) and guarantee the continuous correctness of the $\boldsymbol{f}$-skyline at the coordinator.

Example 2. Building on the ISP monitoring scenario of Example 1, the set of remote processing sites $\mathcal{P}$ includes all edge routers in the ISP network, which collect workload statistics for all target IP addresses (or, subnets) contained in $\mathcal{O}$. Assume that we want to monitor the 2-dimensional skyline shown in Fig. 1(c) (average number of packets and variance of transfer volume across all routers, per IP address). Since our $\boldsymbol{f}$*-skylines* are defined on averaged global vectors, we rewrite the variance function using the average transfer volume and the average squared transfer volume per IP at all routers. In particular, each router p_j maintains a three-dimensional vector $\vec{v}(o_i, p_j)$ for each IP address o_i: $\vec{v}[0](o_i, p_j)$ stores the count of all observed packets destined for o_i and routed through p_j, $\vec{v}[1](o_i, p_j)$ stores the sum of the packet sizes, and $\vec{v}[2](o_i, p_j)$ stores $(\vec{v}[1](o_i, p_j))^2$. The global statistics vector for each IP address o_i is the average of the local statistics vectors over all routers, i.e., $\vec{v}(o_i) = \sum_{p_j \in \mathcal{P}(o_i)} \vec{v}(o_i, p_j)/|\mathcal{P}(o_i)|$. The desired skyline space is then defined by function $\boldsymbol{f}$: $\boldsymbol{f}[0] = \vec{v}[0](o_i)$, i.e., the identity function of the average number of packets for each IP address, and $\boldsymbol{f}[1] = Var(\{\vec{v}(o_i, p_j) | p_j \in \mathcal{P}(o_i)\}) = \sum_{p_j \in \mathcal{P}(o_i)} \frac{\vec{v}[2](o_i, p_j)}{|\mathcal{P}(o_i)|} - \left(\sum_{p_j \in \mathcal{P}(o_i)} \frac{\vec{v}[1](o_i, p_j)}{|\mathcal{P}(o_i)|}\right)^2 = \vec{v}[2](o_i) - (\vec{v}[1](o_i))^2$. ■

Background: The Geometric Method. Our algorithms decompose functional fragmented skyline monitoring to a small set of distributed threshold crossing queries, which can be monitored locally at each site using the geometric method. We now describe the elements of the geometric method needed for this paper. Further details can be found in [7].

The geometric method addresses the basic problem of monitoring *distributed threshold-crossing queries*; that is, monitor whether $f(\vec{v}(o)) < \tau$ or $f(\vec{v}(o)) > \tau$, for any arbitrary, possibly complex, non-linear function $f()$ of a global statistics vector $\vec{v}(o)$ fragmented over N sites, and a fixed threshold τ. The core idea is that, since it is generally impossible to connect the values of $f()$ on the local statistics vectors to the global value $f(\vec{v}(o))$, one can employ geometric arguments to monitor the *domain* (rather than the range) of $f()$.

To initialize the monitoring process, at time t_0 all nodes $p \in \mathcal{P}(o)$ send their local statistics vectors for the object $\vec{v}(o, p, t_0)$ to a coordinator, where the global statistics vector $\vec{v}(o, t_0)$ is computed. This global statistics vector is also called the global estimate vector $\vec{e}(o)$, and is sent to all network nodes. Whenever a node p_j receives a new local value for o, say, at time t, it updates its local statistics vector and checks whether the new value may cause a threshold crossing. For this check, p_j extracts the statistics delta vector $\Delta\vec{v}(o, p_j) = \vec{v}(o, p_j, t) - \vec{v}(o, p_j, t_0)$. The *drift vector* is then defined as $\vec{u}(o, p_j) = \vec{e}(o) + \Delta\vec{v}(o, p_j)$. These vectors can be used to bound the location of the global statistics vector, which, by definition, is guaranteed to lie, within the convex hull formed by the drift vectors of all nodes and $\vec{e}(o)$ [7]. Therefore, by checking that the convex hull does not overlap the inadmissible region (i.e., the region $\{\vec{v} \in \mathbb{R}^2 : f(\vec{v}) > \tau\}$ in Fig. 2) we can guarantee that the threshold has not been violated.

The problem of course is that the drift vectors are distributed across the nodes. Therefore, the global convex hull is unknown to the individual nodes. To transform the global condition into a local constraint, we place a d-dimensional *bounding ball* around each local delta vector, of radius $||\vec{e}(o) - \vec{u}(o, p_j)||/2$ and centered at $(\vec{e}(o) + \vec{u}(o, p_j))/2$ (see Fig. 2). It can be shown that the union of all these balls completely covers the convex hull of the drift vectors [7]. Therefore, as long as the bounding ball constructed individually at each node is *monochromatic*, i.e., it does not overlap with the inadmissible region, the threshold has not been violated, and the node can

[1] More generally, the global statistics vector can be defined through any convex combination of the local statistics vectors.

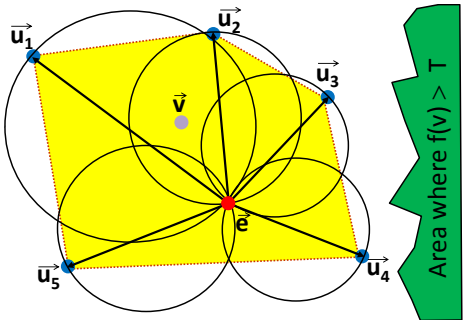


Fig. 2. Estimate vector $\vec{e}$, delta vectors $\Delta\vec{v}(p_i)$ (arrows out of $\vec{e}$), convex hull enclosing the current global vector $\vec{v}$ (dotted outline), and bounding balls $B(\vec{e}, \Delta\vec{v}(p_i))$.

refrain from sending the local update to the coordinator. If this is not the case, we have a *local threshold violation*, and the site communicates its local $\Delta\vec{v}(p_i)$ to the coordinator. The coordinator then initiates a *synchronization process* that typically tries to resolve the local violation by communicating with some of the sites in order to "balance out" the violating $\Delta\vec{v}(p_i)$. Briefly, this process involves collecting the current delta vectors from (a subset of) the sites, and recomputing the minimum and maximum values of $f(\vec{v})$ according to the new, partial, average. In the worst case, the delta vectors from all N sites are collected, leading to an accurate estimate of the current global statistics vector.

In more recent work, Sharfman et al. [13] show that the local bounding balls defined by the geometric method are special cases of a more general theory of *Safe Zones (SZs)*, which can be broadly defined as *convex subsets of the admissible region* of a threshold query. As long as the local drift vectors stay within such a SZ, the global vector is guaranteed (by convexity) to be within the admissible region of the query [13].

III. Monitoring Fragmented Skylines

In this section, we propose two novel algorithms for continuous fragmented skylines: (1) the *Pivot-Based* (PIVOT) algorithm, and (2) the *Direct Monitoring* (DIRECT) algorithm. Both algorithms rely on effectively *decomposing* the continuous fragmented skyline computation into *a collection of threshold-crossing queries*, which can be efficiently monitored at the participating sites using the geometric method. The main difference between PIVOT and DIRECT lies in the details of this decomposition into threshold-crossing conditions. Still, since both algorithms share a common framework, we describe them in parallel, with references to their particularities.

We start with a brief discussion of the high-level distributed-monitoring protocol. Initially, the user configures the continuous skyline query, by first defining the global statistics vector $\vec{v}$, and, second, the possibly complex functions over $\vec{v}$ deriving the skyline dimensions, e.g., variance, L2 norm, or identity function. The system goes through an *initialization phase*, during which the coordinator requests the current local statistics vectors from all sites, and uses them to compute the initial global statistics vectors, the $\boldsymbol{f}$ values for all objects in $\mathcal{O}$, and an initial $\boldsymbol{f}$-skyline, using a standard, centralized algorithm [1]. Then, for each object $o_i \in \mathcal{O}$, the coordinator extracts a set of continuous threshold-crossing queries, denoted as $\mathcal{Q}(o_i)$. While the details of these query sets depend on the employed algorithm (PIVOT or DIRECT), their key property is that they are "safe": *as long as no threshold violation is observed at any site, the skyline is guaranteed not to change*. Finally, the computed global statistics vectors and threshold-crossing queries are shipped to the remote sites observing the corresponding objects, where they are monitored using the geometric method. All updates not violating any threshold query are registered locally at the sites, and only the remaining updates are sent to the coordinator, invoking a synchronization process.

As discussed earlier, a threshold-crossing query focuses on detecting the condition that the value of a function $g()$ over a dynamic vector crosses a fixed threshold value τ. More formally, let t_0 denote the query construction time and let $\vec{v}(t)$ be the dynamic vector; then, using the sign function $\mathrm{sgn}()$, we can define this general threshold-crossing query $Q_{t_0}(g, \vec{v}, \tau)$ as the boolean condition:

$$Q_{t_0}(g, \vec{v}, \tau) \equiv \mathrm{sgn}(g(\vec{v}(t)) - \tau) \neq \mathrm{sgn}(g(\vec{v}(t_0)) - \tau). \quad (1)$$

Both $g()$ and τ can be multi-dimensional, giving rise to a threshold-crossing query that is equivalent to the OR of the boolean conditions across all dimensions; that is, a threshold crossing along *any* of the dimensions causes the query to fire. To keep our descriptions concise, we employ the multi-dimensional form of Query (1) over our skyline function vector $\boldsymbol{f} : \mathbb{R}^d \rightarrow \mathbb{R}^{d'}$ in the ensuing discussion. Obviously, only the subset of relevant dimensions of $\mathbb{R}^d$ are accounted for monitoring each component function $\boldsymbol{f}[k]$ $(k = 1, \ldots, d')$.

In the remainder of this section, we first explain how the two algorithms extract the threshold-crossing queries for each object. Then, we outline the local monitoring and synchronization processes, which are largely common to both algorithms.

A. Threshold-Crossing Query Decomposition

We now discuss the details of decomposing a continuous fragmented skyline into threshold-crossing queries for both PIVOT and DIRECT. PIVOT constructs threshold-crossing queries that pair each object with a set of carefully selected *fixed* pivot points. The purpose of these queries is to ensure that the object remains within a "safe" region, defined by its pivot points in $\mathbb{R}^{d'}$. DIRECT, on the other hand, constructs threshold-crossing queries that correlate each object with a small set of other *(also moving)* objects from $\mathcal{O}$. The purpose of the queries in this case is to detect when the dominance relation between the objects changes.

We describe the query extraction process, starting with a first approach, where each object monitors its relative positioning with respect to all other objects in the system, resulting in $n - 1$ threshold-crossing queries per object in $\mathcal{O}$. We then propose techniques for *drastically reducing the number of queries* (and, therefore, the network resources) required for effective fragmented skyline monitoring.

The PIVOT Algorithm. PIVOT constructs threshold-crossing queries that pair an object $o_i \in \mathcal{O}$ with a set of fixed points in the $\mathbb{R}^{d'}$ space, termed *pivot points*. Specifically, during the initialization phase at time t_0, for each pair of objects $\{o_i, o_j\}$, the coordinator computes the pivot point $\overrightarrow{pp}_{i,j}$ as the midpoint between the $\boldsymbol{f}$-values of o_i and o_j, that is, $\overrightarrow{pp}_{i,j} = \frac{1}{2}(\boldsymbol{f}(\vec{v}(o_i, t_0)) + \boldsymbol{f}(\vec{v}(o_j, t_0)))$. Then, it constructs the two threshold-crossing queries: $Q_{t_0}(\boldsymbol{f}, \vec{v}(o_i), \overrightarrow{pp}_{i,j})$ (installed at sites $\mathcal{P}(o_i)$) and $Q_{t_0}(\boldsymbol{f}, \vec{v}(o_j), \overrightarrow{pp}_{i,j})$ (installed at sites $\mathcal{P}(o_j)$). As an example, Fig. 3(a) depicts a sample data set with five 2-dimensional objects, and Fig. 3(b) shows the same objects in the $\boldsymbol{f}$-skyline space, including the four pivot points

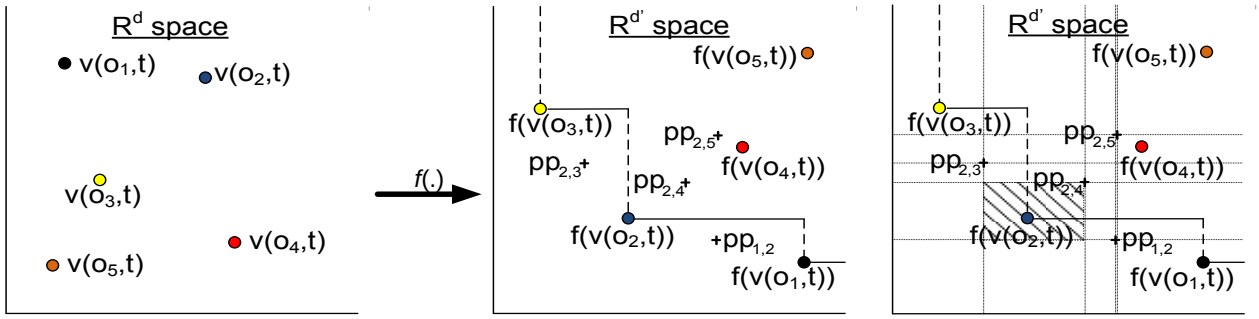


Fig. 3. Pivot-based method: (a) the original $\mathbb{R}^d$ space, (b) the four pivot points for o_2 in the transformed $\mathbb{R}^{d'}$ space, (c) the safe region for o_2.

defined for o_2 with respect to all other objects. A site observing o_2 then has to monitor the following threshold-crossing queries (one per pivot point): $Q_{t_0}(\boldsymbol{f}, \vec{v}(o_2), \overrightarrow{pp}_{1,2})$, $Q_{t_0}(\boldsymbol{f}, \vec{v}(o_2), \overrightarrow{pp}_{3,2})$, $Q_{t_0}(\boldsymbol{f}, \vec{v}(o_2), \overrightarrow{pp}_{4,2})$, and $Q_{t_0}(\boldsymbol{f}, \vec{v}(o_2), \overrightarrow{pp}_{5,2})$.

Consider the geometric interpretation of the PIVOT technique. Each pivot point $\overrightarrow{pp}_{i,j}$ partitions the $\mathbb{R}^{d'}$ space into $3^{d'}$ subspaces: three subspaces for each dimension $k = \{1, \ldots, d'\}$, namely, $\{\vec{x} : \vec{x}[k] < \overrightarrow{pp}_{i,j}[k]\}$, $\{\vec{x} : \vec{x}[k] > \overrightarrow{pp}_{i,j}[k]\}$, and $\{\vec{x} : \vec{x}[k] = \overrightarrow{pp}_{i,j}[k]\}$. The intersection of these $3^{d'}$ subspaces across all threshold-crossing queries for object o_i that contains $\boldsymbol{f}(\vec{v}(o_i))$ effectively defines a *safe region* for o_i; that is, as long as $\boldsymbol{f}(\vec{v}(o_i))$ remains in this region, its relative positioning in the skyline with respect to all other objects in $\mathcal{O}$ remains unchanged. For example, Fig. 3(c) depicts the (shaded) safe region for o_2. Note that the threshold-crossing queries installed at $\mathcal{P}(o_i)$ monitor exactly this safe-region condition for o_i. It is not difficult to prove that this scheme is correct: As long as no PIVOT threshold-crossing query fires, the relative positioning of any object pair in the fragmented skyline (i.e., their relative dominance) remains unchanged, and, thus, the previously-computed skyline remains valid.

The DIRECT Algorithm. Rather than placing fixed pivot points somewhat arbitrarily at the midpoint of two objects, DIRECT *directly monitors* the relative dominance relation across each pair of fragmented objects, based on the vector difference of their $\boldsymbol{f}$-values. Formally, consider any pair of objects $o_i, o_j \in \mathcal{O}$ and, for the time being, assume that both objects are observed at exactly the *same subset of remote sites*, i.e., $\mathcal{P}(o_i) = \mathcal{P}(o_j)$. We define the function-difference vector $\boldsymbol{g}(\vec{v}(o_i)|\vec{v}(o_j)) = \boldsymbol{f}(\vec{v}(o_i)) - \boldsymbol{f}(\vec{v}(o_j))$, where $\vec{v}(o_i)|\vec{v}(o_j)$ denotes the concatenation of the objects' global statistics vectors; thus, $\boldsymbol{g} : \mathbb{R}^{2d} \rightarrow \mathbb{R}^{d'}$. Then, for each such object pair, the coordinator simply constructs the threshold-crossing query $Q_{t_0}(\boldsymbol{g}, \vec{v}(o_i)|\vec{v}(o_j), \vec{0})$ and installs it at all sites in $\mathcal{P}(o_i) = \mathcal{P}(o_j)$ to monitor updates to either o_i or o_j ($\vec{0}$ denotes the all-zero d'-dimensional vector). For instance, in our running example in Fig. 3, the set of DIRECT threshold queries extracted for o_2 is $\mathcal{Q}(o_2) = \{Q_{t_0}(\boldsymbol{g}, \vec{v}(o_2)|\vec{v}(o_j), \vec{0}) : j = 1, 3, 4, 5\}$. Once again, it can be formally shown that, as long as none of the DIRECT threshold-crossing queries fires, the fragmented skyline cannot change.

A number of issues with the DIRECT algorithm are worth noting. First, observe that it effectively *doubles the dimensionality* of the local geometric bounding constraints since it needs to account for updates to both objects. This increased dimensionality typically leads to more frequent local threshold violations and higher communication costs. (This issue can be avoided for certain function types, e.g., when $\boldsymbol{f}$ is linear, but not on the general case.) A second, and perhaps more subtle, issue concerns the extension of DIRECT to handle the general case of object pairs $\{o_i, o_j\}$ that are observed at different subsets of the remote sites (i.e., $\mathcal{P}(o_i) \neq \mathcal{P}(o_j)$), and its effectiveness in such settings. To ensure correctness in this case, the DIRECT threshold query over $\vec{v}(o_i)|\vec{v}(o_j)$ needs to be monitored across all sites in $\mathcal{S} = \mathcal{P}(o_i) \cup \mathcal{P}(o_j)$ (with parts of the local statistics vector zeroed out at sites observing only one of the objects). Furthermore, since the geometric method requires the monitored function(s) $\boldsymbol{g}$ to be defined over the average of the local vectors across all $|\mathcal{S}|$ participating sites, an additional weighting step is needed for the local statistics vectors used in the computation of the $\boldsymbol{f}$-values. The key observation here is that the average global statistics vector $\vec{v}(o_i)$ over all sites in $\mathcal{P}(o_i)$ is equal to the average vector over the super-set $\mathcal{S}$ (assuming zero vectors for sites in $\mathcal{S} - \mathcal{P}(o_i)$) multiplied by $|\mathcal{S}|/|\mathcal{P}(o_i)|$. Therefore, we can apply the geometric method assuming that o_i is monitored by all sites in $\mathcal{S}$, by simply scaling each of its local statistics vectors by $|\mathcal{S}|/|\mathcal{P}(o_i)|$ (and, similarly for o_j). This scaling, however, has the adverse effect of increasing the radius of the local bounding ball for the object, thereby increasing the number of local violations. In fact, it can be formally proved that the performance of DIRECT is *worse* than that of PIVOT under certain such settings.

Theorem 1. *Monitoring the* DIRECT *threshold-crossing query* $Q_{t_0}(\boldsymbol{g}, \vec{v}(o_i)|\vec{v}(o_j), \vec{0})$ *for object* o_i *at sites* $\mathcal{S} = \mathcal{P}(o_i) \cup \mathcal{P}(o_j)$ *is provably less communication-efficient than monitoring the corresponding* PIVOT *threshold query* $Q_{t_0}(\boldsymbol{f}, \vec{v}(o_i), \overrightarrow{pp}_{i,j})$, *when all functions in* $\boldsymbol{f}$ *are linear, and* $\frac{|\mathcal{S}|}{|\mathcal{P}(o_i)|} > 2$.

All proofs are deferred to the extended version of the paper. Similar results can also be shown for other types of functions. Note that the cardinality ratio condition $\frac{|\mathcal{S}|}{|\mathcal{P}(o_i)|} > 2$ is easily satisfied when objects are monitored by distinct subsets of sites; furthermore, some of the optimizations discussed later in this section (e.g., grouping) further exacerbate this problem for the DIRECT algorithm.

B. Reducing the Number of Queries

The total number of threshold crossing queries influences the network cost of PIVOT and DIRECT, since: (a) all queries need to be sent to the sites, during initialization and after threshold crossings, and, (b) a higher number of queries can obviously lead to tighter safe regions and more frequent threshold crossings. To reduce network cost, we need to extract a sufficient subset of queries that can still guarantee the correctness of the skyline. In this section, we show how the total number of queries can be substantially reduced (from quadratic to linear on the number of objects). It is important to note that this re-

duction comes without increasing the tightness of the threshold queries, which would have the adverse effect of increasing the frequency of threshold violations and the induced network cost. In fact, the safe regions are, for most objects, substantially relaxed. To avoid repetition, the ensuing discussion focuses primarily on PIVOT. The same optimizations can be adapted for DIRECT in a reasonably straightforward manner.

Eliminating Redundant Threshold Queries. A crucial observation is that not all changes in pairwise dominance relations between objects in $\mathcal{O}$ are important for skyline monitoring. For example, the skyline will not change if o_4 (Fig. 3(b)) is updated such that it no longer $\boldsymbol{f}$-dominates o_5. In fact, there are only two types of threshold-crossing queries that can signify a change in the skyline: (1) Queries monitoring the *domination of a non-skyline object by a skyline object*, where a violation may indicate the entry of a new object in the skyline; and, (2) Queries monitoring the *dominance (i.e., Pareto optimality) of a skyline object*, where a violation may indicate the removal of an object from the skyline. All other queries are essentially redundant and can be safely dropped.

(1) Queries Monitoring Domination of a Non-Skyline Object: The key observation here is that a non-skyline object cannot enter the skyline as long as it is $\boldsymbol{f}$-dominated by at least one skyline object. Thus, for any given non-skyline object o_i, it suffices to monitor a single threshold-crossing query between o_i and a skyline object o_j that $\boldsymbol{f}$-dominates o_i. Having no knowledge on the distribution of future updates, the best threshold condition to monitor is the one that maximizes the minimum distance (slack) between o_i and the resulting pivot point $\overrightarrow{pp}_{i,j}$ along all d' dimensions; that is, we select the skyline object o_j that $\boldsymbol{f}$-dominates o_i and maximizes $\min_{\ell=1}^{d'}\{\boldsymbol{f}[\ell](\vec{v}(o_i)) - \boldsymbol{f}[\ell](\overrightarrow{pp}_{i,j})\}$. In our Fig. 3(b) example, this gives rise to threshold queries for the object pairs $\{o_2, o_4\}$ and $\{o_2, o_5\}$.

(2) Queries Monitoring Dominance of a Skyline Object: A skyline object o_i may exit the skyline only when some other skyline object o_j moves to $\boldsymbol{f}$-dominate o_i. (A non-skyline object can cause the removal of a skyline object only after itself enters the skyline, thereby causing another threshold query of the previous class to fire.) Furthermore, not all pairs of skyline objects need to be monitored, since some skyline objects impose tighter threshold constraints than others, and will always be violated first. For example, o_1 cannot move to dominate o_3 without first crossing its threshold query with o_2. Specifically, for any skyline object o_i, the coordinator constructs a threshold-crossing query between o_i and all other skyline objects whose $\boldsymbol{f}$ values *immediately* precede or follow $\boldsymbol{f}(o_i)$ along any dimension of the $\mathbb{R}^{d'}$ space. In our Fig. 3(b) example, this gives rise to threshold queries for the pairs $\{o_2, o_3\}$ and $\{o_2, o_1\}$.

Using the above ideas, the total number of threshold-crossing queries in the system is effectively reduced from $\Theta(n^2)$ to (at most) $2(n + s(d' - 1))$, where s denotes the size of the skyline (and, typically, $s << n$).

Grouping of Pivot Points. Even after eliminating redundant queries, skyline objects with dense dominance regions may end up participating in a large number of threshold-crossing queries with different pivot points. This translates to high transfer volume for sending all these threshold queries to sites, both during initialization and after threshold crossings. To further reduce PIVOT's resource requirements, the coordinator forms *groups of pivot points* for each skyline object o_i, and replaces each group with a single "composite" pivot point that imposes equivalent threshold constraints on o_i. Threshold queries are then constructed based on the computed composite pivot points, which are much fewer than the original ones, and typically also enable enlarging the safe-zones for the non-skyline objects.

Precisely, the pivot points of each skyline object o_i are grouped based on their relative positioning with respect to $\boldsymbol{f}(\vec{v}(o_i))$ in all d' dimensions. Any two pivot points $\overrightarrow{pp}_{i,j}$ and $\overrightarrow{pp}_{i,k}$ are grouped together if their $\boldsymbol{f}$ values are on the same side of $\boldsymbol{f}(\vec{v}(o_i))$ in all d' dimensions, or, more formally, if $\text{sgn}(\boldsymbol{f}[\ell](\vec{v}(o_i)) - \overrightarrow{pp}_{i,j}[\ell]) = \text{sgn}(\boldsymbol{f}[\ell](\vec{v}(o_i)) - \overrightarrow{pp}_{i,k}[\ell])$ for all $\ell = 1, \ldots, d'$. All pivot points belonging in the same group $G = \{\overrightarrow{pp}_{i,j}, \overrightarrow{pp}_{i,k}, \ldots\}$ are then replaced by a composite pivot point $\overrightarrow{pp}_{i,G}$, defined as follows:

$$\overrightarrow{pp}_{i,G}[\ell] = \begin{cases} \min_{\overrightarrow{pp} \in G} \overrightarrow{pp}[\ell] & \text{if } \min_{\overrightarrow{pp} \in G}(\overrightarrow{pp}[\ell]) \geq \boldsymbol{f}[\ell](\vec{v}(o_i, t)) \\ \max_{\overrightarrow{pp} \in G} \overrightarrow{pp}[\ell] & \text{if } \max_{\overrightarrow{pp} \in G}(\overrightarrow{pp}[\ell]) < \boldsymbol{f}[\ell](\vec{v}(o_i, t)) \end{cases}$$

for $\ell = 1, \ldots, d'$.

By construction, this composite pivot point imposes the same restrictions on $\boldsymbol{f}(\vec{v}(o_i, t))$ as the collection of pivot points in G. Furthermore, all pivot points in G for objects $\{o_j, o_k, \ldots\}$ are also replaced by the composite pivot point $\overrightarrow{pp}_{i,G}$, which can result in additional slack for these objects, yet without introducing errors. In the example of Fig. 3(c), grouping replaces $G = \{\overrightarrow{pp}_{2,4}, \overrightarrow{pp}_{2,5}\}$ with a single composite pivot point $\overrightarrow{pp}_{2,G}$ that coincides with $\overrightarrow{pp}_{2,4}$, which actually gives additional slack to o_5.

Combining the elimination of redundant queries with pivot-point grouping results in a maximum of $2d' + 1$ threshold-crossing queries for each skyline object ($2d'$ pivot points for its neighboring skyline objects and one composite pivot point for all objects in its dominance region). Each non-skyline object only needs to monitor a single threshold query. Thus, the total number of threshold queries in the system is effectively reduced to (at most) $n + 2sd'$, that is, $O(n)$.

The following theorem summarizes the correctness guarantees offered by the resulting queries.

Theorem 2. *The extracted threshold queries are sufficient for accurate fragmented skyline monitoring, i.e., as long as no threshold violation occurs, the fragmented skyline is guaranteed to stay the same. They are also* minimal*, in the sense that omitting any of the queries breaks the correctness guarantees.*

C. Local Monitoring

Note that the threshold-crossing queries produced by our decomposition do not directly translate to local monitoring conditions, since these are defined over the aggregate object values (the global statistics vectors). However, nodes can exploit the geometric method to efficiently monitor these threshold queries without imposing centralization of all updates. Briefly, a node receiving an update for an object o forms the bounding ball (see Section II), and tests for monochromicity w.r.t. all threshold queries. This test is performed by finding the minimum and maximum value of the monitored function inside the bounding ball. If both values are on the same side of the threshold, the update is safe, i.e., it does not violate the threshold query and cannot invalidate the skyline. Otherwise, the site notifies the coordinator and a synchronization process is initiated.

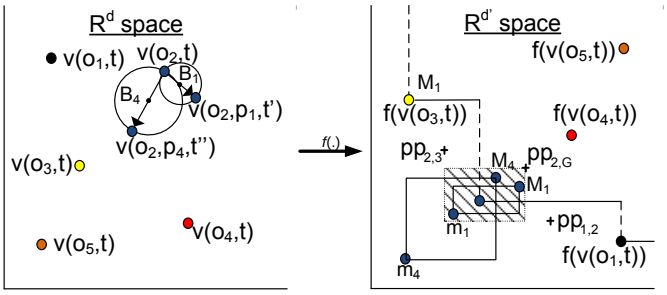


Fig. 4. Handling updates with the pivot-based method: (a) constructing the balls in the $\mathbb{R}^d$ space, (b) constructing the boxes in the $\mathbb{R}^{d'}$ space.

An example is depicted in Fig. 4, with two sites (p_1 and p_4) receiving updates for the same object o_2, and constructing the local bounding balls, B_1 and B_4 (Fig. 4(a)). Let $\overrightarrow{m}_1/\overrightarrow{M}_1$ denote the minimum and maximum values of $\boldsymbol{f}$ inside B_1, as computed at p_1, and $\overrightarrow{m}_4/\overrightarrow{M}_4$ the ones inside B_4. Since both $\overrightarrow{m}_1$ and $\overrightarrow{M}_1$ remain within the safe region defined by the threshold queries in $\mathbb{R}^{d'}$ (Fig. 4(b)), the update at p_1 is safe and registered locally at p_1. On the other hand, the update at p_4 is unsafe, since $\overrightarrow{m}_4$ violates the query corresponding to $\overrightarrow{pp}_{2,3}$. Thus, p_4 notifies the coordinator of its current local vector, initiating a synchronization process.

The local monitoring algorithm also makes use of the more general *safe zone* mechanism for testing local violations (Section II). Safe zones can be defined for various classes of monitoring functions; for instance, using hyperplanes for linear functions. In our work, we employ safe zones whenever applicable, as these can drastically reduce the number of local violations and, consequently, the required network resources. More details on the definition and construction of safe zones can be found in [19], [13].

D. Synchronization

Consider a PIVOT threshold-crossing query Q monitoring the relative dominance relation of the object pair $\{o_i, o_j\}$ that raises a local violation due to an update of object o_i at some site in $\mathcal{P}(o_i)$. As discussed briefly in Section II, the coordinator initiates a *balancing process* to try to resolve the violation on o_i. If that balancing fails to resolve the local threshold violation even after contacting all sites, the coordinator computes the updated $\vec{v}(o_i)$ out of the collected local statistics. Then, if the dominance relation between o_i and o_j has not changed, the coordinator only needs to recompute the pivot point for Q, and send it to $\mathcal{P}(o_i)$ and $\mathcal{P}(o_j)$. Otherwise, it updates the skyline according to the updated global statistics (using a centralized continuous skyline algorithm to reduce computation cost [8]), and recomputes *only* the threshold queries involving at least one of the two objects and a skyline object, according to the process described in Section III-B. All updated and new threshold queries are then sent to the sites monitoring the corresponding objects, and the monitoring protocol continues. The above process relies on cached global statistics vectors of some objects (i.e., o_j), to extract the new threshold queries. It is therefore possible that the local statistics vectors at some of the sites cause immediate threshold violations with the updated threshold queries. In such cases, synchronization is invoked recursively, until no more threshold violations are observed.

An important optimization here is *lazy query updating*, which postpones the replacement of all queries that are still valid, even if the participating objects have changed their skyline status. For example, when an object is removed from the skyline but still dominates a large number of objects, the coordinator need not update the corresponding query. Instead, sites continue monitoring the query, until an update causes a threshold crossing. In our experiments with real-world data sets, this optimization has been shown to enable substantial network savings.

A slight modification is required at the synchronization process for the DIRECT algorithm: Since DIRECT threshold queries are defined on pairs of objects, balancing is always performed for both objects. The rest of the synchronization scheme remains the same.

IV. THE ADAPTIVE METHOD

The geometric method (and, in effect, the proposed algorithms) relies on the existence of a small slack (i.e., freedom to move) for each object, for effectively filtering local updates. In extreme situations, however, the constructed threshold queries may be too tight, leaving little slack for updates and causing frequent synchronizations (e.g., when two objects are very close in $\mathbb{R}^{d'}$). Depending on the frequency and cost of these synchronizations, it may be more network-efficient to identify such costly threshold queries, and exclude their corresponding objects from the geometric monitoring protocol. All updates for these objects are then *directly streamed* to the coordinator, thereby introducing a cost for sending the updates, but eliminating the need for costly synchronizations.

In this section, we propose an adaptive module for identifying such objects. The module is executed by the coordinator each time any object causes a threshold violation, and operates by estimating and comparing the communication cost for keeping the object under geometric monitoring versus directly streaming all its updates. Note that this module is only applicable to PIVOT; since DIRECT always considers objects in pairs, the dependencies across objects make it impossible to exclude an individual object from geometric monitoring.

With $\mathcal{A}_{gm}$ and $\mathcal{A}_{st}$ we denote the two alternative monitoring schemes, the first based on the geometric method (i.e., PIVOT) and the second based on streaming updates. We distinguish two types of threshold violations: (a) *true threshold violations*, where the global statistics vector of the object has changed sufficiently to cause a threshold violation in the global query; and, (b) *false-positive threshold violations*, where only a local statistics vector of the object causes a violation that can be resolved with balancing, without changing the threshold query. Note that both $\mathcal{A}_{gm}$ and $\mathcal{A}_{st}$ will incur the same true threshold violations for the same stream, but only $\mathcal{A}_{gm}$ will run into false-positive violations.

To decide between $\mathcal{A}_{gm}$ and $\mathcal{A}_{st}$ for a given object o, the coordinator needs to predict the network cost required by each scheme for monitoring o until the next true threshold violation for o. Let t denote the time of the last global synchronization for o, and t' the time of the next true threshold violation caused by o. For illustration purposes only, assume that the coordinator has full knowledge of the updates arriving between t and t'. Let $N_{t'}$ denote the number of updates arriving for o in this time range, $N_{fp}(o)$ the number of false positive threshold violations, and $C_{fp}(o)$ the average cost of resolving each such violation. Then, the cost for monitoring o with $\mathcal{A}_{gm}$ is $\mathcal{C}_{gm} = C_{fp}(o) \times N_{fp}(o)$ (for resolving all false positive threshold violations), whereas the cost for $\mathcal{A}_{st}$ is simply $\mathcal{C}_{st} = c \times N_{t'}$, where c is the cost of a single

Algorithm 1: Adaptivity Estimation Algorithm

```
// Executed at the coordinator
function Estimate N_{t'}()
begin
    n ← 1;
    TC ← false; // true when I find a threshold crossing
    repeat
        TC ← probe(n); // check for threshold crossing
        if (!TC) then n ← 2n;
    until (TC);
    // I know that n/2 < N_{t'} ≤ n
    maxN ← n;  minN ← n/2;
    while (maxN-minN>1) do
        n = minN + (maxN-minN)/2;
        if (probe(n)) then maxN ← n; else minN ← n;
    end
    return n;
end
// Checks for threshold crossing, for a given n
function probe(int n)
begin
    for (dim = 1 → d) do
        // Compute left/right bounds for prob 0.5 (see Eqn.3)
        l[dim] ← computeLeftBound(n, 0.5);
        r[dim] ← computeRightBound(n, 0.5);
    end
    // sampleN determines the sampling resolution
    for (int sample=0 → sampleN) do
        p⃗ ←UniformSampleFromHyperCube(l, r);
        // Compute prob to reach p⃗ after n steps (see Eqn.3)
        pr_p ←probToReachPoint(p⃗, v⃗(o, t), n);
        if (pr_p ≥ 0.5 and f(p⃗) causes threshold
        crossing) then return true;
    end
    return false;
end
```

update message. The coordinator chooses the algorithm with the smallest network cost, and notifies the sites monitoring o to switch to that algorithm.

A. Estimating Threshold Violation Costs

Clearly, in a real-world situation, at time $t < t'$, the coordinator cannot know the accurate values of $N_{fp}(o)$, $C_{fp}(o)$, and $N_{t'}(o)$, since these concern future updates in the stream. It can, however, estimate these values through extrapolation on recently observed updates for o. In the remainder of this section, we first describe mathematical models for obtaining these estimates, and then present the detailed algorithm that exploits these models to predict the cost of the geometric and streaming schemes.

Mathematical Preliminaries. To estimate the resolution cost $C_{fp}(o)$, the coordinator employs the average cost for resolving false positive threshold violations over the last ℓ observed violations, where ℓ is a small number, e.g., 10. Estimating $N_{t'}$ and N_{fp} requires a prediction model for future object updates. In the absence of knowledge on the distribution characterizing the updates, we employ a *random walk model* to capture the behavior of object updates. Precisely, the changes in both the global and local statistics vectors for each object o are modeled as d-dimensional random walks. The step length for these walks is determined empirically, by averaging the magnitudes of change for all observed updates of o across all sites.

Let vector $\vec{s}(o)$ denote the average of change magnitudes for the updates observed by all sites in $\mathcal{P}(o)$. According to the random walk model [20], the global statistics vector of o follows a d-dimensional binomial distribution, with variance $\sigma_g[i]^2 = \vec{s}(o)[i]^2 \sum_{p\in\mathcal{P}(o)} n_p$, where n_p denotes the number of updates received for object o at site p since time t. A similar random walk is used to model the local statistics vector of o at each site $p \in \mathcal{P}(o)$: To simplify computation, rather than using per-site update statistics, our model employs the single aggregate change vector $|\mathcal{P}(o)| \times \vec{s}(o)$ for all sites in $\mathcal{P}(o)$ (recall that the global statistics vector is the *average* of the $|\mathcal{P}(o)|$ local statistics vectors). Then, the probability distribution describing the local statistics vector of object o at p is a d-dimensional binomial distribution with variance $\sigma_l[i]^2 = (|\mathcal{P}(o)| \times \vec{s}(o)[i])^2 n_p$ [20].

Through one-sided Chebyshev inequalities we can probabilistically bound the location of the global and local statistics vectors of each object, after n_p updates. Precisely, for any dimension i and any point $l < \vec{v}(o,t)[i]$, the probability of $\vec{v}(o,t')[i]$ crossing l along dimension i is $Pr[\vec{v}(o,t')[i] < l] \leq \frac{\sigma_g[i]^2}{\sigma_g[i]^2+(\vec{v}(o,t)[i]-l)^2}$. Therefore, the value of l satisfying $Pr[\vec{v}(o,t')[i] < l] > pr$ for a desired minimum probability pr is:

$$l \geq \vec{v}(o,t)[i] - \sigma_g[i]\sqrt{(1-pr)/pr} \tag{2}$$

Similar inequalities hold for $Pr[\vec{v}(o,t')[i] > r]$ for all $r > \vec{v}(o,t)[i]$, as well as for the probability of a local statistics vector dimension being less than l or greater than r.

Estimation Algorithm. Alg. 1 exploits the above-derived probabilistic inequalities to estimate $N_{t'}(o)$ and N_{fp}. Starting from $n = 1$ and using a combination of doubling and binary search, we find the maximum number of steps n, such that any point $\vec{p}$ reachable from $\vec{v}(o,t)$ with probability higher than 0.5, does not cause a threshold violation. Formally, let $\mathcal{V}_n = \{\vec{p}_1, \vec{p}_2, \ldots\}$ denote the (possibly infinite) set of points, such that any $\vec{p} \in \mathcal{V}_n$ satisfies the following condition after n updates, for all dimensions $i = 1, \ldots, d$:

$$\prod_{i=1}^{d} pr_i \geq 0.5, \text{with } pr_i = \begin{cases} Pr[\vec{v}(o,t')[i] < \vec{p}[i]], & \text{if } \vec{p}[i] < \vec{v}(o,t)[i] \\ Pr[\vec{v}(o,t')[i] > \vec{p}[i]], & \text{if } \vec{p}[i] > \vec{v}(o,t)[i] \end{cases}$$

The significance of $\mathcal{V}_n$ is that each of the points in the set is likely to be reached from $\vec{v}(o,t)$ after n updates, i.e., with probability ≥ 0.5. $N_{t'}(o)$ is set to the maximum value n, such that for all points $\vec{p} \in \mathcal{V}_n$, $\boldsymbol{f}(\vec{p})$ does not cause a threshold violation for any of the threshold queries for object o. To test the above constraints efficiently, the points $\vec{p}$ are uniformly sampled (using a superimposed grid) over the range defined by l and r, as these are computed per dimension for probability 0.5, (e.g., using Equation 2). The number of repetitions required to estimate $N_{t'}(o)$, is logarithmic in $N_{t'}(o)$, and linear in the resolution of the grid.

The same process is used to predict the number of steps for the next false positive threshold violation, required for estimating the total number of false positive threshold violations N_{fp}. Then, using the described formulas for $\mathcal{C}_{gm}$ and $\mathcal{C}_{st}$, we compute the expected cost for $\mathcal{A}_{gm}$ and $\mathcal{A}_{st}$ and select the most efficient monitoring scheme.

Due to sampling and extrapolation, the above process may fail to detect some local or global threshold violations. A sudden change in stream characteristics may also result in an overestimate or underestimate of the values of N_{fp} or $N_{t'}$. Such inaccuracies, however, do not introduce errors in the skyline; the only possible negative consequence is that the

Data sets	
Name	**synthetic**, WEATHER, MOVIES
Correlation of dim.	**Independent**, Correlated, Anti-correlated
Max. relative change	0.01, **0.02**, 0.04, 0.08, 0.16
# objects	257, 1000, **2000**, 3000, 4000, 5000, 10681
Experimental Configuration	
Function	**Linear**, Norm, L2 distance, Variance
Dimensions	2, 3, 4, 5
# sites	200, 500, **1000**, 1500, 2000, 2500, 5423

TABLE I. EXPERIMENTAL PARAMETERS (DEFAULT VALUE IS BOLD).

adaptive module selects a suboptimal monitoring algorithm for an object, thereby increasing the monitoring cost.

V. EXPERIMENTAL EVALUATION

Our experiments were focused on evaluating the network efficiency and scalability of PIVOT and DIRECT, as well as on providing guidelines for selecting the best algorithm for each configuration. Network efficiency was measured in number of messages and transfer volume. Since both algorithms guarantee maintaining the exact skyline, their accuracy was always 100% and is therefore not presented in the results.

As a baseline, we have used the only available alternative for continuous fragmented functional skylines, which streams the updates to a central node (only the updates that actually alter the local statistics vector of an object were considered). In the following, the baseline will be denoted as CENTR, due to its central nature. Unless noted differently, the results for PIVOT and DIRECT correspond to the fully-fledged variants of the algorithms, i.e., with query reduction, grouping, and the adaptivity extension. In the vast majority of the experiments, each of these extensions was shown to improve the performance of the algorithms, typically reducing the communication overhead by a factor of two.

Data sets. We have used two publicly available real-world data sets, a massive weather-related data set (denoted with WEATHER), and the Movielens movie ratings data set (MOVIES). Furthermore, a set of massive synthetic data streams generated with Kossmann's data generator [1] – the standard generator for evaluation of skyline algorithms – allowed us to study the behavior of the algorithms under different data characteristics. Since Kossmann's generator creates only static data sets, updates were simulated by randomly selecting a site p_i and an object o_j at each step, and shifting the local value of the object to a value uniformly selected within the range $[(1-\text{maxCh})\vec{v}(o_j,p_i,t),\ (1+\text{maxCh})\vec{v}(o_j,p_i,t)]$, with maxCh denoting the *maximum relative change* chosen for the experiment. Unless otherwise specified, the reported results correspond to the average cost over 40 executions, with streams of 10 million updates.

Monitored functions. The proposed algorithms were evaluated using both linear and non-linear functions. For linear functions, we will report results for the identity function of the *average object values*, i.e., $\boldsymbol{f}(\vec{v}(o,t)) = \vec{v}(o,t)$, which enables us to directly observe the influence of the data characteristics to the performance of the algorithms. For non-linear functions, we considered three frequently used functions, variance of a dimension across all sites, euclidean norm on two dimensions, and L2 distance on four dimensions.

Table I summarizes the configuration parameters varied in our experiments, and the default values for each parameter. To avoid repetition, in our discussion we will be noting only the parameters with values different from the default values.

A. Influence of the data characteristics

We first investigate the influence of the following data characteristics to the performance of the proposed algorithms:

- **Correlation between dimensions:** *correlated* (e.g., price Vs performance for computers), *anti-correlated* (price Vs mileage for used cars), or *independent* (shipping cost Vs item price).
- **Maximum change:** We consider values from 1% to 16%.

For this first set of experiments, we have generated different synthetic streams of 2000 two-dimensional objects, varying the properties described earlier. The network was configured such that all objects were monitored by all sites. In order to maintain the stream properties also in the skyline space, $\boldsymbol{f}[0]$ and $\boldsymbol{f}[1]$ were set to be the identity functions on the two dimensions of the objects. The total cost of CENTR in these experiments was always 10 million messages totaling 305 Mbytes.

Correlation between dimensions. Fig. 5(a) plots the transfer volume required by PIVOT and DIRECT, as measured at regular stream intervals. Notice that, for illustration purposes, Y axis is interrupted at $y = 0.0065$. Clearly, both PIVOT and DIRECT enable substantial savings for all data sets. In particular, both algorithms require two to three orders of magnitude less transfer volume compared to CENTR on the data sets with independent and correlated dimensions. The data set with anti-correlated dimensions is more challenging for the two algorithms, since, due to this anti-correlation, skyline objects end up to be close to each other leading to frequent skyline updates. Nevertheless, even for this data set, both PIVOT and DIRECT still enable around 70% reduction of the network cost compared to CENTR. Similar observations are derived by the comparison of the three algorithms in terms of number of messages (Fig. 5(b)).

Also note that DIRECT is more efficient than PIVOT for the streams with the correlated and independent dimensions, both with respect to number of messages and transfer volume. This is not the case for the anti-correlated data set, where PIVOT substantially outperforms DIRECT in terms of number of messages. The reason for this discrepancy is the adaptivity extension of PIVOT, which, for the anti-correlated data set, sets a small set of objects (less than 10%) to the streaming algorithm, reducing the threshold crossings and the incurred network cost. The effect of the additivity extension is more visible at the latter part of the stream, since the extension relies on stream statistics to detect the candidate objects. In terms of transfer volume, this difference becomes apparent only at the end of the stream, since the messages exchanged by PIVOT also include the pivot point coordinates, and are therefore larger than the messages sent by DIRECT.

The initialization phase of PIVOT and DIRECT induces a small network cost, for broadcasting the initial threshold queries to all sites. Notice that this is a *one-time cost*, and, therefore, with a small significance for continuous skyline queries. In the previous experiments, the maximum initialization cost over all runs and for both algorithms was found to be less than 25 Kbytes per node, i.e., less than 25 Mbytes total. The total transfer volume (including initialization cost) required by the algorithms is shown in Fig. 5(c) (the figure corresponding to number of messages is almost identical to Fig. 5(b), and is omitted). We see that, for the anti-correlated data set, CENTR appears to be more efficient at the early stages of the stream compared to PIVOT and DIRECT. This is expected, since CENTR does not require initialization. How-

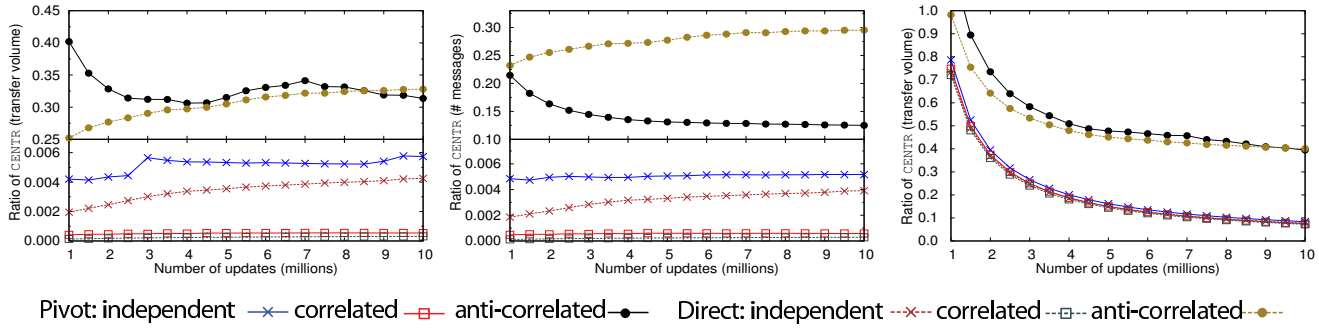


Fig. 5. Effect of the correlation of dimensions to the performance of PIVOT and DIRECT: (a) transfer volume, (b) # messages, (c) transfer volume including one-time initialization cost.

ever, already after around 1.5 million updates, the amortized transfer volume of both PIVOT and DIRECT becomes less than the corresponding cost of CENTR. Since most real-world applications involve long-running – possibly infinite – streams, the one-time initialization cost of the proposed algorithms is not an important concern. Instead, as more updates arrive in the stream, the amortized transfer volume of the two proposed algorithms converges to their running transfer volume. Therefore, the running cost of each algorithm (Fig. 5(a) and (b)) is a more interesting evaluation indicator.

Maximum Change. The streams in the previous simulations were generated assuming a maximum relative change maxCh $= 0.02$ per update. This value is reasonable for simulating real-world applications, since the stream readings usually arrive at regular intervals, e.g., every 30 seconds, and therefore most updates are expected to be small. However, to verify the applicability of PIVOT and DIRECT for fast-changing streams, we have also conducted experiments with different maximum change values, up to 0.16. Fig. 6(a) plots the measured transfer volume and number of messages required by PIVOT and DIRECT for data sets generated with independent dimensions. As expected, increasing maxCh results to an increase of the network cost of both algorithms. Nevertheless, even for maxCh $= 0.16$, PIVOT enables network savings of 80% compared to CENTR in terms of transfer volume, and 90% in terms of messages. DIRECT is even more efficient, requiring 88% less network volume, and 90% less messages compared to CENTR. The 8% difference in the transfer volume between PIVOT and DIRECT is attributed to the more compact threshold queries of DIRECT, which do not need to include the pivot points. For small maxCh values, the network savings of both algorithms are substantially higher, approximating 100%.

B. Scalability

To investigate the scalability of the two algorithms, we also ran experiments with different network sizes. To ensure that each site receives a substantial number of updates for each object, the number of rounds in each experiment was set such that each site receives an expected number of 10000 updates. Therefore, the cost of CENTR varied with the network size, starting from 152 Mbytes for 500 sites, and reaching to 763 Mbytes for the largest network of 2500 sites.

The cost of PIVOT and DIRECT for the different network sizes is presented in Fig. 6(b), as a ratio of the corresponding cost of CENTR for the same setup. Clearly, both PIVOT and DIRECT maintain a steady cost ratio compared to CENTR, independent of the network size (the small peaks visible in the plot for networks of 500 and 1500 sites are due to random artifacts in the generated data sets). For all network sizes, the transfer volume is less than 3% of CENTR for PIVOT and less than 1% for DIRECT, whereas the number of messages remains always below 1% for both.

We have also considered experiments with different numbers of objects (from 1000 to 5000). Similar to the previous experiment, the stream size was adapted to the number of objects (5000 expected updates per object, reaching to a total of 25 million updates for the 5000-objects configuration). As seen in Fig. 6(c), the cost ratio for PIVOT in terms of transfer volume slightly increases with the number of objects. This behavior is expected, since the denser area around the skyline (attributed to the increased number of objects) leads to more frequent threshold crossings and updates in the skyline. This does not affect the number of messages, because all threshold crossings observed due to an update by a node are packed to a single message. Nevertheless, even for the 5000-objects experiment, the transfer volume of both PIVOT and DIRECT does not exceed 4% of CENTR, whereas the number of messages remains always less than 2%. The scalability experiments were also repeated in configurations where each site monitored a subset of the objects, with very similar results.

C. Different function types

The final set of experiments with synthetic data focused on investigating the influence of the number of functions to the performance of PIVOT and DIRECT, and on verifying the applicability of the algorithms to different function types – not necessarily linear. Fig. 7(a) presents the performance of PIVOT and DIRECT when monitoring 2, 3, and 4 linear functions, i.e., the skyline space is of 2, 3, and 4 dimensions. Notice that the transfer volume for the baseline varies with the number of functions, since the number of object dimensions are increased. For 2 dimensions, the transfer volume of CENTR is 305 Mbytes, for 3 dimensions it is 343 Mbytes, and for 4 dimensions it reaches to 381 Mbytes.

We observe that an increase of the number of functions leads to higher network requirements for both PIVOT and DIRECT. The main reason for this observation is that by adding functions – dimensions in the skyline space – we increase the frequency of synchronizations (recall that threshold crossing in a single dimension is sufficient to invoke the synchronization process). Nevertheless, even for the experiment with 4 functions, PIVOT is substantially more efficient than CENTR, reducing the transfer volume by 70%, and the messages by more than 80% by the end of the stream. Notice that skylines of higher dimensions are rarely considered, since in high dimensions most of the objects end up in the skyline, rendering it useless. Also note that PIVOT is more efficient than DIRECT for 3 or more functions. The inefficiency of DIRECT

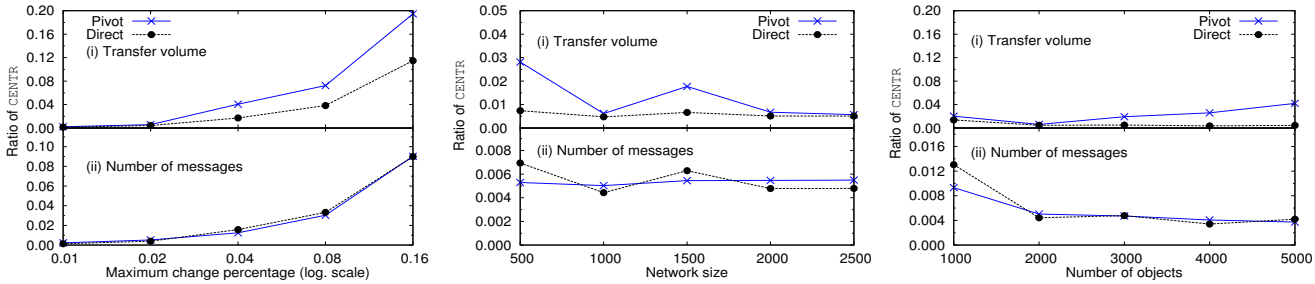


Fig. 6. Effect of (a) maximum relative change, (b) network size, (c) number of objects, to the transfer volume of PIVOT and DIRECT.

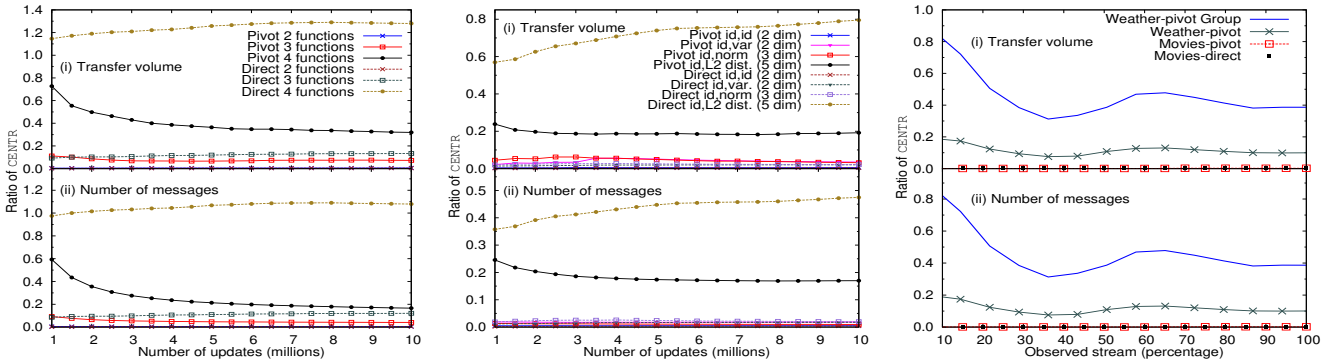


Fig. 7. (a) Effect of the number of functions, (b) Effect of the function types, (c) Experiments with real data sets.

in the experiments with 4 functions is attributed to a high frequency of threshold crossings, due to the increased number of functions. PIVOT on the other hand identifies the objects causing frequent threshold crossings, through the adaptivity extension, and sets them to the streaming algorithm, thereby avoiding the majority of the threshold crossings.

We also conducted experiments with more complex functions, namely the Euclidean norm on two dimensions ($Norm(\vec{v}(o,t)) = \sqrt{\sum_{i=1}^{2} \vec{v}(o,t)[i]^2}$), the L2 distance on four dimensions ($L2(\vec{v}(o,t)) = \sqrt{\sum_{i=1}^{2} \vec{v}(o,t)[i]^2 - \vec{v}(o,t)[i+2]^2}$), and the variance of one dimension on all sites ($Var(\vec{v}(o,t)[i]) = \sum_{p \in \mathcal{P}(o)} (\vec{v}(o,t,p)[i])^2 - \vec{v}(o,t)[i]^2$). In all experiments, the skyline space was 2-dimensional, with $\boldsymbol{f}[0]$ set as the identity function, and $\boldsymbol{f}[1]$ set as one of the three functions above.

Fig. 7(b) plots the network cost incurred by the two algorithms for each function, as the ratio of the corresponding cost of CENTR. For comparison, the figure also includes the cost for the case where both functions are set to the identity function. Notice that, as with the previous experiments, the transfer volume of CENTR was not the same for all functions, since the number of dimensions in the object space differed: for the variance, the transfer volume was 305 Mbytes (2-dimensional objects), for the Euclidean norm 343 Mbytes (3-dim.), and for L2 distance 420 Mbytes (5-dim.).

We see that the proposed algorithms substantially outperform CENTR, also on skylines defined through non-linear functions. The improvement is in fact similar to the improvement observed with linear functions. The only exception involves the experiments with DIRECT used for monitoring the pair of identity and L2 distance functions. For this configuration, DIRECT reduces the transfer volume only by 20% compared to CENTR. DIRECT does not perform well in this configuration due to its local monitoring process, which requires constructing balls in the $2d$-space for each function, i.e., in the 8-dimensions for L2 (cf. Section III-A). This substantially increases the frequency of threshold crossings, and consequently also the transfer volume. PIVOT, on the other hand, reduces the network cost to around 20% of the baseline, since: (a) it constructs balls in the d-dimensional space, and not in the $2d$-dimensional space and, (b) it uses the adaptivity extension, which avoids a large number of threshold crossings.

D. Experiments with real data sets

We have also conducted experiments with two real-world data sets, WEATHER and MOVIES. WEATHER was downloaded from the website of the National Oceanic and Atmospheric Administration (NOAA). The data set includes weather statistics collected from a network of sensors distributed around the globe. For our experiments, we used a subset of the data set for years 2010 and 2011, by excluding the sensors with incomplete location meta-data or infrequent readings. The resulting data set contained 93.6 million readings of 5423 sensors distributed in 257 countries. An interesting characteristic of this data set is that, even though the value of each object (country) is fragmented over many sensors, each sensor always maintains the data of *a single object*, i.e., the weather statistics corresponding to a single country. This has two important consequences. First, as shown in Theorem 1, DIRECT is provably worse than PIVOT for such a setup, and therefore we do not use it in this experiment. Second, our experiments have shown that the query grouping extension introduced at Section III-B is not beneficial for this extreme scenario, since every time a small threshold violation occurs at a composite pivot point, a large number of distinct sensors need to be contacted for updating the composite pivot point. Therefore, for this data set, we present results of PIVOT, both with and without query grouping.

MOVIES is the largest of the Movielens data sets, published by the grouplens group. The data set contains 10 million ratings of 10681 movies provided by 71567 users, and is frequently used for evaluating recommender systems. In the context of this work, MOVIES is used to simulate the scenario

where a large number of servers distributed around the world (such as eBay servers) collaborate to maintain a set of useful skylines on collected user ratings. Since the data set does not contain any kind of user demographics that would allow us to break the stream to sites, we introduced a random distribution of the users to 200 sites. Each site accepts ratings for all movies, and the initial ratings at each site are set based on a sample of the ratings for the movie.

Fig. 7(c) shows the incurred network cost for maintaining two indicative skylines on these data sets: (1) for WEATHER, the skyline of countries with lower average temperatures and lower average dew points, and, (2) for MOVIES, the movies with the highest average ratings and the highest number of ratings in the network. The transfer volume is always reported as a percentage of the corresponding cost of CENTR, which was 2.8 Gbytes for WEATHER, and 248 Mbytes for MOVIES.

Both methods enable substantial improvement on the incurred network cost, similar to the improvement with the synthetic data sets with different correlations (cf. Fig. 5). With respect to WEATHER, PIVOT without query grouping is more efficient than the fully-fledged PIVOT, requiring 4 times less network cost. Compared to CENTR, PIVOT without grouping requires only 10% of the cost of CENTR, both with respect to number of messages and transfer volume. The network savings for MOVIES approached 100% for both algorithms.

We also see that WEATHER is more difficult to handle compared to MOVIES, i.e., the network savings are lower. This is due to the characteristics of the two data sets. On the one hand, MOVIES has correlated dimensions, i.e., a movie with high average rating is highly likely to have a high number of ratings. As discussed in Section V-A, our algorithms thrive in these data sets, requiring a near-zero network cost. On the other hand, WEATHER has two properties that make it a difficult data set: (a) the similar weather statistics observed in nearby countries, leading to tight threshold queries, and to frequent changes in the skyline, and, (b) the periodicity of the readings due to the day-night cycle, which causes frequent changes in the skyline. Extreme weather situations, such as the extremely low temperatures in continental Europe in the winter of 2010-2011 (starting at around 50% of the stream), also cause drastic skyline changes and increased network requirements. Nevertheless, even with this data set, the overall network savings are significant, reaching to 90%.

Summary. The experimental evaluation showed that the proposed algorithms substantially outperform CENTR, the only available alternative. Cost reduction was frequently in the range of two orders of magnitude, as shown with experiments on both real and synthetic data sets, and using different number and types of functions. Both PIVOT and DIRECT were shown to scale well with the number of objects, and number of sites. Furthermore, a thorough experimental comparison of the two algorithms was used to reveal the preferred algorithms for each situation:

- PIVOT is the algorithm of choice for monitoring dense skyline spaces, i.e., with anti-correlated dimensions, and with many functions, due to the adaptivity extension which detects tight threshold queries and assigns their corresponding objects to streaming monitoring.
- PIVOT substantially outperforms DIRECT when monitoring skylines that include non-linear functions with a high number of dimensions, e.g., the L2 distance.
- For 2-dimensional skylines with correlated or independent dimensions, DIRECT is more efficient than PIVOT, since it does not introduce fixed pivot points, allowing higher slack to the objects, and more compact threshold queries.

VI. CONCLUSIONS

In this paper we formally introduced the problem of continuous fragmented skyline queries, i.e., skyline queries defined over *aggregate values* of distributed data, possibly through additional complex functions. To address the problem, we proposed two distributed algorithms that rely on geometric monitoring to reduce the number of updates that need to be transmitted by each site to a central node, thereby drastically reducing the total network cost for maintaining the skyline. We have also described an adaptivity module which enables detecting highly volatile data and handling them more efficiently. An extensive experimental evaluation with massive real-world and synthetic datasets demonstrated the scalability of the algorithm, as well as its significantly improved network efficiency compared to the only available baseline algorithm.

REFERENCES

[1] S. Borzsony, D. Kossmann, and K. Stocker, "The skyline operator," in *ICDE*, 2001.

[2] K. Hose and A. Vlachou, "A survey of skyline processing in highly distributed environments," *VLDB J.*, 2011.

[3] Z. Zhang, R. Cheng, D. Papadias, and A. Tung, "Minimizing the communication cost for continuous skyline maintenance," in *SIGMOD*, 2009.

[4] A. Vlachou, C. Doulkeridis, Y. Kotidis, and M. Vazirgiannis, "Efficient routing of subspace skyline queries over highly distributed data," *TKDE*, vol. 22, no. 12, 2010.

[5] W.-T. Balke, U. Gntzer, and J. X. Zheng, "Efficient distributed skylining for web information systems," in *EDBT*, 2004.

[6] G. Trimponias, I. Bartolini, D. Papadias, and Y. Yang, "Skyline processing on distributed vertical decompositions," *TKDE*, vol. 25, no. 4, 2013.

[7] I. Sharfman, A. Schuster, and D. Keren, "A geometric approach to monitoring threshold functions over distributed data streams," in *SIGMOD*, 2006.

[8] P. Wu, D. Agrawal, Ö. Egecioglu, and A. El Abbadi, "Deltasky: Optimal maintenance of skyline deletions without exclusive dominance region generation," in *ICDE*, 2007.

[9] D. Papadias, G. Fu, M. Chase, and B. Seeger, "Progressive skyline computation in database systems," *TODS*, vol. 30, no. 1, 2005.

[10] Z. Huang, H. Lu, B. C. Ooi, and A. K. H. Tung, "Continuous skyline queries for moving objects," *TKDE*, vol. 18, no. 12, 2006.

[11] Y. Tao, X. Xiao, and J. Pei, "Subsky: Efficient computation of skylines in subspaces," in *ICDE*, 2006.

[12] B. Cui, H. Lu, Q. Xu, L. Chen, Y. Dai, and Y. Zhou, "Parallel distributed processing of constrained skyline queries by filtering," in *ICDE*, 2008.

[13] D. Keren, I. Sharfman, A. Schuster, and A. Livne, "Shape sensitive geometric monitoring," *TKDE*, vol. 24, no. 8, 2012.

[14] G. Cormode and M. Garofalakis, "Approximate continuous querying over distributed streams," *TODS*, vol. 33, no. 2, 2008.

[15] G. Cormode, M. Garofalakis, S. Muthukrishnan, and R. Rastogi, "Holistic aggregates in a networked world: Distributed tracking of approximate quantiles," in *SIGMOD*, 2005.

[16] C. Olston, J. Jiang, and J. Widom, "Adaptive filters for continuous queries over distributed data streams," in *SIGMOD*, 2003.

[17] C. Cranor, T. Johnson, O. Spatscheck, and V. Shkapenyuk, "Gigascope: A stream database for network applications," in *SIGMOD*, 2003.

[18] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "The design of an acquisitional query processor for sensor networks," in *SIGMOD*, 2003.

[19] S. Burdakis and A. Deligiannakis, "Detecting outliers in sensor networks using the geometric approach," in *ICDE*, 2012.

[20] R. Graham, D. Knuth, and O. Patashnik, *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley, 1989.

Geometric Monitoring of Heterogeneous Streams

Daniel Keren[1] , Guy Sagy[2], Amir Abboud[2], David Ben-David[2], Assaf Schuster[2], Izchak Sharfman[2] and Antonios Deligiannakis[3]

[1]Department of Computer Science, Haifa University

[2]Faculty of Computer Science, Israeli Institute of Technology

[3]Department of Electronic and Computer Engineering, Technical University of Crete

Abstract

Interest in stream monitoring is shifting toward the distributed case. In many applications the data is high volume, dynamic, and distributed, making it infeasible to collect the distinct streams to a central node for processing. Often, the monitoring problem consists of determining whether the value of a global function, defined on the union of all streams, crossed a certain threshold. We wish to reduce communication by transforming the global monitoring to the testing of *local* constraints, checked independently at the nodes. *Geometric monitoring* (GM) proved useful for constructing such local constraints for general functions. Alas, in GM the constraints at all nodes share an identical structure and are thus unsuitable for handling heterogeneous streams. Therefore, we propose a general approach for monitoring heterogeneous streams (HGM), which defines constraints tailored to fit the data distributions at the nodes. While we prove that optimally selecting the constraints is NP-hard, we provide a practical solution, which reduces the running time by hierarchically clustering nodes with similar data distributions and then solving simpler optimization problems. We also present a method for efficiently recovering from local violations at the nodes. Experiments yield an improvement of over an order of magnitude in communication relative to GM.

1 Introduction

For a few years now, processing and monitoring of distributed streams has been emerging as a major effort in data management, with dedicated systems being developed for the task [1]. This paper deals with *threshold queries* over distributed streams, which are defined as "retrieve all items x for which $f(x) \leq T$", where $f()$ is a scoring function and T some threshold. Such queries are the building block for many algorithms, such as top-k queries, anomaly detection, and system monitoring. They are also applied in important data processing and data mining tools, including feature selection, decision tree construction, association rule mining, and computing correlations. Another important application is data classification, which is often also achieved by thresholding a function, such as the output of a neural net or support vector machine.

The idea of geometric monitoring [2, 3, 4, 5] has been recently proposed for monitoring such threshold queries over distributed data. While a more detailed presentation is deferred until Section 2.2, we note that geometric monitoring can be applied to the important case of scoring functions $f()$ evaluated at the average (weighted average is handled in a similar way) of dynamic data vectors $v_1(t), \ldots, v_n(t)$, maintained at n distributed nodes. Here, $v_i(t)$ is an m-dimensional data vector, often denoted as *local vector*, at the i-th node N_i at time t (often t will be omitted for brevity). In a nutshell, each node monitors a convex subset, often referred to as the node's *safe-zone*, of the *domain* of these data vectors, as opposed to their *range*. What is guaranteed in the geometric monitoring approach is that the global function $f()$ will not cross its specified threshold as long as all data vectors lie within their corresponding safe-zones. Thus, each node remains silent as long as its data vector lies within its safe zone. Otherwise, in case of a safe-zone breach, communication needs to take place in order to check if the function has truly crossed the given threshold.

The geometric technique can support any scoring function $f()$, evaluated at the average of the dynamic data vectors. Thus, $f()$ is not assumed to obey some simple property (e.g., linearity or monotonicity). To add to the generality of the technique [3, 6], there is absolutely no requirement that the v_i data vectors simply consist of the raw data/measurements of the distributed nodes – in fact, a component of a v_i vector can be either raw data, or any function (i.e., norm, logarithm, power, variance, etc) computed over the data of N_i. Thus, geometric monitoring allows the monitoring of functions that are far more complex and general than simple aggregates. Examples of diverse and important supported functions are:

- Correlation monitoring [2]: Documents may be classified based on the features in them. The local vectors v_i are the contingency tables of e.g. document class vs. feature, and $f()$, evaluated at the average (global) contingency table, is the correlation coefficient or chi-square.
- The analysis of *frequency moments* [7] over distributed data streams, in which a global

function is monitored over the average of the streams. Here v_i are typically local histograms and $f()$ the L^p norm for some p.

- The system monitoring paradigm described in [8]: the v_i are scatter matrices constructed at each node, and $f()$ is computed from the eigenvalues of the average matrix.
- Monitoring various measures (i.e., L^p norms, cosine similarity, Extended Jaccard Coefficient, correlation coefficient) for detecting outliers in sensor networks [3].

A crucial component for reducing the communication required by the geometric method is the design of the safe-zone in each node. Nodes remain silent as long as their local vectors remain within their safe-zone. Thus, good safe-zones increase the probability that nodes will remain silent, while also guaranteeing correctness: a global threshold violation cannot occur unless at least one node's local vector lies outside the corresponding node's safe-zone.

However, prior work on geometric monitoring has failed to take into account the nature of heterogeneous data streams, in which the data distribution of the local vectors at different nodes may vary significantly. This has led to a uniform treatment of all nodes, independently of their characteristics, and the assignment of identical safe-zones (i.e., of the same shape and size) to all nodes.

As we demonstrate in this paper, designing safe-zones that take into account the data distribution of nodes can lead to efficiently monitoring threshold queries at a fraction (requiring an order of magnitude fewer messages) of what prior techniques achieve. However, designing different safe-zones for the nodes is by no means an easy task. In fact, we demonstrate that this problem is not only NP-hard, but also inapproximable. We, thus, propose a more practical solution that hierarchically clusters nodes, based on the similarity of their data distributions, and then seeks to solve many small (and easier) optimization problems. We demonstrate that the proposed solution scales significantly better than the optimal algorithm, while resulting in only a marginal decrease in the quality (target function) of the obtained solution, compared to the optimal case. We emphasize that the optimization algorithm which computes the individual safe-zones is actually performed infrequently, since there is no need to modify the safe-zone of a node unless the value of the function actually crosses the threshold. While the main goal of our algorithms is to reduce communication by minimizing the number of safe-zone breaches by the local vectors (we refer to such breaches as *local violations*), we also propose a method for efficiently recovering from local violations, by carefully picking the nodes to communicate with. The contributions of this work are:

- We formulate a far more general safe-zone assignment problem than those which were treated so far. Instead of constructing one safe-zone which is common to all nodes, we seek to fit each node with a safe-zone that suits its data distribution.

- We study the complexity of this general problem and prove that it is NP-hard and inapproximable.
- We present a practical solution, which uses hierarchical clustering of the nodes to construct the safe-zones, while applying various geometric and computational tools.
- We present an algorithm for efficiently recovering from safe-zone breaches.
- The resulting safe-zones were tested on real data, where we demonstrate that: (i) the hierarchical clustering approach dramatically reduces the running time requirements, compared to an optimal algorithm, with only a marginal decrease in the quality (target function) of the obtained solution, (ii) our techniques may result in one order of magnitude (or even larger) improvements in communication over previous geometric monitoring methods, even for a small number of nodes, and (iii) our algorithm for recovering from safe-zone breaches manages to resolve them by requesting the data (on average) of few nodes.

Outline. We survey related work and also present prior work on the geometric approach in Section 2. In Section 3 we formulate our optimization problem, which involves the design of safe-zones at the nodes. Section 4 presents our algorithmic framework. In Section 5 an algorithm for recovery from a safe-zone violation is discussed. Experiments are resented in Section 6. In Section 7 we prove that the newly presented problem of optimal safe-zone assignment is NP-hard. Lastly, conclusions as well as pointers to future extensions which are beyond the scope of this work, are provided in Section 8.

We, hereafter, denote our proposed method for geometric monitoring of heterogeneous streams as **HGM**, in contrast to prior work on geometric monitoring that is denoted as **GM**.

2 Related Work

2.1 Distributed Monitoring and Summarization Techniques

Previous methods for reducing communication in distributed systems include *sketching,* which reduces the volume of data sent between nodes [7, 9, 10]. Other research concerns detecting "heavy hitters" [11, 12], computing quantiles [13], counting distinct elements [14], optimal sampling [15], and ranking [16]. Theoretical analysis of the monitoring problem is provided in [17], and some non-monotonic functions of frequency moments are treated in [18]. The BBQ system [19] constructs a dynamic probabilistic model for the collection of sensor measurements. The system determines whether it is possible to answer queries only given the model values, or whether it is necessary to poll some of the sensors. In [20], data uncertainty in monitoring linear queries over distributed data is handled by fitting the data with a probabilistic model

and devising an optimal monitoring scheme with respect to this model.

A great deal of work [21] exists for the case in which the threshold function is *linear* (e.g., aggregate, average). However, many functions of interest are *non-linear* and *non-monotonic* and, as elaborated in past work (e.g., [2]), pose a formidable problem for monitoring tasks, whose solution requires a radically different approach than those used for linear/monotonic functions. One possible solution is for a coordinator node to periodically poll system variables and check whether a global condition was violated. Intelligent polling algorithms were proposed [22], but they may miss constraint violations unless the polling interval is set to be very short. [8] applies a distributed paradigm to decide on the dimension of an approximating subspace for distributed data, with the application of detecting system anomalies such as a DDoS attack. A theoretical paper discussed functional approximation in a distributed setting [17], but it only deals with obtaining lower bounds for vector norm functions. In [23] an algorithm is provided for determining whether the norm of the average vector in a distributed system is below a certain threshold, or more generally, whether it lies inside some convex set; however, it does not deal with general functions. In [24] the value of a polynomial in one variable is monitored. A great deal of work was dedicated to distributed monitoring of monotonic functions, usually weighted averages, *max* and *min* operators etc. [25]. Querying non-monotonic functions by representing them as a difference of monotonic ones is presented in [6], but for a static database. Ratio queries over streams are treated in [26], based on a dynamic model which monitors the local ratios vis-a-vis optimally chosen local thresholds. We also handle ratio queries, but in a different method, which is based on optimizing with respect to the data distribution at the nodes. The work in [26] does handle the case of issuing an alert if the ratio of two aggregated (over time) dynamic variables crosses a certain threshold, but does not naturally extend to handle instantaneous (based only on the current values) ratios, such as the ones that we target in our experiments.

2.2 Geometric Monitoring

We now describe some basic ideas regarding the geometric monitoring (GM) technique, which was introduced and applied to monitor distributed data streams in [2, 27]. Table 1 summarizes some important notation.

As described in Section 1, each node N_i maintains a local vector v_i, while the monitoring function $f()$ is evaluated at the average v of the v_i vectors. Before the monitoring process, each node N_i is assigned a subset of the data space, denoted as S_i – its *safe-zone* – such that, as long as the local vectors are inside their respective safe-zones, it is guaranteed that the global function's value did not cross the threshold; thus the node remains silent as long as its local

vector v_i is inside S_i. For details and scope see [5] and the survey in [28]. Recently, GM was successfully applied to detecting outliers in sensor networks [3], extended to prediction-based monitoring [4], and applied to other monitoring problems [29, 30, 31].

Table 1: Glossary of common notations

$N_1, \ldots, N_n$	n distributed nodes with a central coordinator.	C	A convex subset of A, defined as in previous work [5].
$v_i(t)$ (or v_i)	Dynamic data vector at node N_i.	p_i	Probability distribution function (pdf) at node N_i.
$v(t)$ (or v)	The average of the v_i vectors.	S_i	Safe-zone at node N_i.
$f()$	The monitored function.	GM	Previous geometric monitoring methods.
HGM	The proposed approach for geometric monitoring of heterogeneous streams.	T	Threshold: the monitored condition is $f(v) \leq T$.
A	The admissible region, defined as all v such that $f(v) \leq T$.	Local violation global violation	When $v_i \notin S_i$ When $v \notin C$

A basic construct in GM is the *admissible region*, defined by $A \triangleq \{v | f(v) \leq T\}$. Since the value we wish to monitor is $f\left(\frac{v_1+\ldots+v_n}{n}\right)$, any viable assignment of safe-zones must satisfy $\bigwedge_{i=1}^{n} (v_i \in S_i) \rightarrow v = \frac{v_1 + \ldots + v_n}{n} \in A$. This guarantees that as long as all nodes are silent, the average of the v_i vectors remains in A and, therefore, the function has not crossed the threshold. In case of a local violation (safe-zone breach) communication needs to take place in order to check if the function has truly crossed the given threshold. This process may naively always require the collection of the v_i vectors from all nodes, or it can be performed more efficiently using our violation recovery algorithm, described in Section 5. The question is, of course, how to determine the safe-zone S_i of each node N_i; in a sense to be made precise in Section 3, it is desirable for the safe-zones to be as large as possible.

In [5] it was proved that all existing variants of GM share the following property: each of them defines some convex subset C of A (different methods induce different C's), such that each safe-zone S_i is a translation of C (see Figure 1) – that is, there exist vectors u_i $(1 \leq i \leq n)$ such that $S_i = \{u_i + c | c \in C\}$ and $\sum_{i=1}^{n} u_i = 0$.

This observation unifies the distinct variants of GM, and also allows to easily see why $\bigwedge_{i=1}^{n} (v_i \in S_i)$ implies that $v = \frac{v_1+\ldots+v_n}{n} \in C \subseteq A$ – it follows immediately from the fact that

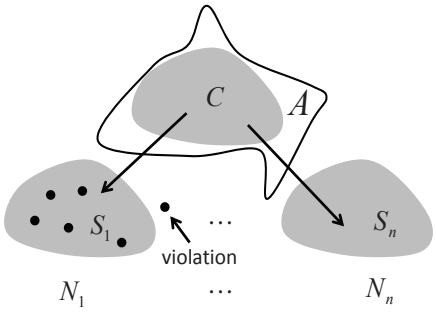


Figure 1: Schematic description of the GM method. A convex subset C of the admissible region A is determined, and then each node N_k is assigned a safe-zone which is a translation of C.

convex subsets are closed under taking averages and from the fact that the u_i vectors sum to zero. Methods for constructing C include:

- The method of *bounding spheres* [2]. A point $p \in A$ (the "reference point") is defined to be the average of the vectors $v_i(0)$, and C is defined as the set of all points q such that the ball whose diameter is the segment $\overline{pq}$ is entirely contained in A (see Fig. 2).

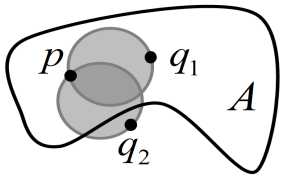


Figure 2: "Bounding spheres" approach. The point q_1 belongs to C, since the ball whose diameter is the segment $\overline{pq_1}$ is contained in A; however, q_2 does not belong to C, as the corresponding ball is not contained in A.

- "Shape sensitive geometric monitoring" [27, 5]. In this approach the bounding spheres algorithm was improved by a better choice of the reference point p, as well as replacing the bounding spheres by ellipsoids whose parameters approximate the distribution of the data at the nodes (but with the underlying assumption that this distribution is shared by all nodes).

- The "optimized safe-zones" approach, briefly introduced in [5], consists of searching for a subset C of A which is both convex and "good", in the sense that $\int_C p(v)dv$ is large, where $p(v)$ is the probability density function (termed as pdf hereafter) of the data distribution. The motivation is straightforward – as the integral becomes larger, then on the average it will take the data vectors more time to exit C.

In this paper we assume that C is given; it can be provided by any of the abovementioned methods (obviously, if A itself is convex, we just choose $C = A$). We propose to extend previous work in a more general direction. Our goal here is to handle a basic problem which haunts all the existing GM variants: *the shapes of the safe-zones at different nodes are identical.* Thus, if the data is heterogeneous across the distinct streams (an example is depicted in Figure 3), meaning that the data at different nodes obeys different distributions, existing GM algorithms will perform poorly, causing many local violations that do not correspond to global threshold crossing (false alarms).

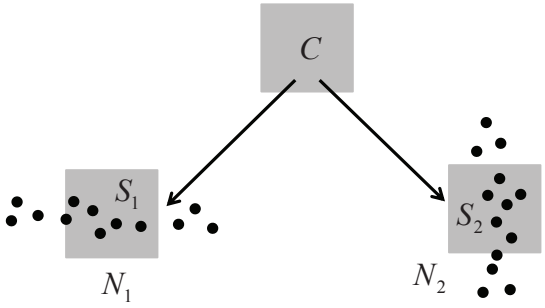


Figure 3: Why GM may fail for heterogeneous streams. Here C is equal to a square, and the data distribution at the two nodes is schematically represented by samples. In GM, the safe-zones at both nodes are restricted to be a translation of C, and thus cannot cover the data; HGM will allow much better safe-zones (see Section 3, and Figure 4).

In this paper, we present a more general approach that allows to assign *differently shaped* safe-zones to different nodes. As we will shortly demonstrate, our approach requires tackling a difficult optimization problem, for which practical solutions need to be devised.

3 Safe-Zone Design as an Optimization Problem

We now seek to formulate an optimization problem, whose output is the safe-zones at all nodes. The safe-zones should satisfy the following properties:

Correctness: If S_i denotes the safe-zone at node N_i, we must have:
$\bigwedge_{i=1}^{n} (v_i \in S_i) \rightarrow \frac{v_1 + ... + v_n}{n} \in C$. This ensures that every threshold crossing by $f(v)$ will result in a safe-zone breach in at least one node.

Expansiveness: Every safe-zone breach (local violation) triggers communication, so the safe-zones should be as "large" as possible. We measure the "size" of a safe-zone S_i by its probability volume, defined as $\int_{S_i} p_i(v)dv$ where p_i is the pdf of the data at node N_i. Probabilistic models

have proved useful in predicting missing and future stream values in various monitoring and processing tasks [19, 20, 32], including previous geometric methods [5], and their incorporation in our algorithms proved useful in monitoring real data (Section 6).

Simplicity: At each time step, every node must check whether its data vector is in its safe-zone. To render this task computationally feasible (especially for thin nodes, such as sensors) it is desirable that the safe-zones be simple geometric constructs. The simplicity of the safe-zones is governed by the user, as he/she defines the family of shapes to which the safe-zones belong (Sections 4.2, 6.1.2).

To handle the correctness and expansiveness requirements, we formulate a constrained optimization problem as follows:

Given: (1) probability distribution functions $p_1, \ldots, p_n$ at n nodes
(2) A convex subset C of the admissible region A

$$\text{Maximize} \int_{S_1} p_1 dv_1 \cdot \ldots \cdot \int_{S_n} p_n dv_n \quad \text{(expansiveness)}$$

$$\text{Subject to } \frac{S_1 \oplus \cdots \oplus S_n}{n} \subseteq C \quad \text{(correctness)}$$

where $\frac{S_1 \oplus \cdots \oplus S_n}{n} = \left\{ \frac{v_1 + \cdots + v_n}{n} \,|v_1 \in S_1, \ldots, v_n \in S_n \right\}$, or the *Minkowski sum* [33] of $S_1, \ldots, S_n$, in which every element is divided by n (the *Minkowski average*). Introducing the Minkowski average is necessary in order to guarantee correctness, since v_i must be able to range over the entire safe-zone S_i. Note that instead of using the constraint $\frac{S_1 \oplus \ldots \oplus S_n}{n} \subseteq A$, we use $\frac{S_1 \oplus \ldots \oplus S_n}{n} \subseteq C$. This preserves correctness, since $C \subseteq A$. The reason we chose to use C is that, as it turns out, typically it's much easier to check the constraint for the Minkowski average containment in a convex set; this is discussed in Section 4.4.

To derive the target function $\int_{S_1} p_1 dv_1 \cdot \ldots \cdot \int_{S_n} p_n dv_n$, which estimates the probability that the local vectors of all nodes will remain in their safe-zones, we assumed that the data is not correlated between nodes, as it was the case in the experiments in Section 6 (see also [20] and the discussion therein). If the data is correlated, the algorithm is essentially the same, with the expression for the probability that data at some node breaches its safe-zone modified accordingly. We plan to target such cases in the future.

Note that correctness and expansiveness have to reach a "compromise": figuratively speaking, the correctness constraint restricts the size of the safe-zones, while the probability volume increases as the safe-zones become larger. This trade-off is central in the solution of the optimization problem.

The advantage of the resulting safe-zones is demonstrated by a schematic example (Figure 4),

in which C and the stream pdfs are identical to those in Figure 3. In HGM, however, the individual safe-zones can be shaped very differently from C, allowing a much better coverage of the pdfs, while adhering to the correctness constraint. Intuitively speaking, nodes can trade "geometric slack" between them; here S_1 trades "vertical slack" for "horizontal slack".

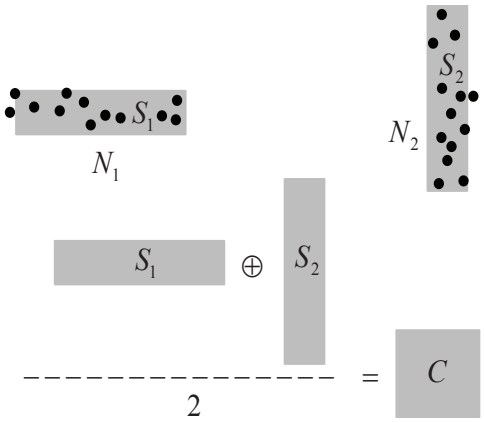


Figure 4: Schematic example of HGM safe-zone assignment for two nodes, which also demonstrates the advantage over previous work. The convex set C is a square, and the pdf at the left (right) node is uniform over a rectangle elongated along the horizontal (vertical) direction. HGM can handle this case by assigning the two rectangles S_1, S_2 as safe-zones, which satisfies the correctness requirement (since their Minkowski average is equal to C). GM (Figure 3) will perform poorly in this case.

The Complexity of the Safe-Zone Problem. The input to the safe-zone problem consists of C and the probability distributions p_i. The difficulty of the problem increases with the number of nodes, the dimensionality of the data, and the complexity of the p_is. In Section 7 we prove that the problem is NP-hard and inapproximable. In the next section, we thus seek to devise practical algorithms to solve this problem.

4 Constructing the Safe-Zones

We now briefly describe the overall operation of the distributed nodes. The computation of the safe-zones is initially performed by a coordinator node, using a process described in this section. This process is performed infrequently, since there is no need to modify the safe-zone of a node unless a global threshold violation occurs. As described in Section 3, the input to the algorithm is: (1) The probability distribution functions $p_1, \ldots, p_n$ at the n nodes. These pdfs can be of any kind (e.g., Gaussian [27], random walk [24], uniform, etc). Moreover, these pdfs

may be given either analytically or by a sample. As discussed in Section 4.5, message passing between the nodes and the coordinator in this initial phase can be reduced by using compact data representations; (2) A convex subset C of the admissible region A.

After the individual safe-zones have been computed, they are assigned to each node. This process is the end of the *initialization stage*. Each node N_k does not initiate communication as long as $v_k(t) \in S_k$. If $v_k(t) \notin S_k$ (local violation) the algorithm described in Section 5 is applied to try and resolve the violation.

4.1 Solving the Optimization Problem

In order to efficiently solve our optimization problem, we need to answer several questions:

- What kinds of shapes to consider for candidate safe-zones? This is discussed in Section 4.2.
- The target function is defined as the product of integrals of the respective pdfs on the candidate safe-zones. Given candidate safe-zones, how do we efficiently compute the target function? This is discussed in Section 4.3.
- Given candidate safe-zones, how do we efficiently test if their Minkowski average lies in C? This is discussed in Section 4.4.
- As we will point out, the number of variables to optimize over is very large, with this number increasing with the number of nodes. It is well-known that the computational cost of general optimization routines increases at a super-linear rate with the number of variables. To remedy this issue, we propose in Section 4.5 a hierarchical clustering approach, which uses a divide-and-conquer algorithm to reduce the problem to that of recursively computing safe-zones for small numbers of nodes.

4.2 Shape of Safe-Zones to Consider

The first step in solving an optimization problem is determining the parameters to optimize over. Here, the space of parameters is huge – *all* subsets of the Euclidean space are candidates for safe-zones. The space of all subsets is extremely complicated: not only is it infinite-dimensional, but also non-linear [34]. For one-dimensional (scalar) data, intervals provide a reasonable choice for safe-zones, but for higher dimensions no clear candidate exists.

To achieve a practical solution, we choose the safe-zones from a parametric family of shapes, denoted by S. This family of shapes should satisfy the following requirements:

- It should be broad enough so that its members can reasonably approximate every subset which is a viable candidate for a safe-zone. If this does not hold, the solution may be grossly sub-optimal. This point is discussed and explained in Section 6.1.2.

- The members of S should have a relatively simple shape (the "simplicity" property, Section 3). In practice, this means that they are polytopes with a restricted number of vertices, or can be defined by a small number of implicit equations (e.g., polynomials [35]). See further discussion in Section 6.1.2.
- It should not be too difficult to compute the integral of the various pdfs over members of S (Section 4.3).
- It should not be too difficult to compute, or bound, the Minkowski average of members of S (Section 4.4).

The last two conditions allow efficient optimization. If computing the integrals of the pdf or the Minkowski average are time consuming, the optimization process may be lengthy. We thus considered and applied in our algorithms various polytopes (such as triangles, boxes, or more general polytopes) as safe-zones, a choice which yielded good results for the simpler problem in [5].

As explained in the experimental evaluation (see discussion in Section 6.1.2), this choice depends on the function $f()$. However, the challenge of choosing the best shape for arbitrary functions and data distributions is, by itself, quite formidable; we now elaborate on a very general paradigm to obtain a good choice.

Bayesian model selection. We propose to view the problem of choosing the family of safe-zones to optimize over as an instance of the *model selection* problem. Very often, it is required to fit a model (e.g. a polynomial approximation) to data. This model is chosen from a graded family of models of increasing complexity and descriptive power: for polynomials the models are ordered by their degree, in our case the safe-zones are ordered by the number of their vertices. One must choose the sub-family of the model to be used (e.g. the degree of the polynomial or the number of safe-zone vertices). The optimal choice achieves a tradeoff between the approximation quality of the model and the number of its parameters (in that regard, it resembles the *minimum length description* paradigm [36]); obviously, the approximation quality increases with the number of parameters, but eventually it is "saturated", meaning that adding more parameters hardly affects the quality, leads to overfitting, and violates the "simplicity of the safe-zone" demand (Section 3). To obtain the optimal choice, we have applied *Bayesian model selection* [37]. Space does not permit to survey the derivations, so we will just provide the result:

Lemma 1 *For any natural number m, denote by F_m the solution of the safe-zone assignment problem which applies polytopes with m vertices, and denote the value of the target function by E_m. The optimal model is the one which maximizes $\log(E_m)+c\,|H_m|^{\frac{1}{2}}$, where H_m is the Hessian matrix of the solution parameters, and c a constant which depends only on the dimension of the*

data vectors. The expression $\log(E_m) + c\,|H_m|^{\frac{1}{2}}$ *is referred to as the* **model evidence**.

The practical meaning of this criterion is the following: as m increases, we obtain a better approximation of the pdf's at the nodes, hence $\log(E_m)$ increases. However, since the safe-zones will eventually "saturate" the pdf's (meaning that increasing m only slightly increases the target function), the optimal fit becomes very "flat" in parameter space, hence the determinant of the Hessian will become very small. Bayesian model selection thus provides a rigorous choice of a model which obtains the optimal tradeoff between the quality of the model and the number of parameters. To quickly find the optimal m, binary search can be applied.

In the experiments (Section 6) we applied both Bayesian model selection and simple safe-zones (triangles and boxes), for monitoring non-linear, non-monotonic functions.

A note on the complexity of the problem. The optimization problem we must solve to obtain the safe-zones is quite more demanding than that in [5], which uses the same safe-zone for each node. For example, suppose that the data vectors are two-dimensional, there are 100 nodes, and we apply octagonal safe-zones. The optimization problem in [5] requires optimizing over 16 parameters (8 vertices for the single octagonal safe-zone common to all nodes, with each vertex having two coordinates), but for the general case treated here, an octagonal region should be defined for every node, yielding a total of 1,600 parameters. Further, the Minkowski sum constraint, $\frac{S_1 \oplus \ldots \oplus S_n}{n} \subseteq C$, introduces a coupling between the parameters, making it impossible to separate the optimization problem into disjoint problems with a smaller number of parameters. No simple solution to this problem exists, since it is NP-hard and inapproximable (Section 7).

4.3 Computing the Target Function

The target function is defined as the product of integrals of the respective pdfs on the candidate safe-zones. Typically, data is provided as discrete samples. The integral can be computed by first approximating the discrete samples by a continuous pdf, and then integrating it over the safe-zone. We used this approach, fitting a GMM (Gaussian Mixture Model) to the discrete data and integrating it over the safe-zones, which were defined as polytopes. To accelerate the computation of the integral, we used Green's Theorem to reduce a double integral to a one-dimensional integral over the polygon's boundary: the integral of a general two-dimensional Gaussian in x, y, $\exp\left(-\left(A^2 + Bxy + C^2 + Dx + Ey + F\right)\right)$ over a region R equals the integral of $\frac{\sqrt{\pi}}{2\sqrt{A}} \exp\left(-\left(Ey + Cy^2 + F - \frac{(D+By)^2}{4A}\right)\right) \mathrm{erf}\left(\sqrt{A}x + \frac{D+By}{2\sqrt{A}}\right)$ over R's boundary. For higher dimensions, the integral can also be reduced to integrals of lower dimensions, or computed using Monte-Carlo methods.

4.4 Checking the Constraints

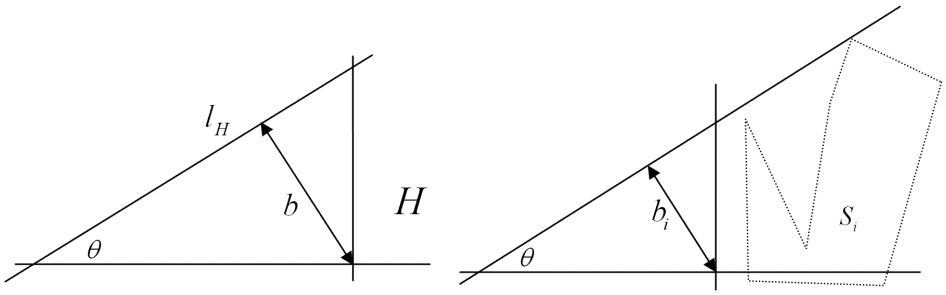


Figure 5: The half-plane approach, for C equal to H which is defined by θ and b (left), by using a supporting hyper-plane for every safe-zone candidate S_i (right).

The Half-Planes Method. To implement a constrained optimization routine, a function is required which checks whether the current parameters satisfy the constraints. This function should return zero if the constraints are satisfied, and a positive number if not. It should also behave smoothly – if the constraint violation is small, it should return a small value.

In our case, the constraint is that the Minkowski average of the safe-zones is contained in C. One way to check the constraint is to compute the Minkowski average and test whether it is inside C and, if not, determine some measure of its deviation from C. Alas, this may entail very high computational complexity, especially in high dimensions, in which computing the Minkowski sum is computationally extensive [38]. To overcome this, we used the following method, which allows to test the constraint without explicitly computing the Minkowski average. For the sake of simplicity we explain the solution for two-dimensional data, but the algorithm proceeds similarly in every dimension. We start with the simple case, in which C is a half-plane (Figure 5), denoted H, and then generalize. Denote the half-plane's boundary by l_H, so H is the set of points below l_h (the algorithm proceeds similarly if l_H is H's lower boundary). The line l_H is defined by the angle θ and by b, its distance from the origin (in higher dimensions similar definitions hold, with the direction of the b vector replaced by a unit vector perpendicular to the hyperplane). Then, in order to determine whether the Minkowski average of the candidate safe-zones $S_1, \ldots, S_n$ is contained in H, one has to find for each S_i the upper supporting line (in higher dimensions, upper supporting hyperplane) in the same direction as that of l_h. For a polytope S_i, this requires rather low computational complexity – only the vertices need to be considered and line sweep algorithms can be applied to further reduce running time. In order for the Minkowski average of the S_i's to be contained in H, it is sufficient that $\frac{\sum b_i}{n} \leq b$. This algorithm also allows to measure the amount of constraint violation, which depends on the value of $\frac{\sum b_i}{n} - b$.

If C is a convex polytope, then C is equal to the intersection of half-planes, and the measure of constraint violation is defined as the max (using the sum is also plausible) of the measures corresponding to the individual half-planes. A general convex C can be efficiently approximated by an inscribed convex polytope [39]. Alternatively, the following method can be used.

The Direct Method. A more direct method to test the Minkowski sum constraint relies on the following result [40]:

Lemma 2 *If P and Q are convex polytopes with vertices $\{P_i\}$, $\{Q_j\}$, then $P \oplus Q$ is equal to the convex hull of the set $\{P_i + Q_j\}$.*

Now, assume we wish to test whether the Minkowski average of P and Q is contained in C. Since C is convex, it contains the convex hull of every of its subsets; hence it suffices to test whether the points $(P_i + Q_j)/2$ are in C, for all i, j. If not all points are inside C, then the constraint violation can be measured by the maximal distance of a point $(P_i + Q_j)/2$ from C's boundary. The method easily generalizes to more polytopes: for three polytopes it is required to test the average of all triplets of vertices, etc.

Comparison of the Half-Planes and Direct Methods. Assume candidate safe-zones $S_1, \ldots, S_n$, where S_i has $V(S_i)$ vertices. Since in the half-planes method the position of every half-plane vs. every summand is computed separately, the running time is proportional to $H_C \sum_{i=1}^{n} V(S_i)$, where H_C is the number of half-planes defining C. The running time of the direct method is proportional to $t_C \prod_{i=1}^{n} V(S_i)$, where t_C is the time required for checking if a point is inside C. Given the types of the candidate safe-zones, their number, and the definition of C, one can choose the method with the lower computational complexity. For example, when many nodes are present, the half-planes method will usually do better. If C is defined by an implicit equation $g()$ (e.g., an ellipsoid) then testing for the presence of a point v in C is easy (i.e., $v \in C$ iff $g(v) \leq 0$), and the direct method is then better (unless $\prod_{i=1}^{n} V(S_i)$ is large enough to offset this advantage). In Section 6 some examples are given as to how these considerations are applied.

4.5 Hierarchical Clustering

While the algorithms presented in Sections 4.3-4.4 reduce the running time for computing the safe-zones, our optimization problem still poses a formidable difficulty. For example, as discussed in Section 4.2, fitting octagonal safe-zones to 100 nodes with two-dimensional data requires to optimize over 1,600 variables, which is quite high for a general optimization problem. To alleviate this problem, we first organize the nodes in a hierarchical structure, which allows

us to then solve the problem recursively (top-down in this hierarchical structure) by reducing it to sub-problems, each containing a much smaller number of nodes.

We first perform a bottom-up hierarchical clustering of the nodes. To achieve this, a distance measure between nodes needs to be defined. Since a node is represented by its data vectors, a distance measure should be defined between subsets of the Euclidean space. We apply the method in [41], which defines the distance between sets by the L^2 distance between their moment vectors (vectors whose coordinates are low-order moments of the set). The moments have to be computed only once, in the initialization stage. The leaves of the cluster tree are individual nodes, and the inner vertices can be thought of as "super nodes", each containing the union (Minkowski average) of the data of nodes in the respective sub-tree. Since the moments of a union of sets are simply the sum of the individual sets' moments, the computation of the moment vectors for the inner nodes is very fast.

After the hierarchical clustering is completed, the safe-zones are assigned top-down: first, the children of the root are assigned safe-zones under the constraint that their Minkowski average is contained in C. In the next level, the grandchildren of the root are assigned safe-zones under the constraint that their Minkowski average is contained in their parent nodes' safe-zones, etc. The leaves are either individual nodes, or clusters which are uniform enough and can all be assigned safe-zones with identical shapes.

Reducing the Transmitted Parameters. Data should be sent from the nodes to the coordinator in order to allow computing the safe-zones, and the coordinator has to send the nodes the definition of their safe-zones. In our experiments this was achieved not by passing the entire data, but the far more compact parameters of the pdf and the first and second order moments of the data, which suffice to compute the safe-zones. Sending the safe-zone parameters to the nodes is also very fast, as the safe-zones are described by a relatively small number of parameters.

5 Violation Recovery

Recall that we define a *local violation* at time t at N_i if its local vector $v_i(t)$ wandered out of its safe-zone S_i. A *global violation* occurs when the average of the vectors in all nodes exits C. Clearly, a global violation implies at least one local violation, however the opposite may not be true, in which case the local violation constitutes a *false alarm*, which should be detected as such and resolved with minimal communication. Often, a different node can remedy the violation: assume that the current local vector $v_j(t)$ at N_j satisfies $v_i(t)+v_j(t) \in S_i \oplus S_j$. Then, N_i and N_j "balance" each other and the local violation is resolved with minimal communication

overhead (the only communication is between nodes N_i and N_j, either directly or through the coordinator). In previous work [2, 27] the coordinator randomly chose nodes in an attempt to balance local violations. Here, we present a more rigorous and efficient method, based on partitioning the collection of nodes to disjoint pairs which optimally balance each other. This proceeds as follows: denoting the data set of node N_i by D_i, define a non-directed, weighted, complete graph over the nodes with the weight of the edge connecting N_i and N_j given by $w(i,j) = \Pr(x_i + x_j \in S_i \oplus S_j | x_i \in D_i \wedge x_j \in D_j)$. These weights are all computed at the coordinator. Please note that $w(i,j)$ can also be computed from the pdfs of the nodes, transmitted to the coordinator in a compact manner, as described at the end of Section 4.5. Then, perform a maximal perfect matching in this graph. The weight of an edge between two nodes is defined as the percentage of pairs of data vectors from both nodes whose sum is in the Minkowski sum of the nodes' safe-zones; therefore, the larger the weight, the higher the probability that the two nodes will balance each other. This matching partitions the nodes into *disjoint* pairs which have an overall high probability to resolve each other's local violations. Please note that if every node N_i simply attempts to balance with node N_j such that $w(i,j)$ is maximal, the system may quickly run into deadlock when many nodes try to balance with the same node – this cannot happen with a disjoint partition.

To handle cases in which the optimal pairing fails to resolve the violation, a hierarchical structure, somewhat resembling the one used for hierarchical clustering, is applied to generalize the disjoint pairing. Following the optimal pairing of nodes, each pair is viewed as a single node whose data is the union of the data at both nodes, and maximal perfect matching is performed on the resulting graph (which has half the number of vertices as the original graph). This process is continued, with the nodes of the next graph corresponding to quadruples of nodes, etc. Call the original nodes "Type 1 node", the pairs obtained in the first maximal perfect matching "Type 2 nodes", the nodes corresponding to quadruples "Type 4 nodes" etc. When the safe-zone at a Type 1 node (i.e., node N_i) is breached, the coordinator attempts to resolve the violation with N_i's partner in the Type 2 node it belongs to. If unsuccessful, it attempts to resolve the violation with the nodes that belong in the subtree of its partner in the Type 4 node it belongs to: if the Type 4 node containing N_i also contains nodes N_j, N_k, N_l, the violation is resolved if $v_i(t) + v_j(t) + v_k(t) + v_l(t) \in S_i \oplus S_j \oplus S_k \oplus S_l$, etc. The overall operation requires $\log(n)$ rounds in the worst case, which helps bound its latency.

Figure 6 demonstrates a scenario with 8 nodes. We explain two cases: (1) N_7 is the only node with a local violation. Then, the coordinator requests the local vector from node N_5. If the violation is not resolved, it asks the local vectors from N_1 and N_4, etc. (2) N_7 and N_1 exhibit local violations. Then, the local vectors from N_7 and N_1 are transmitted to the coordinator,

which can then examine whether these two nodes balance each other. If not, it requests the data of their Type 2 neighbors, namely from nodes N_5 and N_4. In this case, the coordinator can skip the tests of whether the pairs of nodes balance each other, and proceed to test whether all 4 nodes jointly can resolve the violation. If not, the coordinator asks for the local vectors of nodes N_2, N_3, N_6 and N_8.

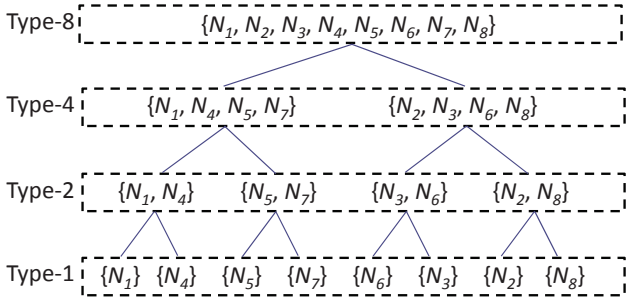


Figure 6: Conceptual hierarchy of nodes used for violation recovery. Depicting the Type-k neighbors for a sample network of 8 nodes.

6 Experiments

HGM was implemented and compared with the GM method, as described in [5], which is the most recent variant of previous work on geometric monitoring that we know of. We are not aware of other algorithms which can be applied to monitor the functions treated here (the ratio queries in [26] deal with accumulative ratios and not instantaneous ones as in our experiments).

6.1 Data, Setup and Monitored Functions

6.1.1 Data and Monitored Functions

Our data consists of air pollutant measurements taken from "AirBase – The European Air Quality Database" [42], measured in micrograms per cubic meter. Nodes correspond to sensors at different geographical locations. The data at different nodes greatly varies in size and shape (see Section 6.2.2) and is irregular as a function of time, as shown in Figure 7 (note however that we use the pdf of the data values per se, not as a time series). The quality of results was measured by the amount of reduction in communication, which is proportional to the number of safe-zone violations.

The monitored functions were chosen due to their practical importance, and also as they are non-linear and non-monotonic and, thus, cannot be handled by most existing methods. In

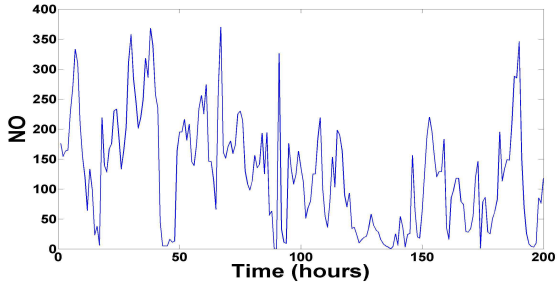


Figure 7: Typical behavior of NO concentrations at a node as a function of time. The behavior of NO_2 is similar.

Section 6.2 results are presented for monitoring the ratio of NO to NO_2, which is known to be an important indicator in air quality analysis [43]. In Section 6.3 the chi-square distance between histograms was monitored for five-dimensional data. An example of monitoring a quadratic function in three variables is also presented (Section 6.4); quadratic functions are important in numerous applications (e.g., the variance is a quadratic function in the variables, and a normal distribution is the exponent of a quadratic function, hence thresholding it is equivalent to thresholding the quadratic).

6.1.2 Choosing the Family of Safe-Zones

To solve the optimization problem, it is necessary to define a parametric family of shapes S from which the safe-zones will be chosen. Section 4.2 discusses the properties this family should satisfy. In [5], the suitability of some families of polytopes is studied for the simpler, but related, problem of finding a safe-zone common to all nodes. The motivations for choosing S here were:

- Ratio queries (Section 6.2) – the triangular safe-zones (Figure 8) have the same structure, but not size or location, as C, and are very simple to define and apply.

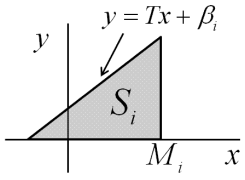


Figure 8: Triangular safe-zones used for ratio monitoring.

- Chi-square monitoring (Section 6.3) – here the motivation was to choose safe-zones with a very simple shape, which still enabled a very substantial improvement over GM. We thus

applied boxes (hyperrectangles), which are also attractive as they render the computation of the Minkowski average trivial.

- Quadratic function (Section 6.4) – here we allowed general polytopes, and tested the results for increasing numbers of vertices, using the Bayesian model selection paradigm (Section 4.2). The model selected was with 12 vertices.

Considering the complexity of the safe-zone problem, and the richness of the space of possible solutions, the challenge of choosing S for arbitrary functions and data distributions is quite formidable. The solutions provided here offer very good performance in terms of safe-zone simplicity and communication reduction over previous work, but the general problem deserves further study.

6.1.3 Optimization Parameters and Tools

The triangular safe-zones (Section 6.2) have two degrees of freedom each (M_i and β_i, see Figure 8), hence for n nodes we have $2n$ parameters to optimize over. For the chi-square monitoring (Section 6.3), each safe-zone is a box in $\mathcal{R}^5$ and therefore is defined by 10 parameters. The safe-zones in Section 6.4 require 36 parameters each. In all cases we used the Matlab routine fmincon to solve the optimization problem [44]. To compute the integral of the pdf on the safe-zones, data was approximated by a Gaussian Mixture Model (GMM), using a Matlab routine [45].

6.1.4 Training/Testing Data, and Stability of the pdf over Time

The proposed algorithm first learns the pdfs at the nodes, and then applies them in order to construct safe-zones. In order to assign safe-zones which will perform well over time, a probabilistic model of the pdfs is required, as in [5, 19]. In our experiments, we learned the model from the "Airbase" data [42] for every month in the year 2006 (training set), and used it to monitor the data for 2007-2008 (testing set). As it can be seen in Figure 9, the average concentration of NO_2 was stable over these three years, and so was the NO concentration.

6.2 Ratio Queries

This set of experiments concerned monitoring the ratio between two pollutants, NO and NO_2, measured in distinct sensors. Each of the n nodes holds a vector (x_i, y_i) (the two concentrations), and the monitored function is $\frac{\sum y_i}{\sum x_i}$ (in [26] ratio is monitored but over aggregates over time, while here we monitor the instantaneous ratio for the current readings). An alert must be sent whenever this function is above a threshold T (taken as 4 in the experiments), and/or when

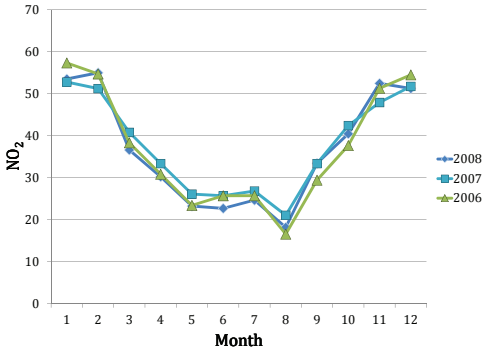


Figure 9: Monthly NO_2 concentrations over three years.

the NO_2 concentration is above 250. The admissible region A is a triangle, therefore convex, so $C = A$. The safe-zones tested were triangles of the form depicted in Figure 8, a choice motivated by the shape of C. The half-planes method (Section 4.4) was used to test the constraints. An example with four nodes, which demonstrates the advantage of allowing different safe-zones at the distinct nodes, is depicted in Figure 10.

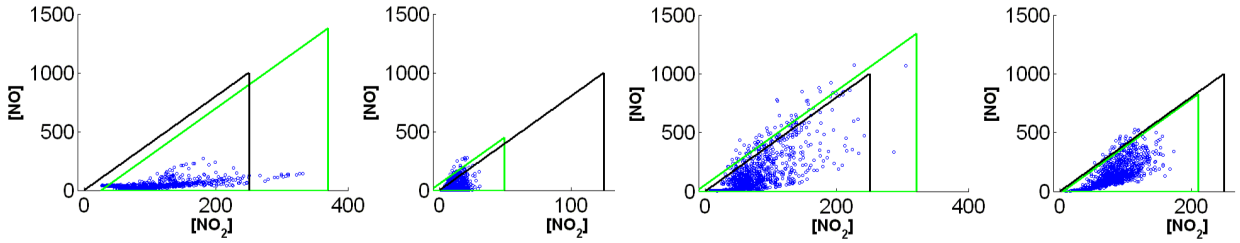


Figure 10: Example of safe-zones with four nodes. The convex set C is the triangle outlined in black, safe-zones are outlined in green. Nodes with more compact distributions are assigned smaller safe-zones, and nodes with high values of the monitored function (NO/NO_2 ratio) are assigned safe-zones which are translated to the left in order to cover more data. This is especially evident in the second from left node, in which the safe-zone is shifted to the left so it can cover almost all the data points. In order to satisfy the Minkowski sum constraint, the safe-zone of the first from left node is shifted to the right, which in that node hardly sacrifices any data points; also, the larger safe-zones are balanced by the smaller ones. Note that HGM allows safe-zones which are *larger* than the admissible region A, as opposed to previous work, in which the safe-zones are subsets of A.

6.2.1 Improvement Over Previous GM Work

We compare HGM with GM in terms of the number of produced local violations. Unless specified otherwise, HGM always uses the hierarchical clustering algorithm for deriving the safe-zones at different nodes. In Figure 11, the number of safe-zone violations is compared for various numbers of nodes. HGM results in significantly fewer local violations, even for a small number of nodes. As the number of nodes increases, the benefits of HGM over GM increase. For a modest network size of 10 nodes, HGM requires less than an order of magnitude fewer messages than GM. In Figure 12 we compare some of the safe-zones that were obtained for both methods in this experiment.

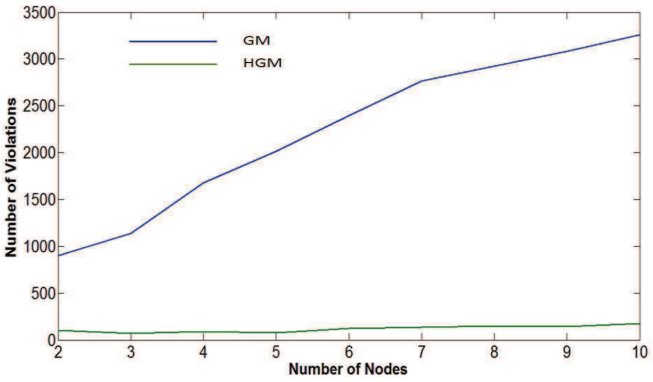


Figure 11: Comparison of our HGM (green) to GM [5] (blue) in terms of number of violations, up to 10 nodes.

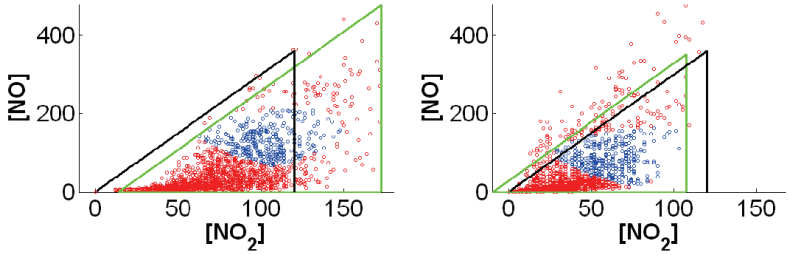


Figure 12: Comparison of HGM to GM in terms of points which cause a violation. At each node, the set C is depicted (dark triangle), the safe-zones of HGM (green triangles), and a sample of the data points (red dots). The points which satisfy the GM constraints are depicted in blue. The advantage of HGM is clear.

6.2.2 Hierarchical Implementation for Ratio Queries

Hierarchical clustering of the nodes was applied in order to reduce the running time of the safe-zone computation. In Figure 13 a typical result is depicted: 92 nodes were clustered into four groups. Two representatives from each of the three largest groups are shown, which correspond to three typical data types: "small" ones (left), indicating low concentrations of NO/NO_2, "drift" ones (middle), with many measurements near the origin but also a sizable number of measurements with high NO concentrations, and "vertical" ones (right), where most measurements are concentrated in a vertical stripe near the origin and there are fewer measurements with high NO. The Matlab routine kmeans [46] was used for clustering the moment vectors.

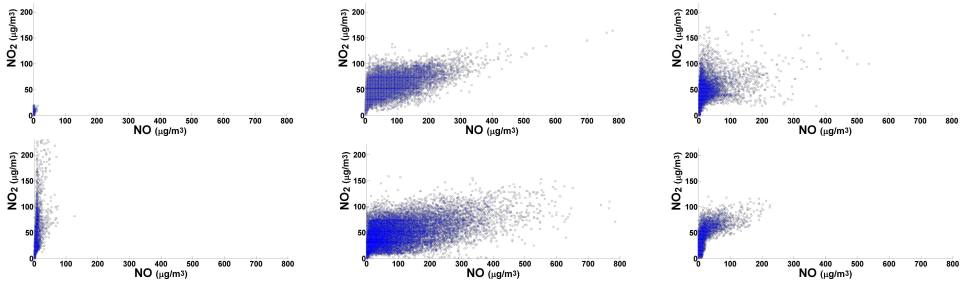


Figure 13: Clustering example. Each column depicts two nodes from one clusters.

In order to test running time and performance, we ran the ratio monitoring algorithm for $n = 30$ to 240 nodes with various thresholds, using either a "flat" mode (direct optimization over $2n$ variables) or the hierarchical clustering method. Table 2 summarizes the results for $n = 60$ and various values of the threshold T, where in each table entry the first number stands for running time (seconds) and the second number for the value of the target function (TF). The running time is higher for the "flat" mode, as the number of parameters to optimize over is much higher, but the performance is slightly better. In Figure 14 we plot the running time of "flat" vs. "hierarchical" as we vary the number of nodes. The running time for "hierarchical" increases linearly with the number of nodes. For 240 nodes, "hierarchical" results in a 40-fold reduction over "flat".

6.3 Chi-Square Monitoring

Another example of an important non-linear, non-monotonic function is the chi-square distance between two histograms, defined by $\chi(f,g) = \sum \frac{(f_i - g_i)^2}{f_i + g_i}$ for histograms f, g. Each histogram was defined as the concentration levels of five pollutants and the monitored function was the

Table 2: Optimization time and target function (TF) for ratio queries, varying the threshold T.

	T = 4		T = 3		T = 2	
T	Time	TF	Time	TF	Time	TF
Tree	23.9 secs.	0.980	23.4 secs.	0.985	23.8 secs.	0.962
Flat	243.6 secs.	0.999	244.9 secs.	0.994	177.4 secs.	0.970

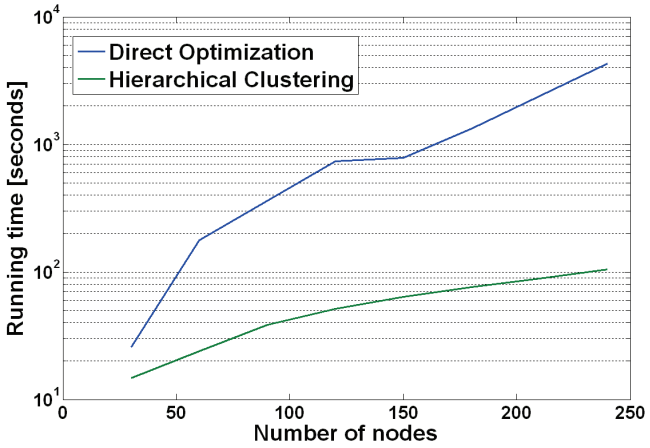


Figure 14: Running time (in logarithmic scale) for "flat"/direct optimization over all the nodes (blue) vs. hierarchical clustering (green).

chi-square distance between the hourly average of two nodes and their average calculated over the previous week (i.e., a measure of how much the hourly distribution deviates from last week's average). The set C was defined as in [5]. The family of safe-zones consisted of five dimensional axis-aligned boxes. An exact solution for box safe-zones is NP-hard even for one-dimensional data (Section 7); so, as in the other experiments, an optimization toolbox was used (Section 6.1.3). When the data distributions in two nodes substantially differ, the advantage of HGM over GM is very clear, since HGM can adapt its safe-zones to fit the distinct distributions at the nodes, allowing a much larger safe-zone to the node with the more varying data. In Figures 15, 16 the different behavior between the nodes is demonstrated and the safe-zones allocated to them is depicted. In Figure 17 we plot the ratio of violations of GM over HGM for various thresholds, for a period of 1,000 hours. As the threshold increases, HGM becomes more superior. For the low thresholds, 0.5 to 0.6, there are actual (global) violations, but as the threshold increases, GM suffers from many "false alarms" (local violations which do not indicate a global violation), but HGM performs well.

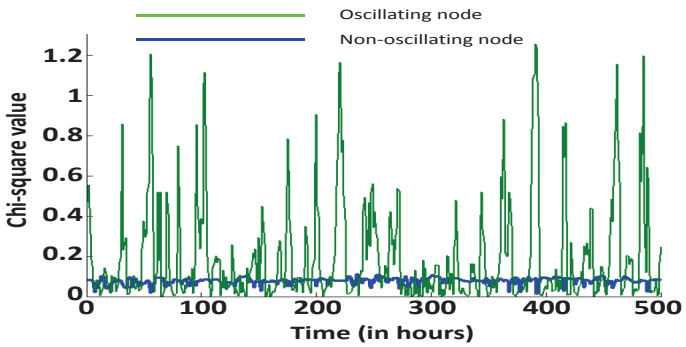


Figure 15: Plots of the chi-square function for two nodes, an "oscillating" one (highly varying data) in green, and a more stable node (in blue).

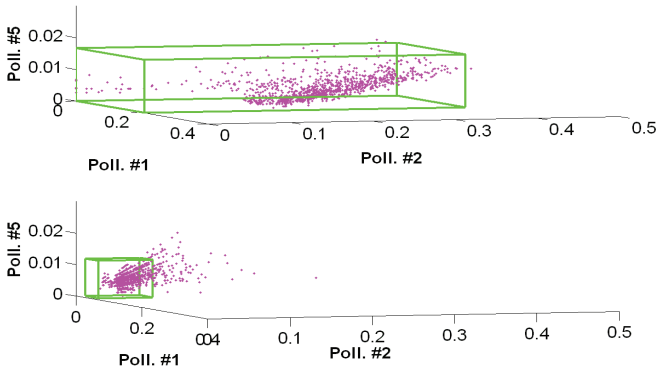


Figure 16: The safe-zones assigned to the two nodes in Fig. 15. The "oscillating" node (top) is assigned a much larger safe-zone, to account for its higher variability. The data is five-dimensional, and a three-dimensional projection is depicted, corresponding to the pollutants NO, NO2, and SO2. Pink dots denote samples from the data, safe-zones are in green.

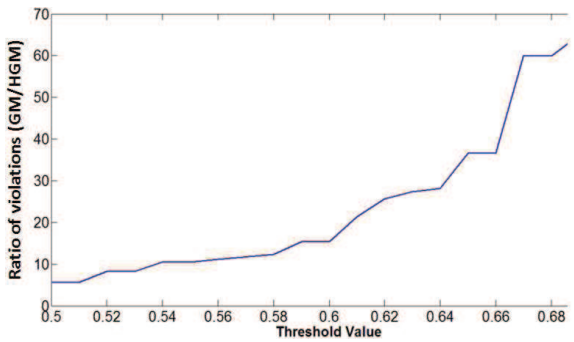


Figure 17: Comparing the number of GM vs. HGM violations. Horizontal axis is the threshold T for the chi-square function, vertical axis is the ratio between numbers of violations in GM vs. HGM.

6.4 Monitoring a Quadratic Function

Another example consists of monitoring a quadratic function with more general polyhedral safe-zones in three variables (Figure 18). The data consists of measurements of three pollutants (NO, NO_2, SO_2), and the safe-zones are polyhedra with 12 vertices, where the number of vertices was chosen according to the Bayesian model selection paradigm (Section 4.2). In Figure 19, the model evidence (Lemma 1) is plotted as a function of the number of safe-zone vertices (since the dimension is three, the minimal number of vertices is four). The admissible region A is the ellipsoid depicted in pink; since it is convex, $C = A$. As the extent of the data is far larger than A, the safe-zones surround the regions in which the data is denser. In order to check the constraints, the direct method was applied (Section 4.4), with the implicit quadratic function defining the ellipsoid.

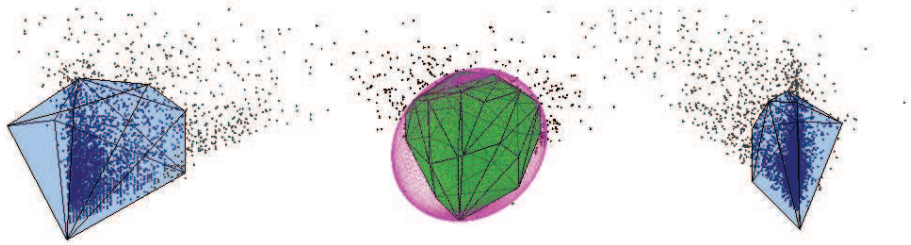


Figure 18: Monitoring a quadratic function. The set C is the pink ellipsoid, the safe-zones are polyhedra with 12 vertices each (in pale blue), and their Minkowski average is in green.

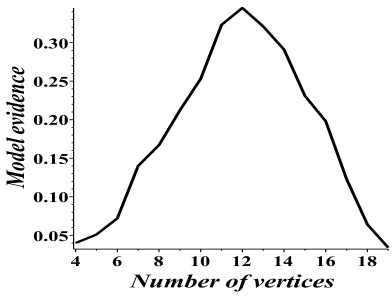


Figure 19: Evidence for modeling safe-zones for the data in Figure 18 as a function of the number of vertices.

6.5 Violation Recovery Performance

We tested the violation recovery algorithm (Section 5) for ratio monitoring (Section 6.2). We randomly chose 64 nodes, over 10,000 hourly measurements. On average, the recovery algorithm enabled 61% of the local violations to be resolved between pairs (Type 2 nodes), 23% required

Type 4 nodes, 10% required Type 8 nodes, 3% required Type 16 nodes, 2% required Type 32 nodes, and only 1% required collecting data from all 64 nodes. Thus, on the average, only 4.7 nodes out of 64 were required to resolve a violation.

7 Complexity of the Safe-Zone Problem

We next provide three theorems which concern the complexity of the safe-zone assignment problem, as formulated in Section 3.

Theorem 1 *The safe-zone problem, even for two nodes and one-dimensional data, is NP-hard and inapproximable.*

Proof We show that the *biclique problem* [47] – known to be NP-hard and inapproximable – can be reduced to the safe-zone problem. Biclique is formulated as follows: given a bipartite graph G with sides L, R, find a biclique with the maximal number of edges, where a biclique is defined as a complete bipartite graph (that is, subsets L' of L and R' of R such that G contains all edges between vertices in L' and R'). Given such a graph, we construct a safe-zone problem whose solution provides a solution to biclique. Assume R has nodes $r_1...r_n$, and L nodes $l_1...l_m$. Let the set of edges, E, be a subset of $\{i, j\}$, where $i \in \{1...n\},\ j \in \{1...m\}$. Associate with this graph the distributions P_R having delta function (pointwise) probability distributions at locations $x_i,\ i = 1..n$, and same for P_L at locations $y_j,\ j = 1..m$ (narrow Gaussians will also do). The only restriction on $\{x_i, y_j\}$ is that $x_i + y_j = x_{i'} + y_{j'} \Rightarrow i = i', j = j'$, which is trivial to achieve. Now, define $A = \{x_i + y_j | (i, j) \in E\}$. Note that the safe-zones must be subsets of $\{x_i\}$ and $\{y_j\}$ (including other points will not add any probability, as all the probability mass resides in x_i, y_j). Note also that S_x, S_y satisfy the Minkowski sum constraint iff the respective subsets of L, R form a biclique, and that the target function for the two safe-zones is proportional to the number of edges in that biclique. Therefore, the safe-zones allow to derive a solution to biclique. Figure 20 provides a drawing illustrating the proof. □

One may suspect that the difficulty of the safe-zone problem is due to allowing a discrete, disconnected admissible region as above. The following theorems prove that is not the case; due to lack of space, we could not include the proofs.

Theorem 2 *If the dimension of the data vectors is at least 4, the optimal safe-zone problem is NP-complete for two nodes even when A is convex.*

Theorem 3 *For more than two nodes, the safe-zone problem is NP-Complete when A is a one-dimensional interval.*

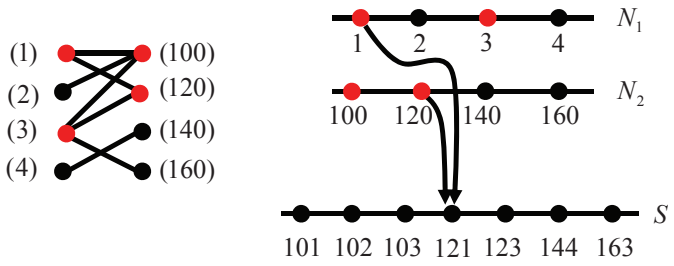


Figure 20: A schematic example of the proof of Theorem 1. The bipartite graph (left), nodes (right-top), and A (right-bottom).

8 Conclusions and Future Work

A method for monitoring threshold queries over heterogeneous distributed streams was presented. A paradigm for minimizing communication is formulated as an optimization problem of a geometric and probabilistic flavor, whose solution assigns each node a "safe-zone" with the property that a node may remain silent as long as its data vector is in its safe-zone. While the problem is shown to be difficult, a practical solution using a hierarchical clustering algorithm is presented and implemented for two, three, and five dimensional data, allowing to achieve substantial improvement over previous work, while using rather simple safe-zones which also reduce the computational effort at the nodes.

We now outline, as space permits, some directions for future work.

- **Correlated streams**. Here, the data we used was uncorrelated between nodes. If some of the streams are correlated, the goal will still be to seek an optimized solution as described in Section 3, the difference being that the overall probability to remain in the distinct safe-zones will not factor to the product of the probabilities for the individual safe-zones.

- **Dynamic change of data distribution within a stream**. If the pdfs at some of the nodes change, the safe-zone optimization process may need to be run again. However, it may suffice to modify the safe-zones only of the nodes whose pdf changed. To see this, assume without loss of generality that we have n nodes with safe-zones $s_i, i = 1 \ldots n$, and that the pdf at nodes $1 \ldots k$ had changed. We can treat these nodes only, and assign them new safe-zones S_i', such that $S_1' \oplus \ldots \oplus S_k' \subset S_1 \oplus \ldots \oplus S_k$. Since Minkowski sums are monotonic with respect to inclusion, this guarantees that the overall Minkowski sum (that is, of all n nodes) will still be contained in the set C.

- **Handling global violations**. If a global violation occurs (that is, $f(v) > T$), the algorithm switches to the monitoring of the condition $v \in C'$, where C' is a convex subset

of A's complement. A plausible solution is to prepare appropriate safe-zones in advance, for this case as well as for different values of the threshold T; the monitoring scheme then simply switches to the appropriate safe-zones.

References

[1] D. J. Abadi, Y. Ahmad, M. Balazinska, U. Çetintemel, M. Cherniack, J.-H. Hwang, W. Lindner, A. Maskey, A. Rasin, E. Ryvkina, N. Tatbul, Y. Xing, and S. B. Zdonik, "The design of the borealis stream processing engine," in *CIDR*, 2005.

[2] I. Sharfman, A. Schuster, and D. Keren, "A geometric approach to monitoring threshold functions over distributed data streams," *ACM Trans. Database Syst.*, vol. 32, no. 4, 2007.

[3] S. Burdakis and A. Deligiannakis, "Detecting outliers in sensor networks using the geometric approach," in *ICDE*, 2012.

[4] N. Giatrakos, A. Deligiannakis, M. N. Garofalakis, I. Sharfman, and A. Schuster, "Prediction-based geometric monitoring over distributed data streams," in *SIGMOD*, 2012.

[5] D. Keren, I. Sharfman, A. Schuster, and A. Livne, "Shape sensitive geometric monitoring," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 8, 2012.

[6] G. Sagy, D. Keren, I. Sharfman, and A. Schuster, "Distributed threshold querying of general functions by a difference of monotonic representation," *PVLDB*, vol. 4, no. 2, 2010.

[7] G. Cormode and M. N. Garofalakis, "Sketching streams through the net: Distributed approximate query tracking," in *VLDB*, 2005.

[8] L. Huang, X. Nguyen, M. N. Garofalakis, J. M. Hellerstein, M. I. Jordan, A. D. Joseph, and N. Taft, "Communication-efficient online detection of network-wide anomalies," in *INFOCOM*, 2007.

[9] A. Arasu and G. S. Manku, "Approximate counts and quantiles over sliding windows," in *PODS*, 2004.

[10] G. Cormode and M. N. Garofalakis, "Histograms and wavelets on probabilistic data," in *ICDE*, 2009.

[11] A. Manjhi, V. Shkapenyuk, K. Dhamdhere, and C. Olston, "Finding (recently) frequent items in distributed data streams," in *ICDE*, 2005.

[12] K. Yi and Q. Zhang, "Optimal tracking of distributed heavy hitters and quantiles," in *PODS*, 2009.

[13] G. Cormode, M. N. Garofalakis, S. Muthukrishnan, and R. Rastogi, "Holistic aggregates in a networked world: Distributed tracking of approximate quantiles," in *SIGMOD*, 2005.

[14] G. Cormode, S. Muthukrishnan, and W. Zhuang, "What's different: Distributed, continuous monitoring of duplicate-resilient aggregates on data streams," in *ICDE*, 2006.

[15] G. Cormode, S. Muthukrishnan, K. Yi, and Q. Zhang, "Optimal sampling from distributed streams," in *PODS*, 2010.

[16] F. Li, K. Yi, and J. Jestes, "Ranking distributed probabilistic data," in *SIGMOD*, 2009.

[17] G. Cormode, S. Muthukrishnan, and K. Yi, "Algorithms for distributed functional monitoring," in *SODA*, 2008.

[18] C. Arackaparambil, J. Brody, and A. Chakrabarti, "Functional monitoring without monotonicity," in *ICALP (1)*, 2009.

[19] A. Deshpande, C. Guestrin, S. Madden, J. M. Hellerstein, and W. Hong, "Model-driven data acquisition in sensor networks," in *VLDB*, 2004.

[20] M. Tang, F. Li, J. M. Phillips, and J. Jestes, "Efficient threshold monitoring for distributed probabilistic data," in *ICDE*, 2012.

[21] R. Keralapura, G. Cormode, and J. Ramamirtham, "Communication-efficient distributed monitoring of thresholded counts," in *SIGMOD*, 2006.

[22] K. Yoshihara, K. Sugiyama, H. Horiuchi, and S. Obana, "Dynamic polling scheme based on time variation of network management information values," in *Proceedings of 11th IFIP/IEEE International Symposium on Integrated Network Management*, 1999.

[23] R. Wolff, K. Bhaduri, and H. Kargupta, "A generic local algorithm for mining data streams in large distributed systems," *IEEE Trans. on Knowl. and Data Eng.*, vol. 21, no. 4, 2009.

[24] S. Shah and K. Ramamritham, "Handling non-linear polynomial queries over dynamic data," in *ICDE*, 2008.

[25] S. Michel, P. Triantafillou, and G. Weikum, "Klee: a framework for distributed top-k query algorithms," in *VLDB '05*. VLDB Endowment, 2005.

[26] R. Gupta, K. Ramamritham, and M. K. Mohania, "Ratio threshold queries over distributed data sources," in *ICDE*, 2010.

[27] I. Sharfman, A. Schuster, and D. Keren, "Shape sensitive geometric monitoring," in *PODS*, 2008.

[28] G. Cormode, "Algorithms for continuous distributing monitoring: A survey," in *AlMoDEP*, 2011.

[29] J. Kogan, "Feature selection over distributed data streams through optimization," in *SDM*, 2012.

[30] O. Papapetrou, M. N. Garofalakis, and A. Deligiannakis, "Sketch-based querying of distributed sliding-window data streams," *PVLDB*, vol. 5, no. 10, 2012.

[31] M. N. Garofalakis, D. Keren, and V. Samoladas, "Sketch-based geometric monitoring of distributed stream queries," *PVLDB*, 2013.

[32] B. Kanagal and A. Deshpande, "Online filtering, smoothing and probabilistic modeling of streaming data," in *ICDE*, 2008.

[33] J. Serra, "Image analysis and mathematical morphology," in *Academic Press, London*, 1982.

[34] B. Wirth, L. Bar, M. Rumpf, and G. Sapiro, "A continuum mechanical approach to geodesics in shape space," *International Journal of Computer Vision*, vol. 93, no. 3, 2011.

[35] D. Keren, D. B. Cooper, and J. Subrahmonia, "Describing complicated objects by implicit polynomials," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 1, 1994.

[36] J. Rissanen, "A universal prior for integers and estimation by minimum description length," *The Annals of statistics*, vol. 11, no. 2, pp. 416–431, 1983.

[37] D. Mackay, "Bayesian interpolation," *Neural Computation*, vol. 4, pp. 415–447, 1992.

[38] H. R. Tiwary, "On the hardness of minkowski addition and related operations," in *Symposium on Computational Geometry*, 2007.

[39] Y. Gordon, M. Meyer, and S. Reisner, "Constructing a polytope to approximate a convex body," in *Geometriae Dedicata*, 1995.

[40] E. Fogel and D. Halperin, "Exact and efficient construction of minkowski sums of convex polyhedra with applications," *Computer-Aided Design*, vol. 39, no. 11, 2007.

[41] M. Elad, A. Tal, and S. Ar, "Content based retrieval of vrml objects: an iterative and interactive approach," in *Proceedings of the sixth Eurographics workshop on Multimedia 2001*, 2002.

[42] "The european air quality database," in *http://tinyurl.com/ct9bh7x*.

[43] M. Kurpius and A. Goldstein, "Gas-phase chemistry dominates o3 loss to a forest, implying a source of aerosols and hydroxyl radicals to the atmosphere," *Geophysical Research Letters*, vol. 30, no. 7, 2007.

[44] http://tinyurl.com/kxssfgl.

[45] DCPR (Data Clustering and Pattern Recognition) Toolbox, http://tinyurl.com/nxospq2.

[46] http://www.mathworks.com/help/toolbox/stats/kmeans.html.

[47] C. Ambühl, M. Mastrolilli, and O. Svensson, "Inapproximability results for maximum edge biclique, minimum linear arrangement, and sparsest cut," *SIAM J. Comput.*, vol. 40, no. 2, 2011.

Communication-efficient Outlier Detection for Scale-out Systems

Moshe Gabel
Technion
Haifa, Israel
mgabel@cs.technion.ac.il

Daniel Keren
Haifa University
Haifa, Israel
dkeren@cs.haifa.ac.il

Assaf Schuster
Technion
Haifa, Israel
assaf@cs.technion.ac.il

ABSTRACT

Modern scale-out services are built on top of large datacenters composed of thousands of individual machines. These must be continuously monitored because unexpected failures can overload fail-over mechanism and cause large-scale outages. Such monitoring can be accomplished by periodically measuring hundreds of performance metrics and looking for outliers, often caused by misconfigurations, hardware failures or even software bugs. Previous work has shown that many failures are indeed preceded by such performance outliers, known as *performance problems* or *latent faults*.

In this work we adapt an existing unsupervised statistical framework for latent fault detection to provide an online, communication- and computation-reduced version. The existing framework is effective in predicting machine failures days before they happen, but requires each monitored machine to send all its periodic metric measurements, which is prohibitive in some settings and requires that the datacenter provide parallel storage and processing. Our adapted framework is able to reduce the amount of data sent and the processing cost at the central coordinator by processing the data in situ, making it usable in wider settings.

We utilize techniques from the domain of stream processing, specifically *sketching* and *safe zones*, to trade-off accuracy for communication and computation, without compromising its advantages. Like the original framework, our adapted framework is unsupervised, does not require domain knowledge, and provides statistical guarantees on the rate of false positives. Initial experiments show that scores yielded by the adapted framework match the original scores very well, while reducing communications by over 90%.

1. INTRODUCTION

In recent years the demand for computing power and storage has increased. Modern Web services and clouds rely on large datacenters, often comprised of thousands of machines. For such large services, it is unreasonable to assume that all machines are working properly and are well configured.

Monitoring is essential in datacenters, since unnoticed faults might accumulate to the point where redundancy and fail-over mechanisms break. Yet the large number of machines in datacenters makes manual monitoring impractical. Instead machines are usually monitored by collecting and analyzing performance counters [3, 5, 10]. Hundreds of counters per machine are reported by the various service layers, from service-specific metrics (such as database query statistics) to general metrics (such as CPU utilization).

In this work we adapt an existing fault detection algorithm [8] using sketching [7, 17] and safe zones [16, 20] to reduce communication and processing requirements by an order of magnitude, while preserving its advantages.

Many existing failure detectors are inflexible [8], and most require centralizing the data in some form. Rule-based failure detectors define a set of watchdogs [10] that monitor specific counters and trigger an alert whenever a predefined threshold is crossed. However, maintaining these static rules requires ongoing manual adjustments.

More advanced methods model service behavior from historical logs. Supervised machine learning approaches [3, 6, 19, 4] train detectors on historic annotated data. Others [5] analyze logs from periods from when the service is guaranteed to be healthy to extract model parameters. Such approaches are sensitive to deviations in workloads and changes in the monitored service itself [22, 9]. After such changes the historical logs and the learned model are no longer relevant. Approaches that require labeled data can be expensive, since labels can be difficult to obtain, and re-labeling may be needed after service changes.

More flexible, unsupervised approaches have been proposed for high performance computing (HPC). Typical approaches [18, 21] analyze textual console logs to detect system or machine failures by examining frequency of log messages. Console logs are impractical in high-volume services for bandwidth and performance reasons: transactions are very short, time-sensitive, and rapid.

Finally, some approaches [13, 15] are unsupervised and flexible, but are not domain independent. They make use of domain insights and knowledge of the monitored service, for example in the domain of distributed file systems, and are therefore limited to specific systems.

Recent approaches to the monitoring problem [8, 15, 14] focus on early detection and handling of performance problems, or *latent faults*. These are outliers – machine behaviors that are indicative of a fault, or could eventually result in a fault, yet fly under the radar of monitoring systems because they are not acute enough, or were not anticipated by the

monitoring system designers. Early detection of latent faults can help prevent future failures and increase the reliability of services.

In previous work [8] we provided evidence that latent faults are common, and we presented a novel, unsupervised outlier detection framework for latent fault detection. In experiments on a real-world production system comprised of 4500 machines, we showed that over 20% of machine failures were preceded by latent faults. Furthermore, we were able to detect latent faults up to 14 days in advance of actual machine or software failures with up to 70% precision and 2% false positive rate – comparable to state of the art supervised techniques in controlled settings [4]. We demonstrated that our system is adaptable, requiring no domain knowledge, no labeled examples, and no parameter tuning in the face of workload changes and software updates. Finally, our system has proven and demonstrated guarantees on the false positive rates, it is non-intrusive, and it scales to very large services.

One drawback of previous work is the large communication and processing costs, prohibitive in some settings. Modern data centers are large, and consequently the resultant counter logs are also large. It may be very difficult to centralize and process such a large amount of data. In the experiments described in [8], the log files were over 10TB per day – too large to centralize and process in one location. Instead we relied on a data-parallel infrastructure [11] built into the data center. Parallel processing may not always be feasible in all situations, however. Furthermore, some large systems are not confined to a single datacenter but are geographically distributed.

In this work we extend our latent fault detection using techniques from the field of stream processing to reduce the size of the data by an order of magnitude, reducing communication and processing requirements, and allowing continuous online processing of distributed streams. The resulting technique is essentially a distributed outlier detector for multiple multivariate data streams, designed for monitoring large-scale online services.

2. SUMMARY OF PREVIOUS WORK

In [8] we presented a statistical latent fault detection framework with 3 derived tests. What follows is a short summary of that work, with the sign test as example.

2.1 Framework

We begin with a reasonable assumption: in a large cluster of machines doing the same job, most machines perform well most of the time. Further, we expect similar machines with similar hardware and software[1] to exhibit roughly similar behavior when measuring performance counters. We therefore compare these machines to find those whose performance differs notably.

There are M machines, each reporting C performance counters at every time point t. We denote by $x(m,t)$ the vector of counter values for machine m at time t. The hypothesis is that the inspected machine is working properly and hence the statistical process that generated this vector for machine m is the same statistical process that generated the vector for any other machine m'. However, if we see that the vector $x(m,t)$ for machine m is notably different from the vectors of other machines, we reject the hypothesis and flag the machine m as *suspicious*, meaning we suspect it manifests a latent fault.

We now make explicit our assumptions on the behavior of the monitored machines: *a*) the majority of machines are working properly at any given point in time; *b*) the machines are homogeneous, meaning they perform a similar task and use similar hardware and software[2]; *c*) on average, the workload is balanced across all machines; *d*) the counters are ordinal and are reported at the same rate; and *e*) the counter values are memoryless in the sense that they depend only on the current time period (and are independent of the identity of the machine).

Formally, we assume that $x(m,t)$ is a realization of a random variable $X(t)$ whenever machine m is working properly. Since all machines perform the same task, and since the load balancer attempts to split the load evenly between the machines, the homogeneous assumption implies that we should expect $x(m,t)$ to show similar behavior. We do expect to see changes over time, due to changes in the workload, for example. However, we expect these changes to be similarly reflected in all machines.

At any time t, the input $x(t)$ to a test S consists of the vectors $x(m,t)$ for all machines m. The test $S(m,x(t))$ analyzes the data and assigns a *score* (either a scalar or a vector) to machine m at time t. Given a test S, and a significance level $\alpha > 0$, we can present the framework as follows:

1. Preprocess: select counters and scale to unit variance;
2. Compute for every machine m the vector: $v_m = \frac{1}{T}\sum_t S(m,x(t))$ (integration phase);
3. Compute the p-values (defined below) $p(m)$ from v_m;
4. Report every machine with $p(m) < \alpha$ as suspicious.

Essentially, the scores for machine m are aggregated over time, so that eventually the norm of the aggregated scores converges, and is used to compute a p-value for m. The longer the allowed time period for aggregating the scores is, the more sensitive the test will be. At the same time, aggregating over long periods of time creates latencies in the detection process. In our previous work we aggregated data over 24 hour intervals, as a compromise between sensitivity and latency.

The p-value for a machine m is a bound on the probability that a random healthy machine would exhibit such aberrant counter values. If the p-value falls below a predefined significance level α, the null hypothesis is rejected, and the machine is flagged as suspicious.

In [8] we derived and evaluated 3 different tests within the framework (different S functions). The sign test accumulates the average normalized direction from machine m to the rest of the machines. The Tukey test measures the average depth of $x(m,t)$ compared to the vectors of other machines at the same time. The LOF test similarly compares the local density of points around $x(m,t)$ to the local density of its neighbors. What follows is a summary of the sign test.

[1]These are reasonable assumptions in practice for many services and datacenters [13, 18].

[2]If this is not the case, we can often split the collection of machines to a few large homogeneous clusters.)

2.2 The Sign Test

The sign test extends the classic statistical sign test to allow the simultaneous comparison of multiple machines. The "sign" of a machine m at time t is the average direction of its vector $x(m,t)$ to all other machines' vectors, and its score v_m is the sum of all these directions, divided by T.

The intuition is that healthy machines are similar on average, and any differences are random. Average directions are therefore random and tend to cancel each other out when added together, meaning v_m will be a relatively short vector for healthy machines. Conversely, if m has a latent fault, then some of its metrics are consistently different from healthy machines, and so the average directions are similar in some dimensions. When summing up these average directions, these similarities reinforce each other and therefore v_m tends to be a longer vector.

Formally, let $\mathcal{M}$ denote the set of all machines in a test, and $M = |\mathcal{M}|$ the number of machines. $\mathcal{T}$ are the time points where counters are sampled during preprocessing (for instance, every 5 minutes for 24 hours in our experiments), t denote a specific time point, and $T = |\mathcal{T}|$. Let m and m' be two machines and let $x(m,t)$ and $x(m',t)$ be the vectors of their reported and preprocessed counters at time t. We use the test

$$S\left(m, x(t)\right) = \frac{1}{M-1} \sum_{m' \neq m} \frac{x(m,t) - x\left(m',t\right)}{\|x(m,t) - x\left(m',t\right)\|} \quad (1)$$

as a multivariate version of the sign function. If all the machines are working properly, we expect this value to be small. Therefore, the sum of several samples over time is also expected not to grow far from zero.

Algorithm 1: The sign test.

foreach *machine* m **do**
  $S\left(m, x(t)\right) \leftarrow \frac{1}{M-1} \sum_{m' \neq m} \frac{x(m,t) - x\left(m',t\right)}{\|x(m,t) - x\left(m',t\right)\|}$;
  $v_m \leftarrow \frac{1}{T} \sum_t S\left(m, x(t)\right)$;
end
$\hat{v} \leftarrow \frac{1}{M} \sum_m \|v_m\|$;
foreach *machine* m **do**
  $\gamma \leftarrow \max\left(0, \|v_m\| - \hat{v}\right)$;
  $p(m) \leftarrow (M+1) \exp\left(-\frac{TM\gamma^2}{2\left(\sqrt{M}+2\right)^2}\right)$;
  if $p(m) \leq \alpha$ **then**
    Report machine m as suspicious;
  end
end

If all machines are working properly, the norm of $v_m = \frac{1}{T} \sum_t S(m, x(t))$ should not be much larger than its empirical mean. The p-value $p(m)$ in Algorithm 1 controls this statistic by guaranteeing a small number of false detections, depending on the significance level α.

3. ONLINE DETECTOR WITH REDUCED COMMUNICATION

We describe an online, communication-efficient version of the latent fault detector summarized in Section 2.

Detecting latent faults requires that each node must send all performance counters measured at each time point: T samples of C counters for each of the M machines. Beyond bandwidth costs, processing so much data is difficult to do on a single machine in a timely manner, due to the size and high dimensionality of the data. We apply two techniques to alleviate this issue.

Sketching is used to reduce the amount of data sent from each machine and processed by the coordinator. Instead of sending all counters, each node calculates a sketch of the said counters and sends only that. The coordinator (or monitoring node) can then perform latent fault detection using the sketches, rather than the original data. In addition to reducing the communication load, this has the added benefit of reducing the computational load, since the dimensionality of the data is greatly reduced.

The framework in Section 2.1 requires that counter values be normalized during preprocessing (step 1), and this is true as well for the sketched version[3]. We use the safe zone approach [16] to monitor both the global mean and the global variance of each counter so that they do not deviate too much from their last known values. Each machine monitors whether its data satisfies a local constraint. If all local constraints at all machines are satisfied, the global mean and variance are known not to have deviated too far from their last known values. These last known values are then used to normalize the counter values at each node, before computing the sketch. If there is any violation, the coordinator polls each node for the current mean and variance, and distributes the new global mean and variance to all nodes.

The general pseudocode is shown in Algorithm 2 and explained in detail below.

3.1 Sketches

Sketching [17, 7] is a common technique used to process large, unpredictable data streams without having to send, store and process all data. It reduces the size of the data, while still enabling queries.

For our purposes, a *sketch* is a summary function that takes a vector and transforms it to a smaller vector while approximately preserving some desired property, for example inner products [1].

We use sketches to modify our tests to greatly reduce the amount of data that must be sent and processed. For example, 200 counters could be reduced to 10 dimensions, achieving an immediate 95% reduction in data size.

Formally, rather than apply test S to the set of all local counter vectors $x(m,t)$, each machine m will first apply a sketching function f to its vectors, and send only the sketch $\hat{x} = f(x(m,t))$ for processing. The modified test $\hat{S}$ will be applied to the sketches rather than the original vector: $v_m = \frac{1}{T} \sum_t \hat{S}(m, \hat{x}(t))$.

One well-suited sketch is the AMS sketch [1], which involves a random linear projection to k dimensions. In our setting, each machine would project its counter vectors to k dimensions using a specially constructed projection matrix: $\hat{x}(m,t) = f(x(m,t)) = Rx(m,t)$ where R is a random $C \times k$ matrix constructed as described in [1].

The AMS sketch is general enough so that the same sketch can be used as input to different tests. Because the sign test relies on normalized directions, and since AMS sketches are linear projections, the sign test can be applied directly to the sketch. In other words, the sum of projected vectors is

[3]Automatic counter selection (part step 1) can be done in advance, offline, using the method described in [8].

Algorithm 2: Online detection pseudocode.

OFFLINE:
Automatically select counters.

INIT / COORDINATOR SYNC:
foreach *counter i in counters* **do**
  Poll all nodes for mean and variance of counter i.
  Distribute new global mean, variance, safe zones.
end

NODE at time point t:
foreach *counter i in counters* **do**
  if *counter not in safe zone* **then**
    Violation: send local mean, variance to coordinator.
    Wait for new global mean and variance.
  end
  Let x_i = value of counter i at time t .
  Normalize x_i with last known global mean and variance.
end
Let x = vector of normalized counter values.
Compute sketch of x and send to coordinator.

COORDINATOR at time point t:
if *violation for counter i* **then**
  Run SYNC.
end
Receive sketches from all nodes.
Compute test function S on received sketches.
Add most recent test function result to v_m.
Subtract least recent test function result from v_m.
Calculate p-value for all machines and issue warnings.

the same as projecting the sum of the vectors. The resulting vector is still small for healthy machines and large for outliers. The Tukey test described in our previous work already relies on a very similar technique, and has been shown to be very effective. The LOF test depends on the distance of pairs of points. In this case, the Johnson-Lindenstrauss lemma [12] guarantees that the projection to $k = O\left(\frac{\log M}{\epsilon^2}\right)$ preserves the distances within a factor of $1 \pm \epsilon$. Since our method averages T comparisons per day in the integration phase, we can further expect that in practice the error will be smaller.

3.1.1 Sign Test on Linear Sketches

The sign test function (1) from Section 2.2 depends only on the normalized direction from $x(m,t)$ to the other vectors. Let B be the unit sphere in C dimensions. Given the assumptions in Section 2.1, for healthy machines the normalized directions to other machines tend to be distributed spherically symmetric over B, resulting in the vector $v_m = \frac{1}{T}\sum_t S(m, x(t))$ being relatively short. Conversely, for machines with consistently anomalous behavior, v_m is a relatively long vector.

Given the sketched vectors $\hat{x}(m,t) = Rx(m,t)$, the sign test is still the sum of normalized directions from $x(m,t)$, after some transformation R. We now show that applying R to the unit sphere B maintains this symmetrical distribution. Let $R = UDV^T$ be the *singular value decomposition* of R.

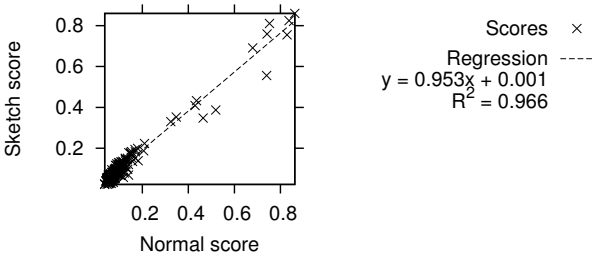


Figure 1: Sign test scores with AMS sketch compared to original scores. Sketch size is 8% of original data.

U and V^T are unitary matrices, and D is a diagonal matrix with positive elements. In geometrical terms, the transformation $R = UDV^T$ is a composition of rotation, followed by non-uniform scaling and dimensional reduction, and finally another rotation – all of which preserve the symmetric distribution around the origin. Therefore the transformation R maps the unit sphere B (in C dimensions) to an ellipsoid B' in k dimensions while preserving the symmetric distribution around the origin.

In summary, since the sign-test uses normalized directions and R preserves their symmetry around $x(m,t)$, we can apply the sign test directly to the sketched vectors $\hat{x}(m,t)$. Moreover, the sign test p-value does not depend on the dimensionality of the vectors, and so we can use it as is.

Preliminary experiments on counter logs from a small sample of 260 machines in a single day show that sign test scores and p-values computed on sketched data match the original very well. Figure 1 shows a comparison of sign test scores based on AMS sketches to regular (centralized, or parallel) sign test scores. The figure and linear regression show that the scores match very well, with $R^2 = 0.966$, very close to 1. The sketch reduced the data size by 92% – from 123 counters to 10 dimensions. The p-values are similarly close to the original values.

3.1.2 Online Integration Using a Sliding Window

The integration phase in stage 2 of the framework in Section 2.1 computes $v_m = \frac{1}{T}\sum_t S(m, x(t))$. Computing $S(m, x(t))$ only requires the data from time t, and therefore it is trivial to turn any test into an online test by keeping a window of test function (S) outputs for the last T sketches sent from the monitored machines. When new data arrives at time t, the coordinator updates the current v_m by computing and adding $\frac{1}{T}S(m, \hat{x}(t))$, and subtracting the least recent stored test result, $\frac{1}{T}S(m, \hat{x}(t-T-1))$. The p-value for each machine in the time window can then be computed in the usual manner. Since the test function S need only be computed for the most recent time, and since the sketches are of low dimension k, processing and memory costs are low. This allows the computation to be done on a single coordinator machine on time, before the next round starts.

3.2 Scaling By Monitoring Variance

Our tests require the data to be standardized during preprocessing: each counter should be globally centered to zero mean and unit variance. In some settings we can assume that a counter's mean and variance do not change much,

or that they have a daily cycle. However, we might wish to avoid that assumption, and handle unpredictable workloads.

We use the *safe zones* approach [16, 20] to monitor both the global mean and the global variance of each counter. In this approach, each monitored machine receives a local constraint on its data $x(m,t)$ from a coordinator machine, such that if all local constraints are satisfied, the global monitored value $f(x(t))$ for some function f of the global aggregate is within a pre-defined threshold. Violations of local constraints are sent to the coordinator machine, which resolves them and sends updated local constraints to participating machines.

Given the last known global mean and variance of the last T samples, we define some lower and upper threshold, for example 0.9 and 1.1 times the last known values. If there is any violation, the coordinator polls each node for the current mean and variance, and distributes the new global mean and variance to all nodes. We can trade-off accuracy and communication by adjusting the high and low thresholds when monitoring. Violations are less likely if global mean and variance are allowed to drift further from their last known values – reducing communication but also decreasing accuracy [16].

We monitor each counter independently, so it is enough to show how we monitor a single counter X. Further note that all tests described in [8] are invariant to data translation, and so we do not monitor the global mean explicitly.

3.2.1 Notations

The set of values of counter X over the last T times and over M nodes (machines) is denoted by $X(t)$. We denote by $X_i(t)$ the values of X at node i for the last T times up to t. Thus $\mathrm{E}[X_i(t)]$ is the mean of the last T values at node i in time t, while $\mathrm{E}[X(t)]$ is the global mean of the last values at all nodes. Denote $\mu_i(t) = \mathrm{E}[X_i(t)]$ the local means, and $\mu(t) = \mathrm{E}[X(t)]$ the global mean. Similarly, we denote $\lambda_i = \mathrm{E}\left[X_i(t)^2\right]$, the local mean of the squares, and $\lambda = \mathrm{E}\left[X(t)^2\right]$ the global mean. Let $V(t) = (\mu(t), \lambda(t))$, and $V_i = (\mu_i(t), \lambda_i(t))$, the global and local monitored vectors, respectively.

3.2.2 Monitoring

We wish to monitor the global variance $\mathrm{Var}(X)$ at each time t. Recall that:

$$\mathrm{Var}(X) = \mathrm{E}\left[X^2\right] - (\mathrm{E}\left[X\right])^2 = \lambda - \mu^2 \quad .$$

We therefore monitor the conditions $L \leq \lambda - \mu^2 \leq H$, for some lower and upper variance thresholds L and H. Figure 2 shows the *admissible region* (the region in which the conditions hold), $0.5 \leq \lambda - \mu^2 \leq 1.5$. Following [16], we aim to find a convex safe zone G which is contained within the admissible region. Since convex sets are closed under averaging, when all local vectors are inside the safe zone, the global mean is guaranteed to be inside as well.

Let $t = 0$ be the last global synchronization time, and let $V(0) = (\mu(0), \lambda(0))$ be the *reference point*, the last known global mean and mean-of-squares, computed that time. For each node i we define the local drift vector $d_i(t)$ as the drift of the current vector from the node's vector during the last synchronization: $d_i(t) = V_i(t) - V_i(0)$.

Since we wish to monitor that the global $V(t)$ is within some convex set G, we define equivalent local conditions on the drift vectors. The current local vectors can be written

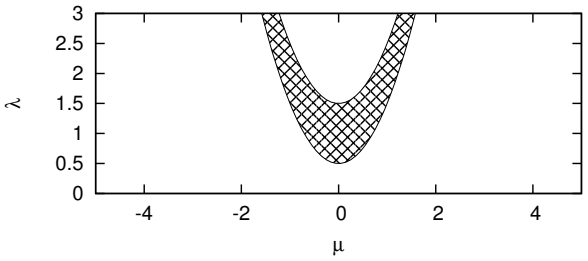


Figure 2: Admissible region for $L = 0.5, H = 1.5$.

in terms of drift vector d_i: $V_i(t) = V_i(0) + d_i(t)$. Note that the global vector is the mean of the local vectors, and can therefore be written as the mean of drifts and the reference point:

$$V(t) = \frac{1}{M}\sum_i V_i(t) = V(0) + \frac{1}{M}\sum_i d_i(t) \quad . \qquad (2)$$

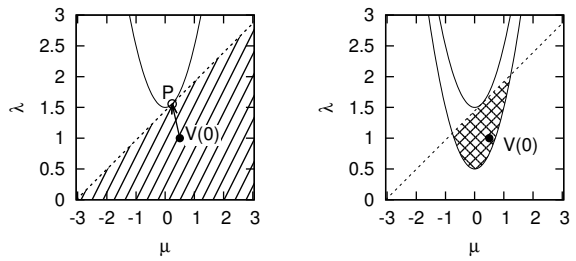
Let $W_i(t) = V(0) + d_i(t)$ be the local drift from the last reference point. Note that $V(t) = \frac{1}{M}\sum_i W_i$, recall G is convex, and from (2) we arrive at the local conditions: if $\forall i, W_i \in G$ then $V(t) \in G$.

To monitor that the variance is between L and H, we derive separate safe zones: one for variance above L and another for variance below H. As long as the local conditions for both safe zones are maintained in all nodes, we are guaranteed that the variance is within the allowed range.

Variance Above Lower Threshold. We wish to define a convex safe zone G_L so that as long as $V(t) \in G_L$ then $\mathrm{Var}(X) \geq L$. This corresponds to monitoring that $\lambda - \mu^2 \geq L$, which is already a convex set – the area above a parabola – and can be directly used as safe zone. Therefore the local condition for each node i is trivial: $I_i(t) \in G_L$: $I_i(t) = V(0) + d_i(t) = (a, b)$ and monitor that $b - a^2 \geq L$.

Variance Below Upper Threshold. We wish to define a convex safe zone G so that as long as $V(t) \in G$ then $\mathrm{Var}(X) \leq H$. This area is the area below a parabola, which is not a convex set. However, we can find a tangent half-plane I below this parabola. This half-plane is a convex set, and since $I \subset G$, then as long as $V(t) \in I$, $V(t) \in G$ and therefore $\mathrm{Var}(X) \leq H$.

We use the reference point $V(0)$ to find the optimal hyperplane. The thresholds H and L are reset during synchronization, so obviously $V(0) \in G$. We can choose any half-space I such that $V(0) \in I$, but to avoid future unnecessary synchronization we choose I such that $V(0)$ is far from the boundary of G. Doing so ensures that drift has to be large to cause a violation. Consequently, we choose I as the tangent at point P, where P is the closest point to $V(0)$ on the parabola $\lambda - \mu^2 = H$, and the local condition is $W_i \in I$. We can find P numerically, or by minimizing the distance from the parabola to $V(0)$. For example, if $V(0) = (0.5, 1)$ and $H = 1.5$, then the closest point on the parabola is $\mu \approx 0.237$. This yields the point $P = (0.237, 1.556)$, and finally the induced safe zone I: the half-plane $\lambda - 0.474\mu < 1.443$. Figure 3(a) shows $V(0)$, P and the resulting safe zone, and Figure 3(b) shows the intersection with the safe zone for the lower limit $L = 0.5$.



(a) Upper threshold. **(b) Upper and lower.**

Figure 3: Safe zones for $L = 0.5, H = 1.5$ where $V(0) = (0.5, 1)$.

3.2.3 Handling Violations

If one of the local conditions $W_j \in G$ is violated, it may be because $\mathrm{Var}(X)$ is no longer in the range, or due to a false alarm. The simplest way to deal with a violation is to perform a global synchronization: each node sends its current $V_i(t)$ to the coordinator. The coordinator "resets the time" to $t = 0$, computes the new global reference point $V(0)$, and sends it to the nodes, where it is used for monitoring and scaling.

In terms of communication, our synchronizations are fairly inexpensive. Each node sends only two numbers per counter (μ and λ), rather than the entire time window of T samples. They also improve the accuracy of scaling, since nodes have fresh global mean and variance. There are safe zone techniques that allow partial synchronization for further communication reduction, for example by balancing a node with local violation with another node that has enough slack [2].

4. FUTURE WORK

This work uses sketching and safe zones to adapt the latent fault detector in [8] to a streaming setting, resulting in an online, communication-efficient outlier detector for common scale-out systems. Preliminary results show that the adapted detector obtains very similar results to those of the original latent fault detector for the sign test. Future work will concentrate on adapting additional tests, evaluating the detector on real-world systems, and exploring the communication-accuracy trade-off.

5. ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Union's Seventh Framework Programme under grant agreement Nº 255951.

6. REFERENCES

[1] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 1999.

[2] D. Ben-David. Violation resolution in distributed stream networks. Master's thesis, Technion I.I.T, 2012.

[3] P. Bodík, M. Goldszmidt, A. Fox, D. B. Woodard, and H. Andersen. Fingerprinting the datacenter: Automated classification of performance crises. In *Proc. EuroSys*, 2010.

[4] G. Bronevetsky, I. Laguna, B. De Supinski, and S. Bagchi. Automatic fault characterization via abnormality-enhanced classification. In *Proc. DSN*, 2012.

[5] H. Chen, G. Jiang, and K. Yoshihira. Failure detection in large-scale internet services by principal subspace mapping. *IEEE Trans. Knowl. Data Eng.*, 2007.

[6] I. Cohen, M. Goldszmidt, T. Kelly, and J. Symons. Correlating instrumentation data to system states: A building block for automated diagnosis and control. In *Proc. OSDI*, 2004.

[7] G. Cormode and M. Garofalakis. Sketching probabilistic data streams. In *SIGMOD*, 2007.

[8] M. Gabel, A. Schuster, R.-G. Bachrach, and N. Bjorner. Latent fault detection in large scale services. In *Proc. DSN*, 2012.

[9] C. Huang, I. Cohen, J. Symons, and T. Abdelzaher. Achieving scalable automated diagnosis of distributed systems performance problems. Technical report, HP Labs, 2007.

[10] M. Isard. Autopilot: automatic data center management. *SIGOPS Oper. Syst. Rev.*, 2007.

[11] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad: distributed data-parallel programs from sequential building blocks. In *Proc. EuroSys*, 2007.

[12] W. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. In *Conference in modern analysis and probability (New Haven, Conn., 1982)*, Contemporary Mathematics. 1984.

[13] M. P. Kasick, J. Tan, R. Gandhi, and P. Narasimhan. Black-box problem diagnosis in parallel file systems. In *Proc. FAST*, 2010.

[14] S. Kavulya, S. Daniels, K. Joshi, M. Hiltunen, R. Gandhi, and P. Narasimhan. Draco: Statistical diagnosis of chronic problems in large distributed systems. In *Proc. DSN*, 2012.

[15] S. Kavulya, R. Gandhi, and P. Narasimhan. Gumshoe: Diagnosing performance problems in replicated file-systems. In *Proc. SRDS*, 2008.

[16] D. Keren, I. Sharfman, A. Schuster, and A. Livne. Shape sensitive geometric monitoring. *Knowledge and Data Engineering, IEEE Transactions on*, 2012.

[17] S. Muthukrishnan. Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science*, 2005.

[18] A. J. Oliner, A. Aiken, and J. Stearley. Alert detection in system logs. In *Proc. ICDM*, 2008.

[19] D. Pelleg, M. Ben-Yehuda, R. Harper, L. Spainhower, and T. Adeshiyan. Vigilant: out-of-band detection of failures in virtual machines. *SIGOPS Oper. Syst. Rev.*, 2008.

[20] I. Sharfman, A. Schuster, and D. Keren. A geometric approach to monitoring threshold functions over distributed data streams. *TODS*, 2007.

[21] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan. Detecting large-scale system problems by mining console logs. In *Proc. SOSP*, 2009.

[22] S. Zhang, I. Cohen, M. Goldszmidt, J. Symons, and A. Fox. Ensembles of models for automated diagnosis of system performance problems. In *Proc. DSN*, 2005.

Predicting the Fundamental Value of Financial Assets by Ridge Regression

Michael Kamp **Mario Boley** **Thomas Gärtner**

Fraunhofer IAIS
Schloss Birlinghoven
53754 Sankt Augustin, Germany
{firstname.lastname}@iais.fraunhofer.de

Abstract

The so-called fair value of a financial asset is the crucial indicator in fundamental investment strategies. At the same time it is hard to assess, because it depends on corporate earnings as well as macro-economic figures, both of which are confidential until the end of fixed accounting periods. We show that ridge regression models based on stock price and price correlation features can estimate fair values competitively with an insider analyst. This analyst is always informed about up-to-date earnings and macro-economic figures, from which she can project the value of the complete accounting period. As opposed to this insider data, the correlation features are publicly available in real-time. Moreover, due to its simplicity, linear regression models are highly scalable to large amounts of price data. The method is evaluated with data of two different stock markets (S&P100 and the German DAX30) from a period of five years.

1 Introduction

Fundamental stock investment [4, 5, 16], also referred to as value investment or "stock picking", subsumes strategies that base investment decisions on quantities that are not directly influenced by the market—so-called fundamentals (as opposed to technical stock investment that is solely based on price trend analysis). Among fundamental strategies, it is common to assess the *fair value* of a stock share as the product of the annual earnings per share of the associated business entity and the "fair-P/E-ratio". That is, the ratio of price to earnings that is considered adequate in the current macro-economic market environment (see, e.g., [6, 17]), as captured, e.g., by current rates of GDP-growth, inflation, and key interest.

Naturally, investment decisions are ideally based on up-to-date fair values. These are, however, inherently hard to assess. Not only are future earnings hard to predict. Even past and present figures are only published at the end of fixed accounting periods and are confidential up to this point. Similarly, most macro-economic figures are only reported retrospectively for past periods. Thus, market analysts have to rely on extrapolations based on published but deprecated data taken from past balance sheets and statistics. On the other hand, assuming a somewhat efficient stock market [18, 19], current price development should reflect fundamental information as it becomes available [1, 20]. Moreover, it is a common rationale in Economics [2] as well as in Machine Learning [26] that it can improve the inference to jointly consider prices of stocks that are empirically correlated. Therefore, in this paper *we investigate the quality of estimating the current fair value of stocks based on publicly available market data*, i.e., stock prices and their correlations. As opposed to confidential current balance sheets, stock prices and their correlations are publicly available in real-time. Just as human analysts usually consider price developments of stocks from the same market segment as indicative, we consider prices of all other stocks in the market weighted by their correlation to the target stock with respect to different time resolutions. The effect of enriching the price representation of the fair value of a single stock by correlation weighted prices of all other stocks in the market is illustrated in Figure 1.

We test supervised regression models based on this feature space against a baseline corresponding to a business and market insider. This insider always knows current earnings and can extrapolate from them to the complete accounting period. We evaluate our model on two different stock markets (S&P100 and German DAX30) based on price, earnings, and macro-economic data from between 2007 and 2011. As the fair value is neither a published statistic nor unambiguously defined, we rely on a data-driven operationalization as ground truth. This operationalization is based on the true

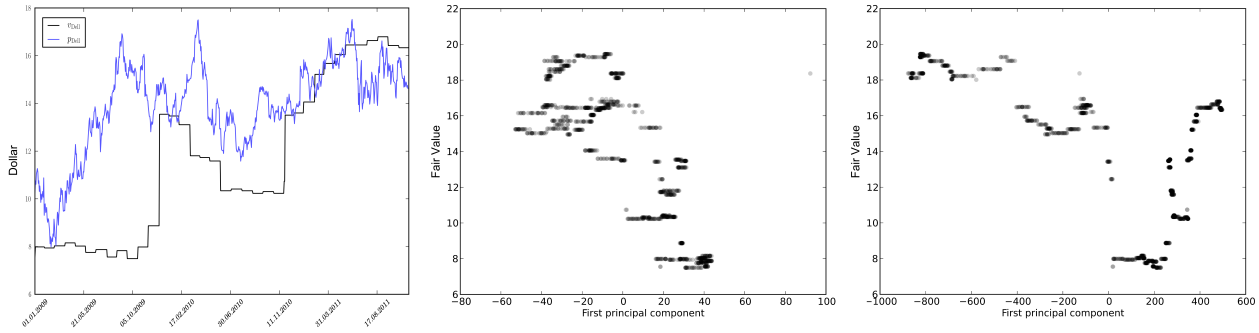


Figure 1: Shares of Dell inc. (NASDAQ:DELL). *Left:* price (blue line) and fair value (black line) development. *Center:* fair value against first principal direction of price, short-, middle- and long-term average price (11, 50 and 200 days). *Right:* fair value against first principal direction of features as in center enriched with all other S&P100 prices weighted by their correlation to Dell.

earning per share (EPS) and a fair market ratio of price to earnings (fair P/E-ratio) that is modeled as a linear function of GDP growth, inflation, and key interest rate. Our experiments show that the insider information can on average be replaced by monitoring price data and correlations: For 50% of the investigated stocks the market based estimations are not worse than projections performed by an insider. For 90% they do not have a worse error than two times the error of the insider estimations. Naturally, the estimation quality can be bad in cases of extreme developments of either price or earning that are not reflected by stocks that were previously correlated.

2 Fair Values of Stock Shares

In this section we formalize the task of fair value estimation from public market prices. A methodological challenge that we face is that the notion of fair value of a stock is neither a published figure nor is it unambiguously defined. Therefore, we give an operationalized definition based on a model that can be fitted to real market data. For the empirical evaluation in Section 5 this implies that some of the available data has to be sacrificed in order to define an independent ground truth in addition to the usual training/test split.

2.1 Price and Earnings Time Series In more detail, we deal with financial time series that correspond to share prices, corporate earnings, and global market figures, as well as dependent figures derived from them. We consider a finite discrete total time horizon from $\{1,\dots,T\}$ where every point in time corresponds to a single day, although all presented methods can be straightforwardly applied to finer time resolutions. Hence, all basic **time series** we consider are real-valued discrete sequences $f\colon\{1,\dots,T\}\to\mathbb{R}$ of an identical finite length $|f|=T$. For a **time interval** $[i,j]$ with $i,j\in\mathbb{N}$ and $i<j\leq|f|$, the **time window** from i to j is the time series $f[i,j]\colon\{1,\dots,j-i\}\to\mathbb{R}$ defined by $f[i,j](t)=f(j-i+t)$. Let S denote the set of **stocks** under consideration. For a stock $s\in S$, the **price** of one of its shares is a real-valued discrete time series, that we denote by $p_s:\{1,\dots,T\}\to\mathbb{R}$. Similarly, by $e_s\colon\{1,\dots,T\}\to\mathbb{R}$ we denote the **earnings per share** of the business entity associated to s. Note that in fundamental analysis one is interested in earnings that are aggregated for some accounting period (usually per annum or quarter). Hence, in contrast to the price series, which change almost in real time, the earnings series are piece-wise constant with piece-length equal to the length of the accounting period. Combining both of theses series, we can define the **price-earnings ratio (P/E-ratio)** of a stock s as the time series given by $r_s(\cdot)=p_s(\cdot)/e_s(\cdot)$. In fundamental analysis, this ratio is compared to a market dependent reference value, which we refer to as **fair P/E-ratio**, and denote $r_{\text{fair}}:\{1,\dots,T\}\to\mathbb{R}$. For a stock s, the condition $r_s(t)>r_{\text{fair}}(t)$ indicates overvaluation at time t, the inverse indicates undervaluation. Correspondingly, we define the **fair value** of stock shares as the product of the company's earnings and the current fair P/E-ratio, $v_s(\cdot)=e_s(\cdot)r_{\text{fair}}(\cdot)$.

2.2 Fair P/E Ratio The fair P/E-ratio reflects an expectation on the value development of a stock given certain earnings, depending on the current global market situation. This global situation can be represented by the cost of borrowing capital, i.e., the **key interest rate**, the cost of holding capital and not investing it, i.e., the **inflation rate**, and the general prospects of the market, i.e., the **GDP growth** rate [3]. We denote the time series of these quantities as δ_{int}, δ_{ifl}, and δ_{grw},

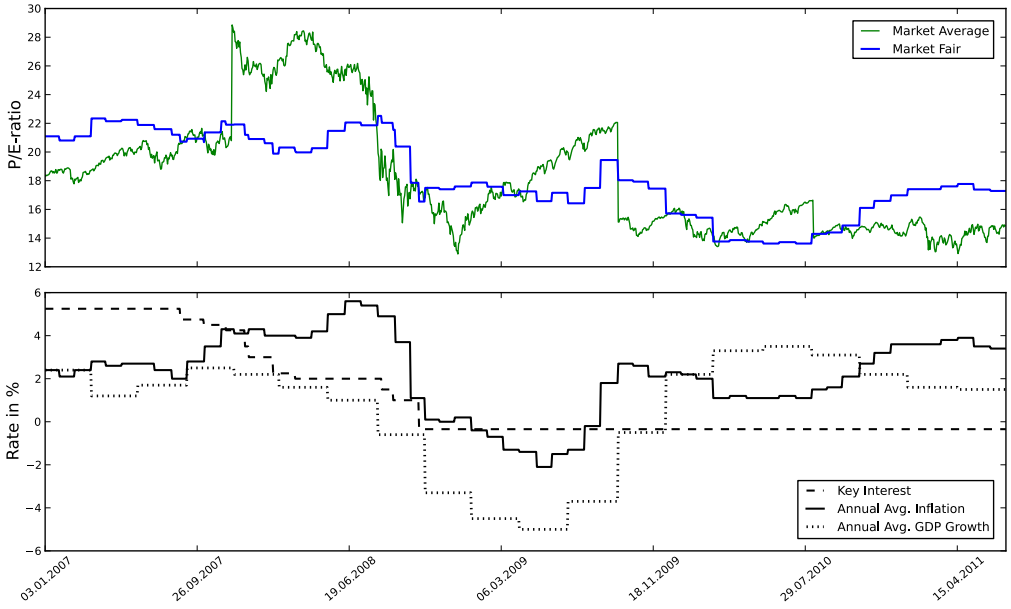


Figure 2: Average P/E-ratio of S&P100 and approximation of fair P/E-ratio (top) based on the US Federal Funds target rate as key interest rate, US inflation rate and US GDP growth rate as published by Trading Economics [23] (bottom); corresponding weights are $w_{\text{int}} = 0.98$, $w_{\text{ifl}} = -0.75$, and $w_{\text{grw}} = 1.02$.

respectively. While the key interest rate can change at arbitrary points in time and is constant until the next change, the inflation and growth series describe aggregated values over some fixed reporting periods (usually per month in case of inflation and per quarter in case of GDP growth). Thus, similar to the earnings, also these two series are piece-wise constant with respect to the length of their associated reporting period. Since the effect of these three rates on capital is additive, we assume the fair P/E ratio to be a linear function of them. Moreover, assuming that the **market average P/E ratio**, $r_S(\cdot) = \sum_{s \in S} r_s(\cdot)/|S|$, is governed by the fair P/E ratio plus normally distributed noise, we can approximate the time series of fair P/E ratios by a least-square fit. That is, we define the **approximated fair P/E-ratio** for fitting interval $[1, j]$ with $j \leq T$ as

$$\tilde{r}^j_{\text{fair}}(\cdot) = w^*_{\text{int}}\delta_{\text{int}}(\cdot) + w^*_{\text{ifl}}\delta_{\text{ifl}}(\cdot) + w^*_{\text{grw}}\delta_{\text{grw}}(\cdot)$$

such that $w^* = (w^*_{\text{int}}, w^*_{\text{ifl}}, w^*_{\text{grw}})$ minimizes the squared difference to the average market P/E ratio

$$\sum_{t=1}^{j} |(w_1\delta_{\text{int}}(t) + w_2\delta_{\text{ifl}}(t) + w_3\delta_{\text{grw}}(t) - r_S(t)|^2$$

over all $w = (w_1, w_2, w_3) \in \mathbb{R}^3$. As an illustrative example, figure 2 shows the resulting approximated fair P/E value for the Standard & Poor's 100 (S&P 100) index together with the different time series that it is based on. Correspondingly, we define the **approximated fair value** of a stock s for fitting interval $[1, j]$ as $\tilde{v}^j_s(\cdot) = e_s(\cdot)\tilde{r}^j_{\text{fair}}(\cdot)$.

2.3 Estimation Problem As opposed to the latent and unobservable concept of a fair P/E ratio, the definition of approximated fair value provides us an operationalization of fair values that can indeed be computed using publicly available data—in retrospect once earnings and global market figures have been published. For this purpose we use the approximated fair value fitted to the complete data up to time T. Dropping the superscript, we denote the resulting time series by $\tilde{v}_s$. Moreover, from the perspective of a real market situation, suppose that the last earnings corresponding to $s \in S$ have been published at time q and that the length of the accounting period is c with $q + c < T$. This naturally defines $[1, q]$ as a training period and $[q, q + c]$ as test period. The formal problem statement of estimating fair values during an accounting period by publicly available data can then be written as follows.

Definition 1. (Fair Value Estimation Problem) ***Given*** *time series of prices of all stocks in the mar-*

ket $\{p_u : u \in S\}$ *as well as all published earnings* $\{e_u[1,q] : u \in S\}$ *and rates of key interest* $\delta_{\text{int}}[1,q]$*, inflation* $\delta_{\text{ifl}}[1,q]$*, and GDP growth* $\delta_{\text{grw}}[1,q]$*,* ***estimate*** *the approximated fair values of a stock for the current accounting period* $\tilde{v}[q+1, q+c]$*.*

3 Price- and Correlation-based Prediction

To solve the fair value estimation problem, we now describe an estimation technique based on public stock prices and their correlations. Stock prices are an indicator for corporate earnings because, as stated in the introduction, there is a dependency between the earnings of a company and its stock price. Similarly, the relative behavior of stock prices indicate the current global market situation. However, there is substantial evidence that the stock market is not perfectly efficient (see, e.g., [22]), and speculation and irrational behavior of investors adds noise to the prices, making it hard to infer the fair value from prices alone.

In order to produce more reliable estimates, we propose to also factor in the prices of other stocks in the market weighted by their relative importance to the target stock. As a second technique to increase the robustness of the estimations, we propose to measure prices and correlations according to different temporal resolutions. It is a standard in fundamental as well as in technical investment to not solely rely on the momentary picture of the last available closing price. Since stock prices are highly volatile, one also considers smoothed prices given by moving averages of different time window lengths.

Formally, let $l \in \mathbb{N}$ be some interval length. Denoting by $\overline{f} = \sum_{t=1}^{|f|} f(t)/|f|$ the **average** value of a finite time series f, this gives rise to the smoothed time series of **moving average prices**

$$a_s^l(t) = \overline{p_s[t-l,t]} \tag{3.1}$$

for all stocks $s \in S$. Similarly, we are interested in the correlation between prices of two stocks for a fixed time window. The **sample Pearson correlation coefficient** $\text{cor}(f,g)$ between two finite time series f, g with $|f| = |g|$ is defined by $\text{cor}(f,g) = \text{cov}(f,g)/(\text{std}(f)\text{std}(g))$ where

$$\text{cov}(f,g) = \frac{1}{|f|-1} \sum_{t=j-i}^{j} (f(t) - \overline{f})(g(t) - \overline{g})$$

and $\text{std}(f) = \sqrt{\sum_{t=j-i}^{j}(f(t)-\overline{f})^2/(|f|-1)}$ denote the sample co-variance and the sample standard deviation. As for the stock prices, given an interval length $l \in \mathbb{N}$, we define the **moving correlation** between two stocks $s, u \in S$ as

$$c_{s,u}^l(t) = \text{cor}(p_s[t-l,t], p_u[t-l,t]) \ . \tag{3.2}$$

With this we can go ahead and construct a feature representation for each stock and each moment in time. For a time interval for which business earnings are already published, this representation can then be used together with the operationalization of fair values in order to fit an estimation model. Following our earlier described intuition, we want to represent a stock s at time t by its price as well as by the prices of all other stocks in the market weighted by their correlation with s. Also, we want to provide this information with respect to different time resolutions in order to capture short term and long term relations at the same time. In accordance with economic practice (see, e.g., [9]), for each moment in time t we consider a **short-term**, a **mid-term**, and a **long-term time window**, looking back 11, 50, and 200 days, respectively. Putting everything together, we can finally define the **feature representation** $\varphi_s \colon \mathbb{N} \to \mathbb{R}^d$ for a stock $s \in S$ as

$$\varphi_s(t) = \underset{u \in S}{\circ}(p_u, a_u^{11} c_{u,s}^{11}, a_u^{50} c_{u,s}^{50}, a_u^{200} c_{u,s}^{200}) \ .$$

with definitions of moving average prices and moving correlations as given in equations (3.1) and (3.2).

Suppose the current accounting period for the business entity associated with stock s started at time $j+1$, and that all previous earnings are published. Then, for a **training interval** $[i,j]$ with $i > 200$, we have a training set

$$\{(\varphi_s(t), \tilde{v}_s(t)) : i \le t \le j\} \subseteq \mathbb{R}^{4|S|} \times \mathbb{R}$$

that can be used to fit an estimation model $f_s : \{1, \ldots, T\} \to \mathbb{R}$. For this task, in principle, any regression technique can be used. In this paper we perform our experiments with Ridge Regression, i.e., regularized least squares regression, because of its simplicity and scalability. This means that the **price and correlation-based model** for estimating the fair value of stock $s \in S$ is given as $f_s^{\text{pc}}(\cdot) = \langle w_s^*, \varphi_s(\cdot) \rangle$ where $w_s^* \in \mathbb{R}^{4|S|}$ solves

$$\min_{w \in \mathbb{R}^{4|S|}} \sum_{t=i}^{j} |w^T \varphi_s(t) - \tilde{v}_s(t)|^2 + \nu \|w\|_2^2 \ .$$

with some positive regularization parameter $\nu \in \mathbb{R}_+$.

4 Prediction based on Insider Information

The estimation model based on price and correlation features relies on the idea that confidential information can be compensated by publicly available market data.

In order to assess the validity of this approach we have to contrast it with a baseline that has, to a certain degree, access to this confidential information. To this end we design an estimation method that corresponds to a business and market insider who is informed about up-to-date corporate earnings as well as global market conditions on a daily basis. For the global market condition this information is given in the form of the approximated fair P/E ratio $\tilde{r}_{\text{fair}}^{q+c}$ fitted up to the end of the current accounting period. For the corporate earnings, at time $q+i$, the insider has access to the series of **daily earnings**, denoted $d_s[1, q+i]$. Hence, the **insider information based model** is

$$f_s^{\text{id}}(\cdot) = \tilde{e}_s(\cdot)\, \tilde{r}_{\text{fair}}^{q+c}(\cdot) \ , \tag{4.3}$$

where $\tilde{e}_s$ is linear **earnings projection** from the daily earnings to the end of the accounting period, i.e., $\tilde{e}_s(q+i) = c \sum_{t=q}^{q+i} d_s(t)/i$. Note that, while at the end of the accounting period, $\tilde{e}_s(q+c) = e_s(q+c)$, in general $\tilde{e}_s(q+i) \neq e_s(q+i)$, due to unequally distributed daily earnings during the accounting period. The expected error of this projection exactly depends on this distribution of daily earnings.

Lacking data of real daily earnings, it is a conservative model to assume uniformly distributed earnings, only disturbed by a small Gaussian perturbation. That is, for all points in time $i \in [q+1, < q+c]$, the daily earnings are set to

$$d_s(q+i) = e_s(q+c)/c + \epsilon_{q+i}^s \ ,$$

with independent errors $\epsilon_t^s \sim \mathcal{N}(0, \sigma_s)$ and $d_s(q+c)$ equal to $e_s(q+c) - \sum_{t=q}^{q+c-1} d_s(t)$ in order to ensure consistency with the aggregated earnings $e_s(q+c)$. It remains to set the standard deviation σ_s. Given the empirical standard deviation of the global earnings $\text{std}(e_s)$, one can ensure that the earnings projections are distributed in the same way as the global earnings by setting $\sigma_s = \text{std}(e_s)/c$. Indeed, with this choice, we have $\text{std}(\tilde{e}_s) = c\sigma_s = \text{std}(e_s)$.

Given this model, the earnings projections for $s \in S$ at time $q+i$ can be written as

$$\tilde{e}_s(q+i) = e_s(q+c) + \frac{j}{i} \sum_{t=q}^{q+i} \epsilon_t^s \ .$$

Hence, the expected error of these projections decreases linearly with i approaching c. See figure 3 for an illustrative estimate of the aggregated earnings in comparison to the true aggregates based on a realization of the daily earnings model.

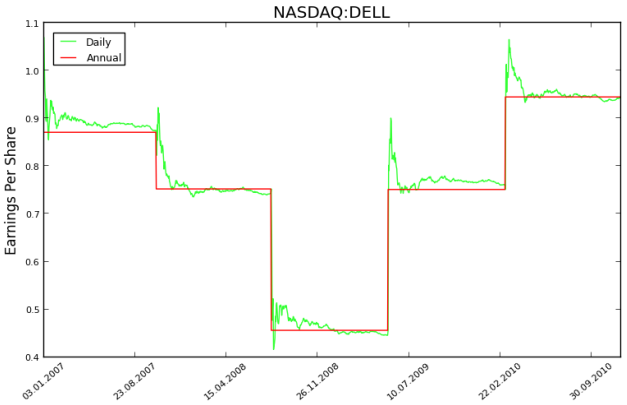


Figure 3: Earnings projection from perturbed daily earnings for Dell Inc. (NASDAQ:DELL).

5 Experiments

We now present an empirical comparison of the price- and correlation-based prediction to the prediction based on insider information. Therefore, experiments have been performed on two independent datasets, i.e., the US Standard & Poor's 100 index (S&P 100) as well as the German DAX 30 in the time from 1.1.2007 to 31.12.2011. For both indexes, the data includes daily stock prices p_s [14] and annually published earnings per share e_s of the corresponding companies [8]. Furthermore, macro-economics, i.e., key interest rate, δ_{int}, inflation rate, δ_{ifl} and GDP growth rate, δ_{grw} are included. In case of the S&P 100, the key interest rate is the Federal Funds Target Rate US [13], while for DAX 30 it is the ECB key interest rate [11]. For S&P 100 and DAX 30, the US American, respectively German quarterly GDP growth rate and monthly inflation rate are provided [23].

We train the price- and correlation-based prediction method f^{pc} on a training interval of length W_{train} and not on the entire data known so far, in order to account for the fact that the properties of financial markets alter over time, i.e, the fair value is affected by concept drifts. On the other hand, training on less than one year involves the risk overfitting to annual business cycles. A training interval of $W_{\text{train}} = 500$ days, i.e., two years, complies with both constraints and is thus chosen for the experiments.

The once trained model predicts the fair value for the subsequent accounting interval. The applied accounting interval is one quarter, i.e., $c = 62$ days. The macro-economic figures are available at least quarterly (GDP-growth rate), but the employed datasets only includes annual earnings. Still, a quarter is a realistic accounting interval, because earnings have to be published

quarterly in the US and while they only have to be reported annually in Germany, most German companies voluntarily publish their quarterly earnings. After each prediction, the training interval is shifted by c days and the model is retrained. The prediction based on insider information is performed on the exact same accounting intervals.

The performance of the price- and correlation-based prediction method f^{pc} is compared to the prediction based on insider information f^{id} in terms of the root mean squared error ($\epsilon_{\text{rmse}}(\cdot)$). We average the ϵ_{rmse} over all test intervals. Both, f^{pc} and f^{id}, predict the fair value of one stock. Thus, we run both methods for each stock and compare their average performance on all stocks as well as their performance on individual shares.

The prediction based on insider information uses an estimation of the earnings per share based on noisy daily earnings data. To ensure that the prediction is not influenced by the particular noise, we generate 10 different daily earnings time series for each stock and average the prediction results .

On average, over all test intervals and all stocks, both methods achieve comparable prediction results, i.e., the root mean squared error of the prediction method is $\epsilon_{\text{rmse}}(f^{\text{pc}}) = 16.86$, while the baseline has an error of $\epsilon_{\text{rmse}}(f^{\text{id}}) = 12.84$. In table 1, we compare both methods in detail. These results show that for half of the stocks in the combined dataset, the prediction method based on publicly available price data outperforms or performs as well as the business insider. For nearly 2/3 of stocks, the error of f^{pc} exceeds the error of f^{id} by no more than 50%. Furthermore, on over 90% of stocks, the error of f^{pc} is less than twice as large as the error of f^{id}.

Furthermore, the average fair value of all stocks over time, as well as prediction and baseline, are defined as

$$\begin{aligned}\tilde{v}_{\text{avg}} &= \text{avg}\{\tilde{v}_s : \forall s \in S\} \\ f^{\text{pc}}_{\text{avg}} &= \text{avg}\{f^{\text{pc}}_s : \forall s \in S\} \\ f^{\text{id}}_{\text{avg}} &= \text{avg}\{f^{\text{id}}_s : \forall s \in S\} \ .\end{aligned}$$

The average results are depicted in figure 4. The error of the average prediction is now lower than the error of the average baseline, i.e., $\epsilon_{\text{rmse}}(f^{\text{pc}}_{\text{avg}}) = 4.88$, while $\epsilon_{\text{rmse}}(f^{\text{id}}_{\text{avg}}) = 7.58$. Thus, on average the price- and correlation-based prediction estimates the fair value even better than the prediction based on insider information.

At the same time, for several stocks f^{pc}, based on public price data, performs significantly worse than f^{id}, that has access to unpublished corporate and macroeconomic figures. We want to analyze the stocks on which f^{pc} performed worst in comparison to the one it performed best. The largest prediction error on S&P 100 occurred for Apple Inc with $\epsilon_{\text{rmse}}(f^{\text{pc}}_{Apple}) = 94.02$, for which the insider information based method has an error of $\epsilon_{\text{rmse}}(f^{\text{id}}_{Apple}) = 40.65$. On the same dataset, Dell Inc performed best with $\epsilon_{\text{rmse}}(f^{\text{pc}}_{Dell}) = 1.91$ compared to $\epsilon_{\text{rmse}}(f^{\text{id}}_{Dell}) = 2.32$. In figure **??**, we plotted f^{pc}, f^{id} and v for Apple Inc and Dell Inc.

Δ	1	1.1	1.5	2
S&P 100	54.4%	62%	70.7%	94.6%
DAX 30	32.1%	35.7%	46.4%	78.6%
Total	49.2%	55.8%	65%	90.8%

Table 1: Relative number of stocks for which the root mean square error of price and correlation based prediction does not exceed the error of the insider prediction by more than a factor of Δ, i.e., $\epsilon_{\text{rmse}}(f^{\text{pc}}) \leq \Delta\, \epsilon_{\text{rmse}}(f^{\text{id}})$.

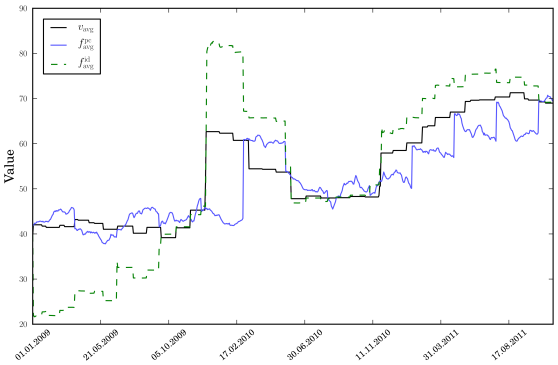


Figure 4: Plot of $\tilde{v}_{\text{avg}}$, $f^{\text{pc}}_{\text{avg}}$ and $f^{\text{id}}_{\text{avg}}$ over the complete testing period. Note that the first 500 days of the dataset are only used for training and thus not included in this plot.

While Dell had a moderate earnings development between 2007 and 2011 ($\sigma_{Dell} = 0.187$), Apples earnings increased by more than 600% in this period ($\sigma_{Apple} = 9.84$). This increase is largely determined by the success of the products released between 2007 and 2011, i.e., the iPhone (released in January, 2007), iPhone 3G (released in July, 2008), iPhone 3GS (released in June, 2009) and the iPad (released March, 2010). Especially the success

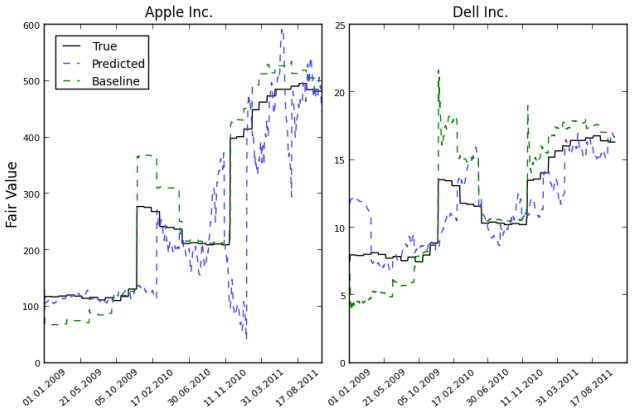


Figure 5: Plot of f^{pc} (Prediction), f^{id} (Baseline) and v (True) for Apple Inc. (left) and Dell Inc. (right) from 1.1.2009 to 31.12.2011, both taken from S&P 100. The first 500 days are omitted, because f^{pc} is trained on this period and no prediction is available.

of the iPhone can be termed a singular event, because it was highly unexpected. Also for the later product releases, the overall success exceeded investors expectations [15].

While on S&P 100 Apple performed worst, on DAX 30, the largest error occurred for Volkswagen with $\epsilon_{\text{rmse}}(f^{\text{pc}}_{Dell}) = 97.89$. The Volkswagen stock price exhibited extreme movements in October 2008 due to a short squeeze, i.e., a high amount of short sellers facing a strong upward trend. This short squeeze occurred, because Porsche AG vainly attempted to take over Volkswagen while most investors sold short on Volkswagen shares. The price increased by a factor of 5 for one day, falling back to the initial price shortly after [21]. Furthermore, Volkswagen has an even higher volatility in earnings ($\sigma_{Apple} = 13.54$), due to very low earnings in 2009 caused by the global Financial Crisis and very high earnings in 2011 partially related to accounting benefits from the failed takeover attempt by Porsche [24].

While this detailed analysis of three stocks is not sufficient to draw a final conclusion on the reasons for the difference in performance of the price- and correlation-based prediction method, it indicates that singular events are hard to anticipate and that highly volatile corporate earnings are harder to predict. At the same time the analysis shows that for shares with moderate earnings and price development, the fair value can be estimated with high accuracy.

6 Discussion

We investigated to what degree the fair value of stock shares can be estimated using only publicly available price data from stock markets. This problem is motivated by the fact that current fair values depends on confidential fundamental information that is only published in the future. While there has been work in predicting prices and returns (e.g, [12, 25, 26]), to the best of our knowledge, the specific task of fundamental-based fair value estimation has not yet been addressed within Machine Learning research.

Our experiments with two stock markets showed that confidential information can indeed be replaced by public price data in assessing the fair value of a stock: for the majority of tested stocks the estimations based on public data have not been worse than the estimations based on insider information of corporate earnings. For future research it is important to re-evaluate these initial positive findings on further markets. Also, while the synthetic daily earnings used for the insider baseline were generated very conservatively, refined experiments with real daily earnings could further increase confidence in this comparison.

In addition to our positive results for the majority of tested stocks, our experiments also revealed some limits of the approach for specific hard stocks. For singular events in the market and highly volatile corporate profits, the reliability of the price/earnings links drops significantly. One approach to address this limitation could be to incorporate news data into the estimation mechanism in order to detect such events. A promising news source for this purpose is the online short message service Twitter, because its messages are extremely timely and are moreover easily accessible via a public API. The Twitter stream is increasingly often studied for various Machine Learning tasks. Examples include the prediction of Influenza outbreaks [10] and even specifically modeling of the stock market sentiment [7].

References

[1] J.S. Abarbanell. Do analysts' earnings forecasts incorporate information in prior stock price changes? *Journal of Accounting and Economics*, 14(2):147–165, 1991.

[2] Andrew W. Alford. The effect of the set of comparable firms on the accuracy of the price-earnings valuation. *Journal of Accounting Research*, 30(1): 94–108.

[3] C. Asness. Fight the fed model: the relationship between stock market yields, bond market yields, and future returns. *Journal of Portfolio Management*.

[4] S. Basu. Investment performance of common stocks in relation to their price-earnings ratios: A test of the efficient market hypothesis. *The Journal of Finance*, 32(3):663–682, 1977.

[5] Leopold A. Bernstein. In defense of fundamental investment analysis. *Financial Analysts Journal*, 31(1):57–61.

[6] Christopher M Bilson, Timothy J Brailsford, and Vincent J Hooper. Selecting macroeconomic variables as explanatory factors of emerging stock market returns. *Pacific-Basin Finance Journal*, 9(4): 401–426, 2001.

[7] Johan Bollen and Huina Mao. Twitter mood as a stock market predictor. *IEEE Computer*, 44(10): 91–94, 2011.

[8] comdirekt bank AG. comdirect Bank.

[9] Credit Suisse. *Technical Analysis - Explained*. Credit Suisse Group AG.

[10] Aron Culotta. Lightweight methods to estimate influenza rates and alcohol sales volume from Twitter messages. *Language Resources and Evaluation, Special Issue on Analysis of Short Texts on the Web*, 2012. to appear.

[11] ECB: European Central Bank. European Central Bank.

[12] David Enke and Suraphan Thawornwong. The use of data mining and neural networks for forecasting stock market returns. *Expert Syst. Appl.*, 29(4): 927–940, 2005.

[13] Federal Open Market Committee. Federal Reserve.

[14] Google Inc. Google Finance.

[15] J. Graham. Apple buffs marketing savvy to a high shine. *USA Today*, 8, 2007.

[16] J. Lakonishok, A. Shleifer, and R.W. Vishny. Contrarian investment, extrapolation, and risk. *Journal of Finance*, 49:1541–1578, 1994.

[17] Martin Lettau, Sydney C. Ludvigson, and Jessica A. Wachter. The declining equity premium: What role does macroeconomic risk play? *Review of Financial Studies*, 21(4):1653–1687, 2008.

[18] B.G. Malkiel. The efficient market hypothesis and its critics. *Journal of Economic Perspectives*, pages 59–82, 2003.

[19] B.G. Malkiel and E.F. Fama. Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, 25(2):383–417, 2012.

[20] R. Morck, B. Yeung, and W. Yu. The information content of stock markets: Why do emerging markets have synchronous stock price movements? *Journal of Financial Economics*, 58(1): 215–260, 2000.

[21] L. Story, M. de la Merced, and C. Dougherty. Panicked traders take vw shares on a wild ride. *New York Times*, 28, 2008.

[22] L.H. Summers. Does the stock market rationally reflect fundamental values? *The Journal of Finance*, 41(3):591–601, 2012.

[23] Trading Economics. Trading Economics, New York City.

[24] Volkswagen Group. Volkswagen group annual financial statement: Result of operations. 28, 2011.

[25] SY Wang W Huang, Y Nakamori. Forecasting stock market movement direction with support vector machine. *Computers & Operations Research*, 32(10):2513–2522, 2005.

[26] A. Wilson and Z. Ghahramani. Generalised wishart processes. In *Proceedings of the Twenty-Seventh Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-11)*.

Beating Human Analysts in Nowcasting Corporate Earnings by using Publicly Available Stock Price and Correlation Features

Michael Kamp, Mario Boley, Thomas Gärtner
{firstname.lastname}@iais.fraunhofer.de
Fraunhofer IAIS
Schloss Birlinghoven
53754 St.Augustin, Germany

***Abstract*—Corporate earnings are a crucial indicator for investment and business valuation. Despite their importance and the fact that classic econometric approaches fail to match analyst forecasts by orders of magnitude, the automatic prediction of corporate earnings from public data is not in the focus of current machine learning research. In this paper, we present for the first time a fully automatized machine learning method for earnings prediction that at the same time a) only relies on publicly available data and b) can outperform human analysts. The latter is shown empirically in an experiment involving all S&P 100 companies in a test period from 2008 to 2012. The approach employs a simple linear regression model based on a novel feature space of stock market prices and their pairwise correlations. With this work we follow the recent trend of nowcasting, i.e., of creating accurate contemporary forecasts of undisclosed target values based on publicly observable proxy variables.**

***Keywords*-Machine Learning; Economic Forecasting**

I. Introduction

Corporate earnings are a crucial signal for investment and business valuation as they are a key indicator of a company's success and the development of its equity [1, 2]. They are, however, only published at the end of fixed accounting periods and are confidential up to this point. Hence, there is a great interest in accurate estimations of this value based on publicly available data. While human analysts provide such estimations with reasonable accuracy [3], automatized methods based on classic econometric and statistical approaches fail to reach the quality of human experts by orders of magnitude [4]. Despite this gap, the earnings prediction problem is not in the focus of modern machine learning research. There are exceptions, which, however, use undisclosed variables for prediction (e.g., [5, 6]) or focus on different objectives such as earning surprises or direct stock price prediction (e.g., [7, 8, 9, 10, 11]).

In this paper, we present for the first time a fully automatized machine learning method for earnings prediction based on completely publicly available data that can outperform human analysts. This is shown empirically in an experiment with the S&P 100 companies in a test period from 2008 to 2012. In this period a total of 1600 earnings forecasts have to be provided, for which the proposed method outperforms the forecasts of human experts on average as well as on a majority of individual stocks. The data required to employ the method are the publicly available stock price time series of the target company along with the price series of as many as possible other reference companies.

In particular, the approach employs a simple linear regression model based on a novel feature space of stock market prices and their pairwise correlations. The rationale for this feature space is that the stock price of a company is a good proxy for its earnings [12, 13]. Furthermore, the earnings of a company depend on other market participants, e.g., its competitors or component suppliers. These rather durable interrelationships can be modeled using linear weights. However, stock prices can behave independently from the companies' performance, e.g., because of speculation or transient trends [14]. In our model, the relation between the target company's earnings from the stock price of another company is credible if their stock prices are congruent, which is measured by the proposed correlation features. The impact of enriching stock prices by correlation features is also illustrated in figure 1, which shows for the target stock of CISCO Systems Inc. and Standard & Poor's 100 index as reference stocks that the enriched feature space has a higher capability of separating high from low earning quarters. The significance of this feature augmentation is also investigated more rigorously on a further extended experiment involving 5200 earnings forecasts for 200 companies. Prediction methods that only use basic price features are substantially outperformed by our price and correlation based method.

With this work we follow the recent trend of nowcasting [15, 16, 17], i.e., of creating accurate contemporary forecasts of undisclosed target values based on publicly observable proxy variables. The idea is that, while the future of complex systems is hard to predict, their present can be "predicted" reasonably well using the massive amounts of data available nowadays. In this paper, the aggregated earnings of a company in a reporting period are the undisclosed target values that are only published at the end of each business quarter. The stock prices act as observable proxy variables. Note that, while our task of nowcasting the aggregated earnings implies not only estimating the current state but also a mild true

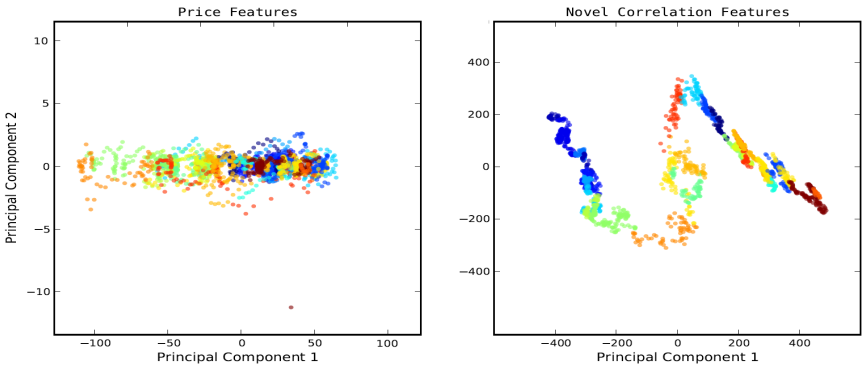


Figure 1. Principle Component Analysis (PCA) of features for target stock CISCO Systems, Inc. (NASDAC:CSCO). Each point represents a trading day in the time between 2004 and 2012. The data is represented as simple price features, i.e., stock prices of all stocks in the market (left) and in the proposed correlation feature space (right). The color indicates the earnings per share (EPS) value of CISCO Systems. The 2D embedding of the data using the first two principle components indicates that the proposed feature space improves the separability of the data points according to their EPS value significantly.

forecasting component, this is also true for other successful nowcasting applications [18, 19]. In particular, towards the end of an accounting period the earnings prediction problem approaches a pure nowcasting setting.

In the remainder of the paper, in section II, we define the correlation feature space as well as the proposed nowcasting method. In section III, we describe the experimental setup followed by the presentation and discussion of the empirical results. Finally, in section IV, we summarize our method and the empirical evaluation and conclude by giving an outlook on future work.

II. Earnings Forecasts

This section describes the correlation feature space, formalizes the task of earnings prediction from public stock prices and presents the employed prediction method.

A. Preliminaries

In the following, we always consider a dedicated target stock s^* from fixed set of stocks S. For this set of stocks the **price time series** $\mathbf{p} : \mathbb{N} \to \mathbb{R}^S$ contains the stock prices of all stocks for each point in time, i.e., $\mathbf{p}_s(t)$ is the price of stock $s \in S$ at time $t \in \mathbb{N}$. For simplicity, we assume a stock price exists for each point in time. In practice, trading days can be mapped to this price time series by either only considering trading days as valid points in time or by keeping the stock price constant if no price update exists.

Furthermore, we define the set of **earnings announcements** $T_e \subset \mathbb{N}$ for the target stock s^*, i.e., at each point in time $t \in T_e$, new earnings have been published. For a point in time $t \in \mathbb{N}$, we can now define $t_{\text{next}} = \min\{t' \in T_e : t' > t\}$ and $t_{\text{last}} = \max\{t' \in T_e : t' < t\}$, i.e., the points in time of the next earnings announcement after and the last earnings announcement before t. The **earnings time series** $\mathbf{e} : \mathbb{N} \to \mathbb{R}$ is a piecewise constant time series with $\mathbf{e}(t) = \mathbf{e}(t_{\text{next}})$ for all $t \in \mathbb{N}$. That is, $\mathbf{e}_{s^*}(t)$ contains the potentially unknown earnings of stock $s^* \in S$ announced at time t_{next}.

In this paper, price updates correspond to daily closing prices of stocks, though the method can be straightforwardly adapted to any finer time resolution. As a measure for the earnings of a company listed on the stock market, the **earnings per share** (EPS) are used, i.e., the accumulated earnings of the company in an accounting interval divided by the number of stock shares the company emitted. The earnings per share are published quarterly in the company's income statement or its earnings announcement.

In practice, prices and earnings are only available for a finite timespan. We represent this fact in our notation as time windows. Given a time series $f : \mathbb{N} \to \mathbb{R}$ and a **time interval** $[i, j]$ with $i, j \in \mathbb{N}$ and $i < j$, the **time window** from i to j is the finite time series

$$f[i, j] \colon \{1, \ldots, j - i\} \to \mathbb{R}$$

defined by

$$f[i, j](t) = f(i + t) \ .$$

Using the definitions above, the problem tackled in this paper can be formulated as follows.

Definition 1: **Given** a point in time $t \in \mathbb{N}$, a set of stocks S, a designated target stock $s^* \in S$, the stock prices $\mathbf{p}[1, t]$ of all stocks $s \in S$ until t, as well as the earnings $\mathbf{e}[1, t_{\text{last}}]$ until the last earnings announcement, **predict** the next earnings $\mathbf{e}_{s^*}(t_{\text{next}})$ of target stock s^*.

B. Price and Correlation Feature Space

We now motivate and describe our proposed correlation feature space that is derived from publicly available stock prices. The strong relationship between stock prices and earnings is well studied [1] and used for investment; the price-earnings ratio for example is an important indicator for stock valuation. At the same time, information related to the companies earnings are priced in the stock rate as soon

as investors learn of it. Thus, we conclude that stock prices are a good proxy for current corporate earnings.

Because the earnings of a company are related to other market participants [20], e.g., component suppliers or competitors, conclusions can be drawn from the performance of those participants and vice versa. For example, prospering business for Volkswagen leads to a higher demand of parts, implying higher sales for its suppliers Schaeffler and Continental. Similarly, higher earnings for Sharp tends to imply higher earnings of Foxconn, for which Sharp is a major supplier. This again inclines to imply higher earnings of Apple, for which Foxconn is a major supplier. By using the stock prices of related companies as proxy for their prosperity, the mutual influence between companies can be incorporated in the feature space.

This steady relationship between the stock prices of a set of companies and the earnings of a dedicated target company can expressed using a linear model. Assuming a normally distributed noise on stock prices, a regularized least squares, or ridge regression is the maximum likelihood estimator and thus a suitable approach.

To increase the robustness of the estimations, we propose to measure prices according to different temporal resolutions. It is a standard in fundamental as well as in technical investment to not solely rely on the momentary picture of the last available closing price. Since stock prices are highly volatile, one also considers smoothed prices given by moving averages of different time window lengths. Formally, let $l \in \mathbb{N}$ be some interval length. Denoting by

$$\overline{f} = \sum_{t=1}^{|f|} f(t)/|f|$$

the **average** value of a finite time series f, this gives rise to the smoothed time series of **moving average prices**

$$a_s^l(t) = \overline{\mathbf{p}_s[t-l,t]} \tag{1}$$

for all stocks $s \in S$.

However, stock prices are susceptible to speculation, trends or temporary effects of high impact that are not related with the company's performance. By also considering the correlation between stock prices, such temporary effects are expressed as a change in correlation. By weighting the stock prices with the current correlation coefficient between the related and the target company's stock price, the influence of the stock price of market participants can be temporarily suspended or reversed if the correlation changes.

For example, the short-selling of Volkswagen stocks by investors in 2008 together with the attempt of Porsche to take over Volkswagen at the same time lead to a spectacular increase in Volkswagen's stock price, even though the automobile sector was performing poorly at that time. Using the price of Volkswagen shares to estimate the earnings of its component supplier Continental would fail in that moment. The 50 days correlation between the stock prices of Volkswagen and Continental decreased significantly in this period but returned to their usual value of around 0.5 shortly after.

Accordingly, we are interested in the correlation between prices of two stocks for a fixed time window. The **sample Pearson correlation coefficient** $\mathrm{cor}(f,g)$ between two finite time series f, g with $|f| = |g|$ is given by

$$\mathrm{cor}(f,g) = \mathrm{cov}(f,g)/(\mathrm{std}(f)\mathrm{std}(g)) \ ,$$

where $\mathrm{cov}(f,g)$ denotes the sample co-variance and $\mathrm{std}(f)$ the sample standard deviation. As for the stock prices, given an interval length $l \in \mathbb{N}$, we define the **moving correlation** between two stocks $s, u \in S$ as

$$c_{s,u}^l(t) = \mathrm{cor}(\mathbf{p}_s[t-l,t], \mathbf{p}_u[t-l,t]) \ . \tag{2}$$

Following our described intuition, we want to represent the earnings of a target company with stock $s \in S$ at time $t \in \mathbb{N}$ by its stock price as well as by the prices of all other stocks in the market weighted by their correlation with s. Also, we want to provide this information with respect to different time resolutions in order to capture short term and long term relations at the same time. In accordance with economic practice (see, e.g., [21]), for each moment in time t we consider a **short-term**, a **mid-term**, and a **long-term time window**, looking back 11, 50, and 200 days, respectively.

Putting everything together, we can define the **feature representation** $\varphi_{s^*} \colon \mathbb{N} \to \mathbb{R}^d$ for a stock $s^* \in S$ as

$$\varphi_{s^*}(\cdot) = \\ \circ_{u \in S} \left(p_u(\cdot), a_u^{11}(\cdot) c_{u,s}^{11}(\cdot), a_u^{50}(\cdot) c_{u,s}^{50}(\cdot), a_u^{200}(\cdot) c_{u,s}^{200}(\cdot) \right)$$

with definitions of moving average prices and moving correlations as given in equations (1) and (2). Here, the symbol $\circ$ denotes the "concatenation" of features.

Note that weighting the average prices of each stock with its correlation to the target stock is not a linear scaling but a proper non-linear augmentation of the features, because the correlation is a non-linear function in both time series.

C. Prediction Method

We now describe our proposed approach to nowcasting earnings per share using the feature representation described above. Given a target stock s^* from a set of stocks S and the definition of the feature representation φ, a point in time t naturally divides the financial data stream into a training and a prediction window. For a point in time $t \in \mathbb{N}$ and the corresponding last earnings announcement t_{last}, the **training set** E for target stock s^* is defined as

$$E = \{ (\varphi_{s^*}(t'), e_{s^*}(t')) \, | t' \leq t_{\text{last}} \} \ .$$

Moreover, we define the **prediction window** P corresponding to t as

$$P = \{\varphi_s(t') | t_{\text{last}} < t' \leq t_{\text{next}}\} \ .$$

In order to prevent susceptibility to concept drifts, it is common practice to use only the last W data points instead of the entire available data for training. That is,

$$E_W = \{(\varphi_{s^*}(t'), e_{s^*}(t')) \, | t_{\text{last}} - W < t' \leq t_{\text{last}}\} \ .$$

Using the above definitions of training set and prediction window, any regression technique can be employed to generate earnings forecasts. We propose using ridge regression [22] to construct a linear model in the correlation feature space, which is a simple and fast method that does not modify the explicitly constructed feature space. For a training set E, the ridge regression model is defined as $w^* \in \mathbb{R}^d$ solving

$$\min_{w \in \mathbb{R}^d} \sum_{(\varphi, e) \in E} \left| w^\top \varphi - e \right|^2 + \nu \|w\|_2^2$$

with some positive regularization parameter $\nu \in \mathbb{R}_+$.

In practice, predictions are made for each price update without updating the model. Only at the time of a new earnings announcement, the stored training set E_W is updated and the model can be recomputed. This scenario can be viewed as online learning with delayed update. We tackle the problem of delayed update following the straightforward approach of [23]. Examples for which the label is yet unknown are stored in a buffer. As soon as their label is revealed, i.e., the earnings are announced, the examples are presented to the learner in order of their arrival together with their now known label. Because the intervals between earnings announcement are fixed, this method only adds a constant factor to the space complexity of the algorithm.

For each point in time $t \in \mathbb{N}$, a standard online regression algorithm estimates $\mathbf{e}_{s^*}(t_{\text{next}})$, which is constant for the entire prediction window. Hence, the variance of the estimates can be reduced by averaging all estimates so far. Thereby, the proposed algorithm potentially improves its prediction quality with every element from the data stream. If the new element is an earnings update, a new training set is constructed and the model is updated. If the new element is a price update, a new prediction is generated that adds to the pool of predictions from which the new earnings nowcast is calculated as the average of all predictions in the pool.

The corresponding algorithm, called CorrelNowcast, is presented in algorithm 1. Given a target stock $s^* \in S$, a training window size W and a regularization parameter ν, the algorithm loops over all points in time t (line 3). Using the new prices $\mathbf{p}(t)$, the feature representation $\varphi_{s^*}(t)$ for target stock s^* is constructed and stored in the prediction window P (line 4). Then an earnings prediction for that point in time is calculated using the linear model w^* and stored in the predictions set Q (line 5). The final earnings nowcast is generated as the average of all earnings predictions in the current prediction set (line 6).

In case of an earnings update at time t (line 7), the training set needs to be updated. Therefore, all elements for which no corresponding earnings have been available yet, i.e., all elements in the prediction window (line 8), are assigned to the current earnings update $\mathbf{e}(t)$. Each thereby obtained tuple of feature representation and earnings value is added to the training set (line 9). After that, the prediction window and set are cleared (line 11).

If by the previous step the training set E_W has been extended beyond its window size W (line 12), the first $|E_W| - W$ elements are removed from the training set (line 13). With the updated training set E_W, a new linear model w^* is calculated (line 15).

Algorithm 1: CorrelNowcast

input : target stock $s^* \in S$, training window size W, regularization parameter ν
output: earnings forecasts

1 initialize $E_W, P, Q \leftarrow \emptyset$
2 initialize $w^* \in \mathbb{R}^{|d|} \leftarrow (0, ..., 0)$
3 **foreach** *point in time t* **do**
4 $\quad P \leftarrow P \cup \{\varphi_{s^*}(t)\}$
5 $\quad Q \leftarrow Q \cup \{\varphi_{s^*}(t)^\top w^*\}$
6 $\quad$ **predict** $avg(Q)$
7 $\quad$ **if** $t \in T_e$ **then**
8 $\quad\quad$ **foreach** $\varphi \in P$ **do**
9 $\quad\quad\quad E_W \leftarrow E_W \cup \{(\varphi, \mathbf{e}_{s^*}(t))\}$
10 $\quad\quad$ **end**
11 $\quad\quad P, Q \leftarrow \emptyset$
12 $\quad\quad$ **if** $|E_W| > W$ **then**
13 $\quad\quad\quad$ remove first $|E_W| - W$ elements from E_W
14 $\quad\quad$ **end**
15 $\quad\quad w^* \leftarrow \arg\min_{w \in \mathbb{R}^d} \sum_{(\varphi, e) \in E_W} \left| w^\top \varphi - e \right|^2 + \nu \|w\|_2^2$
16 $\quad$ **end**
17 **end**

A regularized least squares, or ridge regression can also be transformed into a full online algorithm by introducing a training example matrix A and a prediction vector b [24]. Both, matrix and vector, are incrementally updated with each training step. The weight vector, i.e., the linear model, is given implicitly by $b^\top A^{-1}$. For each step, this online ridge regression minimizes the least squared error as well as the norm of the weight vector. The online ridge regression approach can be adapted to a delayed reward scenario similar to the proposed algorithm. Because of several matrix-vector multiplications for each example, this method is significantly slower than a batched ridge regression. In the case of

earnings predictions, where long delays between new labels make learning only necessary after a considerable amount of examples already arrived, the faster CorrelNowcasting algorithm with a batched learning phase is preferable.

III. Experiments

In this section, we present two experiments on publicly available daily stock price and quarterly earnings data[1]. In total, the experiments involve predicting 5200 earnings updates of 200 US stocks. In the first experiment, we show that the proposed method can outperform human analysts and in the second experiment we show that correlation features are significant to the proposed method. For replicability of results, data will be made available on our website[2].

A. Comparison with Human Analysts

We compare CorrelNowcast with human analysts on the Standard & Poor's 100 index in the time from 2008 to 2012. Analyst consensus forecasts aggregated by Zacks Investment Research for this period have been obtained from bloomberg.com. As additional baselines we employ the classic econometric **ARMA** method (details can be found in the appendix A) and a trivial **constant** method for assessing the difficulty of the problem. This method simply uses the last available earnings as prediction value. Optimal parameter settings for each method are obtained on an independent dataset of 100 randomly selected stocks in the time from 2004 to 2006. For CorrelNowcast, the regularization parameter ν and the prediction window W have been found by a grid search with $\nu \in \{10, 100, 1000, 3000, 5000, 10000\}$ and $W \in \{11, 50, 125, 200, 250, 350, 500, 750\}$.

The results of the experiment are listed in table I. Besides the rooted mean squared error (RMSE), we also provide the mean relative error (MRE), i.e., the absolute error relative to the true value, which is more relevant for investment. The analyst's forecasts are published once per quarter on an arbitrary day before the next earnings announcement. All reported errors are average values over a) all target stocks from S&P 100 and over b) all prediction days starting from the day of analyst forecast until the earnings are published, i.e., the true label is revealed. Additionally, since CorrelNowcast provides increasingly refined predictions with each day (and is also defined prior to analyst forecasts), for this method also results for specific days are given: the first and the last day of a quarter as well as the day on which analyst's forecasts are published.

We can observe that CorrelNowcast outperforms the analyst's forecasts in terms of RMSE and MRE on the whole quarter, as well as on the day of analyst's forecasts and the last day of the quarter. In addition, CorrelNowcast is only 3% worse in terms of RMSE and 35% worse in terms of MRE, when earnings are predicted at the first day of the quarter—on avg. 9.4 days before analyst forecasts are even available. This advantage does not only hold on the average over all stocks but CorrelNowcast also outperforms human analysts on the majority of 62% of stocks in both error measures.

Confirming previous research, the ARMA model is outperformed by both methods by one order of magnitude when measured by the relevant MRE. The fact that ARMA performs better than the analyst's forecasts in terms of RMSE can be attributed to the recession in the United States from 2008 to 2009, which lead to low corporate earnings until 2012 (and corresponding low absolute errors). This interpretation is also confirmed by the constant baseline.

In summary, this experiment verifies that the proposed method is capable of nowcasting corporate earnings from publicly available data that outperform human analysts. It furthermore indicates that the proposed method is able to generate accurate nowcasts even significantly before analyst's forecasts are published.

B. Significance of Correlation Features

It remains to verify the significance of the design choices involved in the definition of CorrelNowcast—in particular the use of the price correlation features. For this purpose we compare the method against two nowcasting approaches that use a reduced feature space containing only prices and average prices:

$$\varphi'_{s^*}(\cdot) = \circ_{u \in S} \left(p_u(\cdot), a_u^{11}(\cdot), a_u^{50}(\cdot), a_u^{200}(\cdot) \right) \ .$$

The two methods are for once the same linear ridge regression as for CorrelNowcast (**LinPriceNowcast**) and moreover a more expressive kernelized ridge regression (**KernelPriceNowcast**) utilizing a polynomial kernel

$$k(x, y) = (\gamma x^\top y + d)^d \ .$$

Again all parameters are optimized on the same tuning set as in the first experiment. For testing we now use an extended setup with two datasets: the S&P 100 from 2004-2012 and an independent set of 100 random US stocks from 2007-2012 (**RAND 100**).

The results are listed in table II. For additional comparison we again also provide results for ARMA and constant. For both error measures, CorrelNowcast substantially outperforms the baselines using only simple price features. In particular on RAND 100 the performance of these methods are closer to the ARMA baseline, which again is weaker than CorrelNowcast by an order of magnitude.

The results show that the correlation features clearly outperform simple price features. Even the more expressive kernel variant cannot lift the simple price features into competitive range. Note that the MRE is significantly smaller on S&P 100 than RAND 100, because the companies listed in S&P 100 are the 100 largest American companies with rather high earnings values, whereas in RAND 100 many

[1]Prices are gathered from Google Finance (www.google.com/finance); earnings from YCharts (www.ycharts.com).

[2]URL is omitted for double blind review process.

	RMSE	MRE
Analyst	0.828	0.417
CorrelNowcast	**0.689**	**0.406**
last day of quarter	0.678	0.319
day of analyst's forecast	0.693	0.385
first day of quarter	0.854	0.563
ARMA	0.776	16.797
Constant	0.888	27.489

Table I
COMPARISON OF THE PROPOSED METHOD WITH ANALYST'S FORECASTS ON THE SP100 STOCKS FROM 2008 TO 2012. RESULTS ARE AVERAGED OVER ALL STOCKS AND ALL RELEVANT PREDICTION DAYS (FOR CORRELNOWCAST ALSO SELECTED INDIVIDUAL DAYS ARE PROVIDED).

	RAND 100 (2007-2012)		S&P 100 (2004-2012)	
	RMSE	MRE	RMSE	MRE
CorrelNowcast	**0.530**	**34.962**	**0.700**	**3.044**
KernelPriceNowcast	0.670	141.224	0.986	4.891
LinPriceNowcast	0.688	148.766	0.996	5.579
ARMA	0.660	220.729	1.099	16.799
Constant	0.796	664.139	0.928	26.884

Table II
AVERAGED RMSE AND MRE ON ALL S&P 100 TARGET STOCKS BETWEEN 2004 AND 2012 AS WELL AS AN ADDITIONAL INDEPENDENT TEST DATASET (RAND 100) OF 100 RANDOM US STOCKS BETWEEN 2007 AND 2012.

stocks with very little earnings are listed so that even a moderate absolute error results in a large relative error.

Altogether this experiment not only confirms the necessity of using price correlation features, but also provides a much broader performance assessment of CorrelNowcast. Besides using more stocks, the included time periods show more variety of the underlying economic environment. It does not only contain data from a stable growth period (2004 to 2006), but also the financial crisis of 2007, the recession from 2008 to 2009 and the recovery period until 2012. Thus, this experiment shows that CorrelNowcast can maintain its good performance on independent data sets and longer time spans.

IV. CONCLUSION

We presented a fully automatized method for nowcasting corporate earnings using a novel correlation feature space derived from publicly available price data. The proposed method is simple, fast and can be applied to any set of stocks, their prices and earnings. Experiments have shown that these nowcasts outperform analyst's forecasts and are even competitive when generated significantly before the an-

alyst's forecasts are published. Besides the implications for the proposed method, these results emphasize the importance and potential capabilities of purely data driven methods for financial data.

The given method implicitly selects relevant stocks from the input set, hence a larger set of stocks potentially provides more relevant information without negative side-effects. Thus, to further improve performance it is desirable to eventually use all stocks traded worldwide as input for our method. With such a large number of stocks, and an even larger number of features, the scalability of the method is placed in the focus of future research. To achieve a high scalability, it appears to be desirable to switch from a centralized prediction approach to distributed online prediction. A straight-forward distribution of the method leads to an enormous communication overhead between the nodes in the distributed system. Therefore, strategies for communication reduction have to be employed. Strategies like the safe-zone approach [25], where communication is omitted as long as a globally defined criterion is fulfilled at each local node, or sketches [26], where data is compressed, appear to be promising

A different interesting direction for follow up research is improving the feature space by including features derived from different data sources, most prominent from financial news [27]. However, with more features, the method becomes computationally more complex, again directing the focus on scalability.

REFERENCES

[1] J. M. Patell, "Corporate forecasts of earnings per share and stock price behavior: Empirical test," *Journal of Accounting Research*, pp. 246–276, 1976.

[2] S. Makridakis, "Forecasting: its role and value for planning and strategy," *International Journal of Forecasting*, vol. 12, no. 4, pp. 513–537, 1996.

[3] J. Affleck-Graves, L. R. Davis, and R. R. Mendenhall, "Forecasts of earnings per share: Possible sources of analyst superiority and bias," *Contemporary Accounting Research*, vol. 6, no. 2, pp. 501–517, 1990.

[4] R. Conroy and R. Harris, "Consensus forecasts of corporate earnings: Analysts' forecasts and time series methods," *Management Science*, vol. 33, no. 6, pp. 725–738, 1987.

[5] W. Zhang, Q. Cao, and M. J. Schniederjans, "Neural network earnings per share forecasting models: a comparative analysis of alternative methods," *Decision Sciences*, vol. 35, no. 2, pp. 205–237, 2004.

[6] Q. Cao and M. E. Parry, "Neural network earnings per share forecasting models: A comparison of backward propagation and the genetic algorithm," *Decision Support Systems*, vol. 47, no. 1, pp. 32–41, 2009.

[7] S. Mahfoud and G. Mani, "Financial forecasting using genetic algorithms," *Applied Artificial Intelligence*, vol. 10, no. 6, pp. 543–566, 1996.

[8] V. Dhar and D. Chou, "A comparison of nonlinear methods for predicting earnings surprises and returns," *Neural Networks, IEEE Transactions on*, vol. 12, no. 4, pp. 907–921, 2001.

[9] M.-A. Mittermayer, "Forecasting intraday stock price trends with text mining techniques," in *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*. IEEE, 2004, pp. 10–pp.

[10] K.-j. Kim, "Financial time series forecasting using support vector machines," *Neurocomputing*, vol. 55, no. 1, pp. 307–319, 2003.

[11] C. Tsai and S. Wang, "Stock price forecasting by hybrid machine learning techniques," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. 1, no. 755, 2009, p. 60.

[12] J. Abarbanell, "Do analysts' earnings forecasts incorporate information in prior stock price changes?" *Journal of Accounting and Economics*, vol. 14, no. 2, pp. 147–165, 1991.

[13] R. G. Sloan, "Do stock prices fully reflect information in accruals and cash flows about future earnings?" *Accounting Review*, pp. 289–315, 1996.

[14] R. King, V. Smith, A. Williams, and M. Van Boening, "The Robustness of Bubbles and Crashes in Experimental Stock Markets," *Nonlinear Dynamics and Evolutionary Economics*, pp. 183–200, 1993.

[15] H. Choi and H. Varian, "Predicting the present with google trends," *Economic Record*, vol. 88, no. s1, pp. 2–9, 2012.

[16] J. Bollen, H. Mao, and X. Zeng, "Twitter mood predicts the stock market," *Journal of Computational Science*, vol. 2, no. 1, pp. 1–8, 2011.

[17] K. A. Aastveit and T. Trovik, "Nowcasting norwegian gdp: The role of asset prices in a small open economy," *Empirical Economics*, vol. 42, no. 1, pp. 95–119, 2012.

[18] M. Ettredge, J. Gerdes, and G. Karuga, "Using web-based search data to predict macroeconomic statistics," *Communications of the ACM*, vol. 48, no. 11, pp. 87–92, 2005.

[19] S. Goel, J. M. Hofman, S. Lahaie, D. M. Pennock, and D. J. Watts, "Predicting consumer behavior with web search," *Proceedings of the National Academy of Sciences*, vol. 107, no. 41, pp. 17 486–17 490, 2010.

[20] A. W. Alford, "The effect of the set of comparable firms on the accuracy of the price-earnings valuation," *Journal of Accounting Research*, vol. 30, no. 1, pp. 94–108.

[21] Credit Suisse, *Technical Analysis - Explained*, Credit Suisse Group AG.

[22] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[23] C. Mesterharm, "On-line learning with delayed label feedback," in *Algorithmic Learning Theory*. Springer, 2005, pp. 399–413.
[24] F. Zhdanov and V. Vovk, "Competing with gaussian linear experts," *Transactions of the IRE Professional Group on Audio*, vol. 30, no. 6, 2009.
[25] I. Sharfman, A. Schuster, and D. Keren, "A geometric approach to monitoring threshold functions over distributed data streams," *ACM Transactions on Database Systems (TODS)*, vol. 32, no. 4, p. 23, 2007.
[26] G. Cormode and M. Garofalakis, "Sketching streams through the net: Distributed approximate query tracking," in *Proceedings of the 31st international conference on Very large data bases*. VLDB Endowment, 2005, pp. 13–24.
[27] R. P. Schumaker and H. Chen, "Textual analysis of stock market prediction using breaking financial news: The azfin text system," *ACM Transactions on Information Systems (TOIS)*, vol. 27, no. 2, p. 12, 2009.
[28] L. D. Brown and M. S. Rozeff, "Univariate time-series models of quarterly accounting earnings per share: A proposed model," *Journal of Accounting Research*, pp. 179–189, 1979.

APPENDIX A.
ARMA

In our experiments we compare CorrelNowcast to a Box-Jenkins method, i.e., an autoregressive moving average (ARMA) model. The ARMA(p,q) model consists of the sum of an autoregressive component of order p and a moving average component of order q. For a time series $x\colon \mathbb{N} \to \mathbb{R}$, the model is defined as

$$x(t) = c + \epsilon_t + \sum_{i=1}^{p} v_i x(t-i) + \sum_{i=1}^{q} w_i \epsilon_{t-i} \ ,$$

with weights $v \in \mathbb{R}^p$ and $w \in \mathbb{R}^q$, an offset constant c and a Gaussian noise terms $\epsilon_t \sim \mathcal{N}(0, \sigma)$.

In our experiment, we apply this approach to the earnings time series [28], updating the model whenever a new earnings value is published. For each model update, a grid search over the parameters p and q is performed to find the best fitting ARMA-model. For the grid search, we used $p, q \in \{1, ..., 5\}$. For each setting of p and q, the parameters v, w, c are fitted to the training time series (containing all known values) of earnings per share values using a least squares regression. For each training set, the best performing model is chosen for prediction.