Deliverable D1.2

Learning graph based quantum algorithm for a problem with small 1-certificate complexity

Due date: Month 12 (April 30, 2014)

Time-Efficient Quantum Walks for 3-Distinctness*

Aleksandrs Belovs[1], Andrew M. Childs[2,4], Stacey Jeffery[3,4], Robin Kothari[3,4], and Frédéric Magniez[5]

[1] Faculty of Computing, University of Latvia
[2] Department of Combinatorics & Optimization, University of Waterloo, Canada
[3] David R. Cheriton School of Computer Science, University of Waterloo, Canada
[4] Institute for Quantum Computing, University of Waterloo, Canada
[5] CNRS, LIAFA, Univ Paris Diderot, Sorbonne Paris-Cité, France

Abstract. We present two quantum walk algorithms for 3-Distinctness. Both algorithms have time complexity $\tilde{O}(n^{5/7})$, improving the previous $\tilde{O}(n^{3/4})$ and matching the best known upper bound for query complexity (obtained via learning graphs) up to log factors. The first algorithm is based on a connection between quantum walks and electric networks. The second algorithm uses an extension of the quantum walk search framework that facilitates quantum walks with nested updates.

1 Introduction

Element Distinctness is a basic computational problem. Given a sequence $\chi = \chi_1, \ldots, \chi_n$ of n integers, the task is to decide if those elements are pairwise distinct. This problem is closely related to Collision, a fundamental problem in cryptanalysis. Given a 2-to-1 function $f : [n] \to [n]$, the aim is to find $a \neq b$ such that $f(a) = f(b)$. One of the best (classical and quantum) algorithms is to run Element Distinctness on f restricted to a random subset of size $\sqrt{n}$.

In the quantum setting, Element Distinctness has received a lot of attention. The first non-trivial algorithm used $\tilde{O}(n^{3/4})$ time [1]. The optimal $\tilde{O}(n^{2/3})$ algorithm is due to Ambainis [2], who introduced an approach based on quantum walks that has become a major tool for quantum query algorithms. The optimality of this algorithm follows from a query lower bound for Collision [3]. In the query model, access to the input χ is provided by an oracle whose answer to query $i \in [n]$ is χ_i. This model is the quantum analog of classical decision tree complexity: the only resource measured is the number of queries to the input.

Quantum query complexity has been a very successful model for studying the power of quantum computation. In particular, bounded-error quantum query

* This paper is a merge of two submitted papers, whose full versions are available at `http://arxiv.org/abs/1302.3143` and `http://arxiv.org/abs/1302.7316`. Support for this work was provided by European Social Fund within the project "Support for Doctoral Studies at University of Latvia", the European Commission IST STREP project 25596 (QCS), NSERC, the Ontario Ministry of Research and Innovation, the US ARO, and the French ANR Blanc project ANR-12-BS02-005 (RDAM).

F.V. Fomin et al. (Eds.): ICALP 2013, Part I, LNCS 7965, pp. 105–122, 2013.

complexity has been exactly characterized in terms of a semidefinite program, the general adversary bound [4, 5]. To design quantum query algorithms, it suffices to exhibit a solution to this semidefinite program. However, this turns out to be difficult in general, as the minimization form of the general adversary bound has exponentially many constraints. Belovs [6] recently introduced the model of learning graphs, which can be viewed as the minimization form of the general adversary bound with additional structure imposed on the form of the solution. This additional structure makes learning graphs much easier to reason about. The learning graph model has already been used to improve the query complexity of many graph problems [6–9] as well as k-Distinctness [10].

One shortcoming of learning graphs is that these upper bounds do not lead explicitly to efficient algorithms in terms of time complexity. Although the study of query complexity is interesting on its own, it is relevant in practice only when a query lower bound is close to the best known time complexity.

Recently, [11] reproduced several known learning graph upper bounds via explicit algorithms in an extension of the quantum walk search framework of [12]. This work produced a new quantum algorithmic tool, quantum walks with nested checking. Algorithms constructed in the framework of [11] can be interpreted as quantum analogs of randomized algorithms, so they are simple to design and analyze for any notion of cost, including time as well as query complexity. This framework has interpreted all known learning graphs as quantum walks, except the very recent *adaptive learning graphs* for k-Distinctness [10].

In k-Distinctness, the problem is to decide if there are k copies of the same element in the input, with $k = 2$ being Element Distinctness. The best lower bound for k-Distinctness is the Element Distinctness lower bound $\Omega(n^{2/3})$, whereas the best query upper bound is $O(n^{1-2^{k-2}/(2^k-1)}) = o(n^{3/4})$ [10], achieved using learning graphs, improving the previous bound of $O(n^{k/(k+1)})$ [2]. However, the best known time complexity remained $\tilde{O}(n^{k/(k+1)})$. We improve this upper bound for the case with $k = 3$ using two distinct approaches.

For the first approach, described in Sections 2 and 3, we use a connection between quantum walks and electric networks. Hitting and commute times of random walks are closely connected to the effective resistance of associated networks of resistors. We develop a similar connection for the case of quantum walks. For any initial distribution over the vertices of a graph, we prove that a quantum walk can detect the presence of a marked element in $O(\sqrt{WR})$ steps, where W is the total weight of the graph and R is the effective resistance. This generalizes a result of Szegedy that only applies if the initial distribution is stationary.

The second approach, described in Sections 4 and 5, uses quantum walks with nested updates. The basic idea is conceptually simple: we walk on sets of 2-collisions and look for a set containing a 2-collision that is part of a 3-collision. We check if a set has this property by searching for an index that evaluates to the same value as one of the 2-collisions in the set. However, to move to a new set of 2-collisions, we need to use a quantum walk subroutine for finding 2-collisions as part of our update step. This simple idea is surprisingly difficult to implement and leads us to develop a new extension of the quantum walk search framework.

2 Quantum Walks and Electric Networks

2.1 Random Walks and Electric Networks

Let $G = (V, E)$ be a simple undirected graph with each edge assigned a *weight* $w_e \geq 0$. Let $W = \sum_{e \in E} w_e$ be the *total weight*. Consider the following *random walk* on G: If the walk is at a vertex $u \in V$, proceed to a vertex v with probability proportional to w_{uv}, i.e., $w_{uv}/(\sum_{ux \in E} w_{ux})$. The random walk has a *stationary probability distribution* $\pi = (\pi_u)$ given by $\pi_u = \sum_{uv \in E} w_{uv}/(2W)$.

Let $\sigma = (\sigma_u)$ be an *initial probability distribution* on the vertices of the graph, and let $M \subseteq V$ be some set of *marked vertices*. We are interested in the *hitting time* $H_{\sigma,M}$ of the random walk: the expected number of steps of the random walk required to reach a vertex in M when the initial vertex is sampled from σ. If σ is concentrated in a vertex $s \in V$, or M consists of a single element $t \in V$, we replace σ by s or M by t. The *commute time* between vertices s and t is defined as $H_{s,t} + H_{t,s}$. We assume G and σ are known, and the task is to determine whether M is non-empty.

Assume M is non-empty, and define a *flow* on G from σ to M as a real-valued function p_e on the (oriented) edges of the graph satisfying the following conditions. First, $p_{uv} = -p_{vu}$. Next, for each non-marked vertex u,

$$\sigma_u = \sum_{uv \in E} p_{uv}. \tag{1}$$

That is, σ_u units of the flow are injected into u, traverse through the graph, and are removed from marked vertices. Define the *energy* of the flow as

$$\sum_{e \in E} \frac{p_e^2}{w_e}. \tag{2}$$

Clearly, the value of (2) does not depend on the orientation of each e. The *effective resistance* $R_{\sigma,M}$ is the minimal energy of a flow from σ to M. For R, as for H, we also replace σ and M by the corresponding singletons. The resistance $R_{\sigma,M}$ equals the energy dissipated by the electric flow where the edges have conductance w_e and σ_u units of the current are injected into each u and collected in M [13]. The following two results can be easily obtained from the results in Ref. [14]:

Theorem 1. *If G, w, W are as above, s, t are two vertices of G, $M \subseteq V$, and π is the stationary distribution on G, then*

(a) the commute time between s and t equals $2WR_{s,t}$ and
(b) the hitting time $H_{\pi,M}$ equals $2WR_{\pi,M}$.

We show a quadratic improvement in the quantum case: If G and σ are known in advance and the superposition $\sum_{u \in V} \sqrt{\sigma_u}|u\rangle$ is given, the presence of a marked vertex in G can be determined in $O(\sqrt{WR})$ steps of a quantum walk. By combining this result with the second statement of Theorem 1, we obtain the main result of the paper by Szegedy [15].

2.2 Tools from Quantum Computing

Although we use the language of electric networks to state our results, we use spectral properties of unitary operators in the algorithms.

Lemma 1 (Effective Spectral Gap Lemma [5]). *Let Π_A and Π_B be two orthogonal projectors in the same vector space, and $R_A = 2\Pi_A - I$ and $R_B = 2\Pi_B - I$ be the reflections about their images. Assume P_Θ, where $\Theta \geq 0$, is the orthogonal projector on the span of the eigenvectors of $R_B R_A$ with eigenvalues $\mathsf{e}^{\mathsf{i}\theta}$ such that $|\theta| \leq \Theta$. Then, for any vector w in the kernel of Π_A, we have*

$$\|P_\Theta \Pi_B w\| \leq \frac{\Theta}{2}\|w\|.$$

Theorem 2 (Phase Estimation [16], [17]). *Assume a unitary U is given as a black box. There exists a quantum algorithm that, given an eigenvector of U with eigenvalue $\mathsf{e}^{\mathsf{i}\phi}$, outputs a real number w such that $|w-\phi| \leq \delta$ with probability at least $9/10$. Moreover, the algorithm uses $O(1/\delta)$ controlled applications of U and $\frac{1}{\delta}\operatorname{polylog}(1/\delta)$ other elementary operations.*

2.3 Szegedy-Type Quantum Walk

In this section, we construct a quantum counterpart of the random walk described in Section 2.1. The quantum walk differs slightly from the quantum walk of Szegedy. The framework of the algorithm goes back to [18], and Lemma 1 is used to analyze its complexity. We assume the notations of Section 2.1 throughout the section.

It is customary to consider quantum walks on bipartite graphs. Let $G = (V, E)$ be a bipartite graph with parts A and B. Also, we assume the support of σ is contained in A. These are not very restrictive assumptions: If either of them fails, consider the bipartite graph G' with the vertex set $V' = V \times \{0, 1\}$, the edge set $E' = \{(u,0)(v,1), (u,1)(v,0) : uv \in E\}$, edge weights $w'_{(u,0)(v,1)} = w'_{(u,1)(v,0)} = w_{uv}$, the initial distribution $\sigma'_{(u,0)} = \sigma_u$, and the set of marked vertices $M' = M \times \{0, 1\}$.

We assume the quantum walk starts in the state $|\varsigma\rangle = \sum_{u\in V} \sqrt{\sigma_u}\,|u\rangle$ that is known in advance. Also, we assume there is an upper bound R known on the effective resistance from σ to M for all potential sets M of marked vertices.

Now we define the vector space of the quantum walk. Let S be the *support* of σ, i.e., the set of vertices u such that $\sigma_u \neq 0$. The vectors $\{|u\rangle : u \in S\} \cup \{|e\rangle : e \in E\}$ form the computational basis of the vector space of the quantum walk. Let $\mathcal{H}_u$ denote the *local space* of u, i.e., the space spanned by all $|uv\rangle$ for $uv \in E$ and, additionally, $|u\rangle$ if $u \in S$. We have that $\bigoplus_{u\in A} \mathcal{H}_u$ equals the whole space of the quantum walk, and $\bigoplus_{u\in B} \mathcal{H}_u$ equals the subspace spanned by $|e\rangle$ for $e \in E$.

The *step of the quantum walk* is defined as $R_B R_A$ where $R_A = \bigoplus_{u\in A} D_u$ and $R_B = \bigoplus_{u\in B} D_u$ are the direct sums of the *diffusion* operations. Each D_u is a reflection operation in $\mathcal{H}_u$. All D_u in R_A or R_B commute, which makes them easy to implement in parallel. They are as follows:

- If a vertex u is marked, then D_u is the identity, i.e., the reflection about $\mathcal{H}_u$;
- If u is not marked, then D_u is the reflection about the orthogonal complement of $|\psi_u\rangle$ in $\mathcal{H}_u$, where

$$|\psi_u\rangle = \sqrt{\frac{\sigma_u}{C_1 R}}\,|u\rangle + \sum_{uv\in E} \sqrt{w_{uv}}\,|uv\rangle \tag{3}$$

for some large constant $C_1 > 0$ we choose later. The vector $|\psi_u\rangle$ is not necessarily normalized. This also holds for $u \notin S$: then the first term in (3) disappears.

Theorem 3. *The presence of a marked vertex can be detected with bounded error in $O(\sqrt{RW})$ steps of the quantum walk.*

Proof. Similarly to the Szegedy algorithm, we may assume S is disjoint from M. We perform phase estimation on $R_B R_A$ starting in $|\varsigma\rangle$ with precision $1/(C\sqrt{RW})$ for some constant C. We accept iff the phase is 1. The complexity estimate follows from Theorem 2. Let us prove the correctness. We start with the positive case. Let p_e be a flow from σ to M with energy at most R. First, using the Cauchy-Schwarz inequality and the fact that S is disjoint from M, we get

$$RW \geq \left(\sum_{e\in E} \frac{p_e^2}{w_e}\right)\left(\sum_{e\in E} w_e\right) \geq \sum_{e\in E} |p_e| \geq 1. \tag{4}$$

Now, we construct an eigenvalue-1 eigenvector

$$|\phi\rangle = \sqrt{C_1 R}\sum_{u\in S} \sqrt{\sigma_u}|u\rangle - \sum_{e\in E} \frac{p_e}{\sqrt{w_e}}|e\rangle$$

of $R_B R_A$ having large overlap with $|\varsigma\rangle$ (assume the orientation of each edge e is from A to B.) Indeed, by (1), $|\phi\rangle$ is orthogonal to all $|\psi_u\rangle$, so it is invariant under the action of both R_A and R_B. Moreover, $\||\phi\rangle\|^2 = C_1 R + \sum_{e\in E} p_e^2/w_e$ and $\langle\phi|\varsigma\rangle = \sqrt{C_1 R}$. Since we assumed $R \geq \sum_{e\in E} p_e^2/w_e$, we get that the normalized vector satisfies

$$\frac{\langle\phi|\varsigma\rangle}{\||\phi\rangle\|} \geq \sqrt{\frac{C_1}{1+C_1}}. \tag{5}$$

Now consider the negative case. Define

$$|w\rangle = \sqrt{C_1 R}\left(\sum_{u\in S} \sqrt{\frac{\sigma_u}{C_1 R}}\,|u\rangle + \sum_{e\in E} \sqrt{w_e}\,|e\rangle\right).$$

Let Π_A and Π_B be the projectors on the invariant subspaces of R_A and R_B, respectively. Since $S \subseteq A$, we have $\Pi_A|w\rangle = 0$ and $\Pi_B|w\rangle = |\varsigma\rangle$. Also

$$\||w\rangle\|^2 = \sum_{u\in S} \sigma_u + C_1 R \sum_{e\in E} w_e = 1 + C_1 RW,$$

hence, by Lemma 1, we have that, if

$$\Theta = \frac{1}{C_2\sqrt{1 + C_1 RW}}$$

for some constant $C_2 > 0$, then the overlap of $|\varsigma\rangle$ with the eigenvectors of $R_B R_A$ with phase less than Θ is at most $1/(2C_2)$. Comparing this with (5), we find that it is sufficient to execute phase estimation with precision Θ if C_1 and C_2 are large enough. Also, assuming $C_1 \geq 1$, we get $\Theta = \Omega(1/\sqrt{RW})$ by (4). □

3 Application to 3-Distinctness

In this section, we describe a quantum algorithm for 3-distinctness having time complexity $\tilde{O}(n^{5/7})$. This is a different algorithm from Ref. [10], and is based on ideas from Ref. [19].

Theorem 4. *The 3-distinctness problem can be solved by a quantum algorithm in time $\tilde{O}(n^{5/7})$ using quantum random access quantum memory (QRAQM) of size $\tilde{O}(n^{5/7})$.*

Recall that Ambainis's algorithm consists of two phases: the setup phase that prepares the uniform superposition, and the quantum walk itself. Our algorithm also consists of these two phases. In our case, the analysis of the quantum walk is quite simple, and can be easily generalized to any k. However, the setup phase is hard to generalize. The case of $k = 3$ has a relatively simple *ad hoc* solution (see the full version of the paper).

Technicalities. We start the section with some notations and algorithmic primitives we need for our algorithm. For more detail on the implementation of these primitives, refer to the paper by Ambainis [2]. Although this paper does not exactly give the primitives we need, it is straightforward to apply the necessary modifications; we omit the details here.

We are given a string $\chi \in [q]^n$. A subset $J \subseteq [n]$ of size ℓ is called an *ℓ-collision* iff $\chi_i = \chi_j$ for all $i, j \in J$. In the k-distinctness problem, the task is to determine whether the given input contains a k-collision. Inputs with a k-collision are called *positive*; the remaining inputs are called *negative*.

Without loss of generality, we may assume that any positive input contains exactly one k-collision [2]. Also, we may assume there are $\Omega(n)$ $(k-1)$-collisions by extending the input with dummy elements.

For a subset $S \subseteq [n]$ and $i \in [k]$, let S_i denote the set of $j \in S$ such that $|\{j' \in S : \chi_{j'} = \chi_j\}| = i$. Denote $r_i = |S_i|/i$, and call $\tau = (r_1, \ldots, r_k)$ the *type* of S.

Our main technical tool is a dynamical quantum data structure that maintains a subset $S \subseteq [n]$ and the values χ_j for $j \in S$. We use notation $|S\rangle_{\mathsf{D}}$ to denote a register containing the data structure for a particular choice of $S \subseteq [n]$.

The data structure can perform several operations in polylogarithmic time. The initial state of the data structure is $|\emptyset\rangle_{\mathsf{D}}$. The update operation adds or

removes an element: $|S\rangle_{\mathsf{D}}|j\rangle|\chi_j\rangle \mapsto |S\triangle\{j\}\rangle_{\mathsf{D}}|j\rangle|0\rangle$, where that $\triangle$ denotes for the symmetric difference. The data structure can perform several query operations. It can give the type τ of S. For integers $i \in [k]$ and $\ell \in [|S_i|]$, it returns the ℓth element of S_i according to some internal ordering. Given an element $j \in [n]$, it detects whether j is in S, and if it is, returns the pair (i, ℓ) such that j is the ℓth element of S_i. Given $a \in [q]$, it returns $i \in [k]$ such that a is a value in S_i or says there is no such i.

The data structure is coherence-friendly, i.e., a subset S has the same representation $|S\rangle_{\mathsf{D}}$ independent of the sequence of update operations that results in this subset. It has an exponentially small error probability of failing that can be ignored. The data structure can be implemented using quantum random access quantum memory (QRAQM) in the terminology of Ref. [20].

The quantum walk part of the algorithm is given in the following theorem. For the $k = 3$, there exists a setup procedure that prepares the state $|\varsigma\rangle$ defined in the statement of the theorem with $r_1 = r_2 = n^{4/7}$ in time $\tilde{O}(n^{5/7})$. Together with the theorem this gives an $\tilde{O}(n^{5/7})$-time quantum algorithm for 3-distinctness.

Theorem 5. *Let $r_1, \ldots, r_{k-1} = o(n)$ be positive integers, let $\chi \in [q]^n$ be an input for the k-distinctness problem, and let V_0 be the set of all $S \subseteq [n]$ having type $(r_1, \ldots, r_{k-1}, 0)$. Given the uniform superposition $|\varsigma\rangle = \frac{1}{\sqrt{|V_0|}} \sum_{S \in V_0} |S\rangle$, the k-distinctness problem can be solved in $\tilde{O}(n/\sqrt{\min\{r_1, \ldots, r_{k-1}\}})$ quantum time.*

Proof. We may assume that any input contains at most one k-collision and $\Omega(n)$ $(k-1)$-collisions. Define $r_k = 0$, and the type τ_i as $(r_1, \ldots, r_{i-1}, r_i + 1, r_{i+1}, \ldots, r_k)$ for $i \in \{0, 1, \ldots, k\}$. Let V_i be the set of all $S \subseteq [n]$ having type τ_i (consistent with our previous notation for V_0). Denote $V = \bigcup_i V_i$. Also, for $i \in [k]$, define the set Z_i of *dead-ends* consisting of vertices of the form (S, j) for $S \in V_{i-1}$ and $j \in [n]$ such that $S \triangle \{j\} \notin V$. Again, $Z = \bigcup_i Z_i$.

The vertex set of G is $V \cup Z$. Each $S \in V \setminus V_k$ is connected to n vertices, one for each $j \in [n]$: if $S \triangle \{j\} \in V$, the vertex corresponding to j is $S \triangle \{j\}$; otherwise, it is $(S, j) \in Z$. A vertex $S \in V_k$ is connected to k vertices in V_{k-1} differing from S in one element. Each $(S, j) \in Z$ is only connected to S. The weight of each edge is 1. A vertex is marked if and only if it is contained in V_k.

The algorithm of Theorem 3 is not applicable here because we do not know the graph in advance (it depends on the input), nor do we know the amplitudes in the initial state $|\varsigma\rangle$. However, we know the graph locally, and our ignorance in the amplitudes of $|\varsigma\rangle$ conveniently cancels with our ignorance in the size of G.

Let us briefly describe the implementation of the quantum walk on G following Section 2.3. Let $G = (V \cup Z, E)$ be the graph described above. It is bipartite: the part A contains all V_i and Z_i for i even, and B contains all V_i and Z_i for i odd. The support of $|\varsigma\rangle$ is contained in A. The reflections R_A and R_B are the direct sums of local reflections D_u over all u in A and B, respectively. They are as follows:

– If $u \in V_k$, then D_u is the identity in $\mathcal{H}_u$.

– If $u \in Z_i$, then D_u negates the amplitude of the only edge incident to u.
– If $u \in V_i$ for $i < k$, then D_u is the reflection about the orthogonal complement of $|\psi_u\rangle$ in $\mathcal{H}_u$. If $u \in V_0$, or $u \in V_i$ with $i > 0$, then $|\psi_u\rangle$ is defined as

$$|\psi_u\rangle = \frac{1}{\sqrt{C_1}}|u\rangle + \sum_{uv \in E} |uv\rangle, \qquad \text{or} \qquad |\psi_u\rangle = \sum_{uv \in E} |uv\rangle,$$

respectively. Here, C_1 is a constant.

The space of the algorithm consists of three registers: D, C and Z. The data register D contains the data structure for $S \subseteq [n]$. The coin register C contains an integer in $\{0, 1, \ldots, n\}$, and the qubit Z indicates whether the vertex is an element of Z. A combination $|S\rangle_{\mathsf{D}}|0\rangle_{\mathsf{C}}|0\rangle_{\mathsf{Z}}$ with $S \in V_0$ indicates a vertex in V_0 that is used in $|\varsigma\rangle$. A combination $|S\rangle_{\mathsf{D}}|j\rangle_{\mathsf{C}}|0\rangle_{\mathsf{Z}}$ with $j > 0$ indicates the edge between S and $S \triangle \{j\}$ or $(S, j) \in Z$. Finally, a combination $|S\rangle_{\mathsf{D}}|j\rangle_{\mathsf{C}}|1\rangle_{\mathsf{Z}}$ indicates the edge between $(S, j) \in Z$ and $S \in V$.

The reflections R_A and R_B are broken down into the *diffuse* and *update* operations. The diffuse operations perform the local reflections in the list above. For the first one, do nothing conditioned on $|S\rangle_{\mathsf{D}}$ being marked. For the second one, negate the phase conditioned on Z containing 1. The third reflection is the Grover diffusion [21] with one special element if $S \in V_0$.

The representation of the edges is asymmetric. One vertex is contained in the D register, and the other is stored jointly by the D and C registers. The update operation changes the representation of the edge to the opposite one.

The update operation can be performed using the primitives from Section 3. Given $|S\rangle_{\mathsf{D}}|j\rangle_{\mathsf{C}}|b\rangle_{\mathsf{Z}}$, calculate whether $S \triangle \{j\} \in V$ in a fresh qubit Y. Conditioned on Y, query the value of χ_j and perform the update operation for the data structure. Conditioned on Y not being set, flip the value of Z. Finally, uncompute the value in Y. In the last step, we use that $|S\rangle_{\mathsf{D}}|j\rangle_{\mathsf{C}}$ represents an edge between vertices in V if and only if $|S \triangle \{j\}\rangle_{\mathsf{D}}|j\rangle_{\mathsf{C}}$ does the same.

Having shown how to implement the step of the quantum walk efficiently, let us estimate the required number of steps. The argument is very similar to the one in Theorem 3. We start with the positive case. Assume $\{a_1, \ldots, a_k\}$ is the unique k-collision. Let V_0' denote the set of $S \in V_0$ that are disjoint from $\{a_1, \ldots, a_k\}$, and σ' be the uniform probability distribution on V_0'. Define the flow p from σ' to V_k as follows. For each $S \in V_i$ such that $i < k$ and $S \cap M = \{a_1, \ldots, a_i\}$, define flow $p_e = 1/|V_0'|$ on the edge e from S to $S \cup \{a_{i+1}\} \in V_{i+1}$. Define $p_e = 0$ for all other edges e. Let

$$|\phi\rangle = \sqrt{C_1} \sum_{S \in V_0'} \frac{1}{|V_0'|}|S\rangle - \sum_{e \in E} p_e |e\rangle.$$

This vector is orthogonal to all ψ_u, so it is invariant under the action of $R_B R_A$. Also, $\||\phi\rangle\|^2 = (k + C_1)/|V_0'|$, and $\langle\phi|\varsigma\rangle = \sqrt{C_1/|V_0|}$. Hence,

$$\left\langle \frac{\phi}{\|\phi\|}, \varsigma \right\rangle = \sqrt{\frac{C_1|V_0'|}{(k + C_1)|V_0|}} \sim \sqrt{\frac{C_1}{k + C_1}}$$

where $\sim$ denotes asymptotic equivalence as $n \to \infty$.

In the negative case, define

$$|w\rangle = \sqrt{\frac{C_1}{|V_0|}}\left(\sum_{S\in V_0}\frac{1}{\sqrt{C_1}}|S\rangle + \sum_{e\in E}|e\rangle\right).$$

Similarly to the proof of Theorem 3, we have $\Pi_A|w\rangle = 0$ and $\Pi_B|w\rangle = |\varsigma\rangle$.

Let us estimate $\||w\rangle\|$. The number of edges in E is at most n times the number of vertices in $V_0 \cup \cdots \cup V_{k-1}$. Thus, we have to estimate $|V_i|$ for $i \in [k-1]$. Consider the relation between V_0 and V_i where $S \in V_0$ and $S' \in V_i$ are in the relation iff $S' \setminus S$ consists of i equal elements. Each element of V_0 has at most $n\binom{k-1}{i} = O(n)$ images in V_i, where the constant behind the big-O depends exponentially on k. This is because there are at most n maximal collisions in the input, and for each of them, there are at most $\binom{k-1}{i}$ variants to extend S with. On the other hand, each element in V_i has exactly $r_i + 1$ preimages in V_0. Thus, $|V_i| = O(n|V_0|/r_i)$, so

$$\||w\rangle\| = O\left(\sqrt{1 + n/r_1 + n/r_2 + \cdots + n/r_{k-1}}\right) = O\left(n/\sqrt{\min\{r_1, \ldots, r_{k-1}\}}\right).$$

By Lemma 1, we have that if $\Theta = \Omega(1/\||w\rangle\|)$, then the overlap of $|\varsigma\rangle$ with the eigenvectors of $R_B R_A$ with phase less than Θ can be made at most $1/C_2$ for any constant $C_2 > 0$. Thus, it suffices to use phase estimation with precision Θ if C_1 and C_2 are large enough. By Theorem 2, this requires $O(n/\sqrt{\min\{r_1, \ldots, r_{k-1}\}})$ iterations of the quantum walk. □

4 Quantum Walks with Nested Updates

4.1 Introduction

Given a Markov chain P with spectral gap δ and success probability ε in its stationary distribution, one can construct a quantum search algorithm with cost $\mathsf{S} + \frac{1}{\sqrt{\varepsilon}}(\frac{1}{\sqrt{\delta}}\mathsf{U} + \mathsf{C})$ [12], where S, U, and C are respectively the setup, update, and checking costs of the quantum analog of P. Using a quantum walk algorithm with costs $\mathsf{S}', \mathsf{U}', \mathsf{C}', \varepsilon', \delta'$ (as in [12]) as a checking subroutine straightforwardly gives complexity $\mathsf{S} + \frac{1}{\sqrt{\varepsilon}}(\frac{1}{\sqrt{\delta}}\mathsf{U} + \mathsf{S}' + \frac{1}{\sqrt{\varepsilon'}}(\frac{1}{\sqrt{\delta'}}\mathsf{U}' + \mathsf{C}'))$. Using nested checking [11], the cost can be reduced to $\mathsf{S} + \mathsf{S}' + \frac{1}{\sqrt{\varepsilon}}(\frac{1}{\sqrt{\delta}}\mathsf{U} + \frac{1}{\sqrt{\varepsilon'}}(\frac{1}{\sqrt{\delta'}}\mathsf{U}' + \mathsf{C}'))$.

It is natural to ask if a quantum walk subroutine can be used for the update step in a similar manner to obtain cost $\mathsf{S} + \mathsf{S}' + \frac{1}{\sqrt{\varepsilon}}(\frac{1}{\sqrt{\delta}}\frac{1}{\sqrt{\varepsilon'}}(\frac{1}{\sqrt{\delta'}}\mathsf{U}' + \mathsf{C}') + \mathsf{C})$. In most applications, the underlying walk is independent of the input, so the update operation is simple, but for some applications a more complex update may be useful (as in [22], where Grover search is used for the update). In Section 4.3, we describe an example showing that it is not even clear how to use a nested quantum walk for the update with the seemingly trivial cost $\mathsf{S} + \frac{1}{\sqrt{\varepsilon}}(\frac{1}{\sqrt{\delta}}(\mathsf{S}' + \frac{1}{\sqrt{\varepsilon'}}(\frac{1}{\sqrt{\delta'}}\mathsf{U}' + \mathsf{C}')) + \mathsf{C})$. Nevertheless, despite the difficulties that arise in implementing nested updates, we show in Section 4.4 how to achieve the more desirable cost expression in certain cases, and a similar one in general.

To accomplish this, we extend the quantum walk search framework by introducing the concept of *coin-dependent data*. This allows us to implement nested updates, with a quantum walk subroutine carrying out the update procedure. Superficially, our modification appears small. Indeed, the proof of the complexity of our framework is nearly the same as that of [12]. However, there are some subtle differences in the implementation of the walk.

As in [11], this concept is simple yet powerful. We demonstrate this by constructing a quantum walk version of the learning graph for 3-Distinctness with matching query complexity (up to poly-logarithmic factors). Because quantum walks are easy to analyze, the time complexity, which matches the query complexity, follows easily.

4.2 Quantum Walk Search

In the rest of the paper, let P be a reversible, ergodic Markov chain on a connected, undirected graph $G = (X, E)$ with stationary distribution π and spectral gap $\delta > 0$. Let $M \subseteq X$ be a set of marked vertices. The Markov chain can be used to detect wether $M = \emptyset$ or $\Pr_{x\sim\pi}(x \in M) \geq \varepsilon$, for some given $\varepsilon > 0$.

Quantizing this algorithm leads to efficient quantum algorithms [12]. The quantization considers P as a walk on directed edges of G. We write $(x, y) \in \vec{E}$ when we consider an edge $\{x, y\} \in E$ with orientation. The notation (x, y) means that the current vertex of the walk is x and the *coin*, indicating the next move, is y. Swapping x and y changes the current vertex to y and the coin to x.

The quantum algorithm may carry some data structure while walking on G; we formalize this as follows. Let $0 \notin X$. Define $D : X \cup \{0\} \to \mathcal{D}$ for some Hilbert space $\mathcal{D}$, with $|D(0)\rangle = |0\rangle$. We associate a cost with each part of the algorithm. The cost can be any measure of complexity such as queries or time.

***Setup cost*:** Let S be the cost of constructing

$$|\pi\rangle = \sum_{x\in X} \sqrt{\pi(x)}\,|x\rangle\,|D(x)\rangle \sum_{y\in X} \sqrt{P(x,y)}\,|y\rangle\,|D(y)\rangle\,.$$

***Update cost*:** Let U be the cost of the Local Diffusion operation, which is controlled on the first two registers and acts as

$$|x\rangle\,|D(x)\rangle\,|0\rangle\,|D(0)\rangle \mapsto |x\rangle\,|D(x)\rangle \sum_{y\in X} \sqrt{P(x,y)}\,|y\rangle\,|D(y)\rangle\,.$$

***Checking cost*:** Let C be the cost of $|x\rangle\,|D(x)\rangle \mapsto \begin{cases} -\,|x\rangle\,|D(x)\rangle & \text{if } x \in M \\ |x\rangle\,|D(x)\rangle & \text{otherwise.} \end{cases}$

Theorem 6 ([12]). *Let P be a reversible, ergodic Markov chain on $G = (X, E)$ with spectral gap $\delta > 0$. Let $M \subseteq X$ be such that $\Pr_{x\sim\pi}(x \in M) \geq \varepsilon$, for some $\varepsilon > 0$, whenever $M \neq \emptyset$. Then there is a quantum algorithm that finds an element of M, if $M \neq \emptyset$, with bounded error and with cost $O(\mathsf{S} + \frac{1}{\sqrt{\varepsilon}}(\frac{1}{\sqrt{\delta}}\mathsf{U} + \mathsf{C}))$.*

Furthermore, we can approximately map $|\pi\rangle$ to $|\pi(M)\rangle$, the normalized projection of $|\pi\rangle$ onto $\operatorname{span}\{|x\rangle\,|D(x)\rangle\,|y\rangle\,|D(y)\rangle : x \in M, y \in X\}$, in cost $\frac{1}{\sqrt{\varepsilon}}(\frac{1}{\sqrt{\delta}}\mathsf{U} + \mathsf{C})$.

4.3 Motivating Example

3-*Distinctness.* Suppose the input is a sequence $\chi = \chi_1, \ldots, \chi_n$ of integers from $[q] := \{1, \ldots, q\}$. We model the input as an oracle whose answer to query $i \in [n]$ is χ_i. As in Section 3, we assume without loss of generality that there is at most one 3-collision and that the number of 2-collisions is in $\Theta(n)$. Note that any two 2-collisions not both part of the 3-collision are disjoint.

Quantum Walk for Element Distinctness. In [2], a quantum walk for solving Element Distinctness was presented. This walk takes place on a Johnson graph, $J(n, r)$, whose vertices are subsets of $[n]$ of size r, denoted $\binom{[n]}{r}$. In $J(n, r)$, two vertices S, S' are adjacent if $|S \cap S'| = r - 1$. The data function is $D(S) = \{(i, \chi_i) : i \in S\}$. The diffusion step of this walk acts as

$$|S\rangle |D(S)\rangle |0\rangle \mapsto |S\rangle |D(S)\rangle \tfrac{1}{\sqrt{r(n-r)}} \textstyle\sum_{i \in S, j \in [n] \setminus S} |(S \setminus i) \cup j\rangle |D((S \setminus i) \cup j)\rangle .$$

We can perform this diffusion in two queries by performing the transformation

$$|S\rangle |D(S)\rangle |0\rangle \mapsto |S\rangle |D(S)\rangle \tfrac{1}{\sqrt{r}} \textstyle\sum_{i \in S} |(i, \chi_i)\rangle \tfrac{1}{\sqrt{n-r}} \sum_{j \in [n] \setminus S} |(j, \chi_j)\rangle .$$

We can reversibly map this to the desired state with no queries, and by using an appropriate encoding of D, we can make this time efficient as well.

To complete the description of this algorithm, we describe the marked set and checking procedure. We deviate slightly from the usual quantum walk algorithm of [2] and instead describe a variation that is analogous to the learning graph for Element Distinctness [6]. We say a vertex S is marked if it contains an index i such that there exists $j \in [n] \setminus \{i\}$ with $\chi_i = \chi_j$ (in [2] both i and j must be in S). To check if S is marked, we search $[n] \setminus S$ for such a j, in cost $O(\sqrt{n})$.

Attempting a Quantum Walk for 3-*Distinctness.* We now attempt to construct an analogous algorithm for 3-Distinctness. Let $\mathcal{P}$ denote the set of collision pairs in the input, and $n_2 := |\mathcal{P}|$. We walk on $J(n_2, s_2)$, with each vertex S_2 corresponding to a set of s_2 collision pairs. The diffusion for this walk is the map $|S_2, D(S_2)\rangle |0\rangle \mapsto |S_2, D(S_2)\rangle \frac{1}{\sqrt{r(n_2 - s_2)}} \sum_{\substack{(i,i') \in S_2 \\ (j,j') \in \mathcal{P} \setminus S_2}} |(S_2 \setminus (i, i')) \cup (j, j')\rangle |D((S_2 \setminus (i, i')) \cup (j, j'))\rangle .$

To accomplish this, we need to generate $\frac{1}{\sqrt{s_2}} \sum_{(i,i') \in S_2} |(i, i', \chi_i)\rangle$ and $\frac{1}{\sqrt{n_2 - s_2}} \sum_{(j,j') \in \mathcal{P} \setminus S_2} |(j, j', \chi_j)\rangle$. The first superposition is easy to generate since we have S_2; the second is more difficult since we must find new collisions.

The obvious approach is to use the quantum walk algorithm for Element Distinctness as a subroutine. However, this algorithm does not return the desired superposition over collisions; rather, it returns a superposition over sets that contain a collision. That is, we have the state $\frac{1}{\sqrt{n_2}} \sum_{(i,i') \in \mathcal{P}} |(i, i', \chi_i)\rangle |\psi(i, i')\rangle$ for some garbage $|\psi(i, i')\rangle$. The garbage may be only slightly entangled with (i, i'), but even this small amount of error in the state is prohibitive. Since we must call the update subroutine many times, we need the error to be very small.

Unlike for nested checking, where bounded-error subroutines are sufficient, we cannot amplify the success probability of an update operator. We cannot directly use the state returned by the Element Distinctness algorithm for several reasons. First, we cannot append garbage each time we update, as this would prevent proper interference in the walk. Second, when we use a nested walk for the update step, we would like to use the same trick as in nested checking: putting a copy of the starting state for the nested walk in the data structure so that we only need to perform the inner setup once. To do this here we would need to preserve the inner walk starting state; in other words, the update would need to output some state close to $\binom{n}{s_1}^{-1/2} \sum_{S_1 \in \binom{[n]}{s_1}} |S_1\rangle$. While we might try to recycle the garbage to produce this state, it is unclear how to extract the part we need for the update coherently, let alone without damaging the rest of the state.

This appears to be a problem for any approach that directly uses a quantum walk for the update, since all known quantum walks use some variant of a Johnson graph. Our modified framework circumvents this issue by allowing us to do the update with some garbage, which we then uncompute. This lets us use a quantum walk subroutine, with setup performed only at the beginning of the algorithm, to accomplish the update step. More generally, using our modified framework, we can tolerate updates that have garbage for any reason, whether the garbage is the result of the update being implemented by a quantum walk, or by some other quantum subroutine.

4.4 Quantum Walks with Nested Updates

Coin-Dependent Data. A quantum analog of a discrete-time random walk on a graph can be constructed as a unitary process on the directed edges. For an edge $\{x, y\}$, we may have a state $|x\rangle |y\rangle$, where $|x\rangle$ represents the current vertex and $|y\rangle$ represents the *coin* or next vertex. In the framework of [12], some data function on the vertices is employed to help implement the search algorithm. We modify the quantum walk framework to allow this data to depend on both the current vertex and the coin, so that it is a function of the directed edges, which seems natural in hindsight. We show that this point of view has algorithmic applications. In particular, this modification enables efficient nested updates.

Let $0 \notin X$. Let $D : (X \times \{0\}) \cup \vec{E} \to \mathcal{D}$ for some Hilbert space $\mathcal{D}$. A quantum analog of P with coin-dependent data structures can be implemented using three operations, as in [12], but the update now has three parts. The first corresponds to Local Diffusion from the framework of [12], as described in Section 4.2. The others are needed because of the new coin-dependent data.

Update cost: Let U be the cost of implementing

- Local Diffusion: $|x, 0\rangle |D(x, 0)\rangle \mapsto \sum_{y \in X} \sqrt{P(x, y)} |x, y\rangle |D(x, y)\rangle \ \forall x \in X$;
- The $(X, 0)$-Phase Flip: $|x, 0\rangle |D(x, 0)\rangle \mapsto -|x, 0\rangle |D(x, 0)\rangle \ \forall x \in X$, and the identity on the orthogonal subspace; and
- The Database Swap: $|x, y\rangle |D(x, y)\rangle \mapsto |y, x\rangle |D(y, x)\rangle \ \forall (x, y) \in \vec{E}$.

By cost, we mean any desired measure of complexity such as queries, time, or space. We also naturally extend the setup and checking costs as follows, where $M \subseteq X$ is a set of marked vertices.

Setup cost: Let S be the cost of constructing

$$|\pi\rangle := \sum_{x \in X} \sqrt{\pi(x)} \sum_{y \in X} \sqrt{P(x,y)}\, |x,y\rangle\, |D(x,y)\rangle\,.$$

Checking cost: Let C be the cost of the reflection

$$|x,y\rangle\, |D(x,y)\rangle \mapsto \begin{cases} -\,|x,y\rangle\, |D(x,y)\rangle & \text{if } x \in M, \\ |x,y\rangle\, |D(x,y)\rangle & \text{otherwise,} \end{cases} \quad \forall\, (x,y) \in \vec{E}.$$

Observe that $|\pi\rangle^0 := \sum_{x \in X} \sqrt{\pi(x)}\, |x,0\rangle\, |D(x,0)\rangle$ can be mapped to $|\pi\rangle$ by the Local Diffusion, which has cost $\mathsf{U} < \mathsf{S}$, so we can also consider S to be the cost of constructing $|\pi\rangle^0$.

Theorem 7. *Let P be a Markov chain on $G = (X, E)$ with spectral gap $\delta > 0$, and let D be a* coin-dependent *data structure for P. Let $M \subseteq X$ satisfy $\Pr_{x \sim \pi}(x \in M) \geq \varepsilon > 0$ whenever $M \neq \emptyset$. Then there is a quantum algorithm that finds an element of M, if $M \neq \emptyset$, with bounded error and with cost $O(\mathsf{S} + \frac{1}{\sqrt{\varepsilon}}(\frac{1}{\sqrt{\delta}}\mathsf{U} + \mathsf{C}))$.*

Proof. Our quantum walk algorithm is nearly identical to that of [12], so the proof of this theorem is also very similar. Just as in [12], we define a walk operator, $W(P)$, and analyze its spectral properties. Let $\mathcal{A} := \mathrm{span}\{\sum_{y \in X} \sqrt{P(x,y)}\, |x,y\rangle\, |D(x,y)\rangle : x \in X\}$ and define $W(P) := ((\textsc{Database Swap}) \cdot \mathrm{ref}_{\mathcal{A}})^2$, where $\mathrm{ref}_{\mathcal{A}}$ denotes the reflection about $\mathcal{A}$.

As in [12], we can define $\mathcal{H} := \mathrm{span}\{|x,y\rangle : (x,y) \in (X \times \{0\}) \cup \vec{E}\}$ and $\mathcal{H}_D := \mathrm{span}\{|x,y,D(x,y)\rangle : (x,y) \in (X \times \{0\}) \cup \vec{E}\}$. Also as in [12], there is a natural isomorphism $|x,y\rangle \mapsto |x,y\rangle_D = |x,y,D(x,y)\rangle$, and $\mathcal{H}_D$ is invariant under both $W(P)$ and the checking operation. Thus, the spectral analysis may be done in $\mathcal{H}$, on states without data, *exactly* as in [12]. However, there are some slight differences in how we implement $W(P)$, which we now discuss.

The first difference is easy to see: in [12], the Database Swap can be accomplished trivially by a SWAP operation, mapping $|x\rangle\, |y\rangle\, |D(x)\rangle\, |D(y)\rangle$ to $|y\rangle\, |x\rangle\, |D(y)\rangle\, |D(x)\rangle$, whereas in our case, there may be a nontrivial cost associated with the mapping $|D(x,y)\rangle \mapsto |D(y,x)\rangle$, which we must include in the calculation of the update cost.

The second difference is more subtle. In [12], $\mathrm{ref}_{\mathcal{A}}$ is implemented by applying $(\textsc{Local Diffusion})^\dagger$, reflecting about $|0, D(0)\rangle$ (since the data only refers to a vertex) in the coin register, and then applying (Local Diffusion). It is simple to reflect about $|0, D(0)\rangle$, since $|D(0)\rangle = |0\rangle$ in the formalism of [12]. In [12], this reflection is sufficient, because the operation $(\textsc{Local Diffusion})^\dagger$ fixes the vertex and its data, $|x\rangle\, |D(x)\rangle$, so in particular, it is still in the space $\mathrm{span}\{|x\rangle\, |D(x)\rangle : x \in X\}$. The register containing the coin and its data,

$|y\rangle\,|D(y)\rangle$, may be moved out of this space by (Local Diffusion)†, so we must reflect about $|0\rangle\,|D(0)\rangle$, but this is straightforward.

With coin-dependent data, a single register $|D(x,0)\rangle$ holds the data for both the vertex and its coin, and the operation (Local Diffusion)† may take the coin as well as the entire data register out of the space $\mathcal{H}_D$, so we need to reflect about $|0\rangle\,|D(x,0)\rangle$, which is not necessarily defined to be $|0\rangle\,|0\rangle$. This explains why the cost of $(X,0)$-Phase Flip is also part of the update cost. In summary, we implement $W(P)$ by $((\text{Database Swap})\cdot(\text{Local Diffusion})\cdot((X,0)\text{-Phase Flip})\cdot(\text{Local Diffusion})^\dagger)^2$. □

Nested Updates. We show how to implement efficient nested updates using the coin-dependent data framework. Let $C : X\cup\{0\}\to\mathcal{C}$ be some coin-independent data structure (that will be a part of the final data structure) with $|C(0)\rangle = |0\rangle$, where we can reflect about $\mathrm{span}\{|x\rangle\,|C(x)\rangle : x\in M\}$ in cost C_C.

Fix $x\in X$. Let P^x be a walk on a graph $G^x=(V^x,E^x)$ with stationary distribution π^x and marked set $M^x\subset V^x$. We use this walk to perform Local Diffusion over $|x\rangle$. Let d^x be the data for this walk.

When there is ambiguity, we specify the data structure with a subscript. For instance, $|\pi\rangle_D = \sum_{x,y\in X}\sqrt{\pi(x)P(x,y)}\,|x,y\rangle\,|D(x,y)\rangle$ and $|\pi\rangle_C^0 = \sum_{x\in X}\sqrt{\pi(x)}\,|x,0\rangle\,|C(x),0\rangle$. Similarly, S_C is the cost to construct the state $|\pi\rangle_C$.

Definition 1. *The family* $(P^x,M^x,d^x)_{x\in X}$ implements the Local Diffusion and Database Swap of (P,C) with cost T *if the following two maps can be implemented with cost* T*:*

Local Diffusion with Garbage*: For some garbage states* $(|\psi(x,y)\rangle)_{(x,y)\in\vec{E}}$*, an operation controlled on the vertex* x *and* $C(x)$*, acting as*

$$|x,0\rangle\,|C(x),0\rangle\,|\pi^x(M^x)\rangle_{d^x}\mapsto\sum_{y\in X}\sqrt{P(x,y)}\,|x,y\rangle\,|C(x),C(y)\rangle\,|\psi(x,y)\rangle\,;$$

Garbage Swap*: For any edge* $(x,y)\in\vec{E}$*,*

$$|x,y\rangle\,|C(x),C(y)\rangle\,|\psi(x,y)\rangle\mapsto|y,x\rangle\,|C(y),C(x)\rangle\,|\psi(y,x)\rangle\,.$$

The data structure of the implementation *is* $|D(x,0)\rangle = |C(x),0\rangle\,|\pi^x(M^x)\rangle_{d^x}$ *for all* $x\in X$ *and* $|D(x,y)\rangle = |C(x),C(y)\rangle\,|\psi(x,y)\rangle$ *for any edge* $(x,y)\in\vec{E}$*.*

Theorem 8. *Let* P *be a reversible, ergodic Markov chain on* $G=(X,E)$ *with spectral gap* $\delta>0$*, and let* C *be a data structure for* P*. Let* $M\subseteq X$ *be such that* $\Pr_{x\sim\pi}(x\in M)\geq\varepsilon$ *for some* $\varepsilon>0$ *whenever* $M\neq\emptyset$*. Let* $(P^x,M^x,d^x)_{x\in X}$ *be a family implementing the* Local Diffusion *and* Database Swap *of* (P,C) *with cost* T*, and let* $\mathsf{S}',\mathsf{U}',\mathsf{C}',1/\varepsilon',1/\delta'$ *be upper bounds on the costs and parameters associated with each of the* (P^x,M^x,d^x)*. Then there is a quantum algorithm that finds an element of* M*, if* $M\neq\emptyset$*, with bounded error and with cost*

$$\tilde{O}\left(\mathsf{S}_C+\mathsf{S}'+\tfrac{1}{\sqrt{\varepsilon}}\left(\tfrac{1}{\sqrt{\delta}}\left(\tfrac{1}{\sqrt{\varepsilon'}}\left(\tfrac{1}{\sqrt{\delta'}}\mathsf{U}'+\mathsf{C}'\right)+\mathsf{T}\right)+\mathsf{C}_C\right)\right).$$

Proof. We achieve this upper bound using the quantization of P with the data structure of the implementation, D. We must compute the cost of the setup, update, and checking operations associated with this walk.

Checking: The checking cost $\mathsf{C} = \mathsf{C}_D$ is the cost to reflect about $\text{span}\{|x\rangle|y\rangle|D(x,y)\rangle : x \in M\} = \text{span}\{|x\rangle|y\rangle|C(x),C(y)\rangle|\psi(x,y)\rangle : x \in M\}$. We can implement this in $\mathcal{H}_D$ by reflecting about $\text{span}\{|x\rangle|C(x)\rangle : x \in M\}$, which costs C_C.

Setup: Recall that $|C(0)\rangle = |0\rangle$. The setup cost $\mathsf{S} = \mathsf{S}_D$ is the cost of constructing the state $\sum_{x\in X}\sqrt{\pi(x)}|x\rangle|0\rangle|D(x,0)\rangle = \sum_{x\in X}\sqrt{\pi(x)}|x\rangle|0\rangle|C(x),0\rangle|\pi^x(M^x)\rangle$. We do this as follows. We first construct $\sum_{x\in X}\sqrt{\pi(x)}|x,0\rangle|C(x),0\rangle$ in cost S_C. Next, we apply the mapping $|x\rangle \mapsto |x\rangle|\pi^x\rangle$ in cost S'. Finally, we use the quantization of P^x to perform the mapping $|x\rangle|\pi^x\rangle \mapsto |x\rangle|\pi^x(M^x)\rangle$ in cost $\frac{1}{\sqrt{\varepsilon'}}(\frac{1}{\sqrt{\delta'}}\mathsf{U}' + \mathsf{C}')$. The full setup cost is then $\mathsf{S} = \mathsf{S}_C + \mathsf{S}' + \frac{1}{\sqrt{\varepsilon'}}(\frac{1}{\sqrt{\delta'}}\mathsf{U}' + \mathsf{C}')$.

Update: The update cost has three contributions. The first is the Local Diffusion operation, which, by the definition of D, is exactly the Local Diffusion with Garbage operation. Similarly, the Database Swap is exactly the Garbage Swap, so these two operations have total cost T. The $(X,0)$-Phase Flip is simply a reflection about states of the form $|x\rangle|D(x,0)\rangle = |x\rangle|C(x)\rangle|\pi^x(M^x)\rangle$. Given any $x \in X$, we can reflect about $|\pi^x(M^x)\rangle$ using the quantization of P^x in cost $\frac{1}{\sqrt{\varepsilon'}}(\frac{1}{\sqrt{\delta'}}\mathsf{U}' + \mathsf{C}')$ by running the algorithm of Theorem 7. In particular, we can run the walk backward to prepare the state $|\pi^x\rangle$, perform phase estimation on the walk operator to implement the reflection about this state, and then run the walk forward to recover $|\pi^x(M^x)\rangle$. However, this transformation is implemented approximately. To keep the overall error small, we need an accuracy of $O(1/\sqrt{\varepsilon\delta\varepsilon'\delta'})$, which leads to an overhead logarithmic in the required accuracy. The reflection about $|\pi^x(M^x)\rangle$, controlled on $|x\rangle$, is sufficient because Local Diffusion with Garbage is controlled on $|x\rangle|C(x)\rangle$, and so it leaves these registers unchanged. Since we apply the $(X,0)$-Phase Flip just after applying $(\text{Local Diffusion})^\dagger$ (see proof of Theorem 7) to a state in $\mathcal{H}_D$, we can guarantee that these registers contain $|x\rangle|C(x)\rangle$ for some $x \in X$. The total update cost (up to log factors) is $\mathsf{U} = \mathsf{T} + \frac{1}{\sqrt{\varepsilon'}}(\frac{1}{\sqrt{\delta'}}\mathsf{U}' + \mathsf{C}')$.

Finally, the full cost of the quantization of P (up to log factors) is

$$\begin{aligned}&\mathsf{S}_C + \mathsf{S}' + \tfrac{1}{\sqrt{\varepsilon'}}\left(\tfrac{1}{\sqrt{\delta'}}\mathsf{U}' + \mathsf{C}'\right) + \tfrac{1}{\sqrt{\varepsilon}}\left(\tfrac{1}{\sqrt{\delta}}\left(\tfrac{1}{\sqrt{\varepsilon'}}\left(\tfrac{1}{\sqrt{\delta'}}\mathsf{U}' + \mathsf{C}'\right) + \mathsf{T}\right) + \mathsf{C}_C\right)\\ &= \tilde{O}\left(\mathsf{S}_C + \mathsf{S}' + \tfrac{1}{\sqrt{\varepsilon}}\left(\tfrac{1}{\sqrt{\delta}}\left(\tfrac{1}{\sqrt{\varepsilon'}}\left(\tfrac{1}{\sqrt{\delta'}}\mathsf{U}' + \mathsf{C}'\right) + \mathsf{T}\right) + \mathsf{C}_C\right)\right).\end{aligned}$$ □

If $\mathsf{T} = 0$ (as when, e.g., the notion of cost is query complexity), then the expression is exactly what we would have liked for nested updates.

5 Application to 3-Distinctness

In this section we sketch an alternate proof of Theorem 4, giving a high-level description of the quantum walk algorithm, before summarizing the cost of each required procedure. First we define some notation.

We partition the input space into three disjoint sets A_1, A_2, A_3 of equal size, and assume that if there is a 3-collision $\{i, j, k\}$, then we have $i \in A_1$, $j \in A_2$ and $k \in A_3$. This assumption holds with constant probability, so we need only repeat the algorithm $O(1)$ times with independent choices of the tripartition to find any 3-collision with high probability. Thus, we assume we have such a partition.

For any set $S_1 \subseteq A_1 \cup A_2$, let $\mathcal{P}(S_1) := \{(i,j) \in A_1 \times A_2 : i, j \in S_1, i \neq j, \chi_i = \chi_j\}$ be the set of 2-collisions in S_1 and for any set $S_2 \subset A_1 \times A_2$, let $\mathcal{I}(S_2) := \bigcup_{(i,j)\in S_2}\{i,j\}$ be the set of indices that are part of pairs in S_2. In general, we only consider 2-collisions in $A_1 \times A_2$; other 2-collisions in χ are ignored. For any pair of sets A, B, let $\mathcal{P}(A,B) := \{(i,j) \in A \times B : i \neq j, \chi_i = \chi_j\}$ be the set of 2-collisions between A and B. For convenience, we define $\mathcal{P} := \mathcal{P}(A_1, A_2)$. Let $n_2 := |\mathcal{P}|$. For any set $S_2 \subseteq \mathcal{P}$, we denote the set of queried values by $Q(S_2) := \{(i, j, \chi_i) : (i,j) \in S_2\}$. Similarly, for any set $S_1 \subset [n]$, we denote the set of queried values by $Q(S_1) := \{(i, \chi_i) : i \in S_1\}$.

The Walk. Our overall strategy is to find a 2-collision $(i,j) \in A_1 \times A_2$ such that $\exists k \in A_3$ with $\{i,j,k\}$ a 3-collision. Let $s_1, s_2 < n$ be parameters to be optimized. We walk on the vertices $X = \binom{\mathcal{P}}{s_2}$, with each vertex corresponding to a set of s_2 2-collisions from $A_1 \times A_2$. A vertex is considered marked if it contains (i,j) such that $\exists k \in A_3$ with $\{i,j,k\}$ a 3-collision. Thus, if $M \neq \emptyset$, the proportion of marked vertices is $\varepsilon = \Omega(\frac{s_2}{n_2})$.

To perform an update, we use an Element Distinctness subroutine that walks on s_1-sized subsets of $A_1 \cup A_2$. However, since n_2 is large by assumption, the expected number of collisions in a set of size s_1 is large if $s_1 \gg \sqrt{n}$, which we suppose holds. It would be a waste to take only one and leave the rest, so we replace multiple elements of S_2 in each step. This motivates using a generalized Johnson graph $J(n_2, s_2, m)$ for the main walk, where we set $m := \frac{s_1^2 n_2}{n^2} = O(\frac{s_1^2}{n})$, the expected number of 2-collisions in a set of size s_1. In $J(n_2, s_2, m)$, two vertices S_2 and S_2' are adjacent if $|S_2 \cap S_2'| = s_2 - m$, so we can move from S_2 to S_2' by replacing m elements of S_2 by m distinct elements. Let $\Gamma(S_2)$ denote the set of vertices adjacent to S_2. The spectral gap of $J(n_2, s_2, m)$ is $\delta = \Omega(\frac{m}{s_2})$.

The Update. To perform an update step on the vertex S_2, we use the Element Distinctness algorithm of [2] as a subroutine, with some difference in how we define the marked set. Specifically, we use the subroutine to look for m 2-collisions, with $m \gg 1$. Furthermore, we only want to find 2-collisions that are not already in S_2, so P^{S_2} is a walk on $J(2n/3 - 2s_2, s_1)$, with vertices corresponding to sets of s_1 indices from $(A_1 \cup A_2) \setminus \mathcal{I}(S_2)$, and we consider a vertex marked if it contains at least m pairs of indices that are 2-collisions (i.e., $M^{S_2} = \{S_1 \in \binom{(A_1 \cup A_2)\setminus \mathcal{I}(S_2)}{s_1} : |\mathcal{P}(S_1)| \geq m\}$).

The Data. We store the value χ_i with each $(i,j) \in S_2$ and $i \in S_1$, i.e., $|C(S_2)\rangle = |Q(S_2)\rangle$ and $\left|d^{S_2}(S_1, S_1')\right\rangle = |Q(S_1), Q(S_1')\rangle$. As in Section 3, we use the data structure of [2]. Although technically this is part of the data, it is classical and coin-independent, so it is straightforward. Furthermore, since S_1 is encoded in $Q(S_1)$ and S_2 in $Q(S_2)$, we simply write $|Q(S_1)\rangle$ instead of $|S_1, Q(S_1)\rangle$ and $|Q(S_2)\rangle$ instead of $|S_2, Q(S_2)\rangle$.

The rest of the data is what is actually interesting. We use the state $\left|\pi^{S_2}(M^{S_2})\right\rangle^0_{d^{S_2}}$ in the following instead of $\left|\pi^{S_2}(M^{S_2})\right\rangle_{d^{S_2}}$ since it is easy to map between these two states. For every $S_2 \in X$, let

$$|D(S_2,0)\rangle := |Q(S_2),0\rangle \left|\pi^{S_2}(M^{S_2})\right\rangle^0_{d^{S_2}} = |Q(S_2)\rangle \frac{1}{\sqrt{|M^{S_2}|}} \sum_{S_1 \in M^{S_2}} |Q(S_1)\rangle\,,$$

and for every edge (S_2, S_2'), let $|D(S_2,S_2')\rangle := |Q(S_2), Q(S_2')\rangle\,|\psi(S_2,S_2')\rangle$ where

$$|\psi(S_2,S_2')\rangle := \sum_{\tilde{S}_1 \in \binom{(A_1 \cup A_2)\setminus \mathcal{I}(S_2 \cup S_2')}{s_1 - 2m}} \sqrt{\frac{\binom{n_2 - s_2}{m}}{\binom{|\mathcal{P}(\tilde{S}_1)|+m}{m}\,|M^{S_2}|}} \left|Q(\tilde{S}_1)\right\rangle. \tag{6}$$

We define $|\psi\rangle$ in this way precisely because it is what naturally occurs when we attempt to perform the diffusion.

Summary of Costs. Our setup is similar to that of Section 3. We create a uniform superposition of sets of s_1 queried indices from A_1 in cost $\tilde{O}(s_1)$, search for s_2 elements of A_2 that collide with the queried indices in cost $\tilde{O}\left(s_2\sqrt{n/s_1}\right)$, and measure those queried indices for which we did not find a collision. We add the measured indices to A_3. This leaves a uniform superposition of sets of s_2 2-collisions in $A_1 \times A_2$. We create a uniform superposition of sets of s_1 queried indices from A_1 in cost $\tilde{O}(s_1)$, for a total setup cost of $\mathsf{S}_C + \mathsf{S}' = \tilde{O}\left(s_1 + s_2\sqrt{n/s_1}\right)$.

The update walk costs follow from the above discussion, with $\delta' = \Omega(\frac{m}{s_2})$ (the spectral gap of $J(n_2, s_2, m)$); $\varepsilon' = \Omega(1)$ (the proportion of sets of size s_1 containing $\geq m$ 2-collisions); $\mathsf{U}' = \tilde{O}(1)$; and $\mathsf{C}' = O(1)$, achievable by keeping a count of the number of 2-collisions in the set.

It's not difficult to see that our garbage is symmetric, so our garbage swap is quite straightforward and requires simply moving $O(m)$ already queried elements between data structures. Similarly, the local diffusion with garbage is accomplished by moving $O(m)$ already queried 2-collisions between data structures, thus we have $\mathsf{T} = \tilde{O}(m)$. The checking is accomplished by searching A_3 for an element in collision with one of the stored 2-collisions, giving $\mathsf{C} = \tilde{O}(\sqrt{n})$.

Plugging these into the formula of Theorem 8 gives an upper bound of $\tilde{O}(n^{5/7})$ time complexity, using the optimal values of $s_1 = n^{5/7}$ and $s_2 = n^{4/7}$.

References

1. Buhrman, H., Dürr, C., Heiligman, M., Høyer, P., Santha, M., Magniez, F., de Wolf, R.: Quantum algorithms for Element Distinctness. SIAM Journal on Computing 34, 1324–1330 (2005)
2. Ambainis, A.: Quantum walk algorithm for element distinctness. In: 45th IEEE FOCS, pp. 22–31 (2004)
3. Aaronson, S., Shi, Y.: Quantum lower bounds for the collision and element distinctness problems. Journal of the ACM 51, 595–605 (2004)
4. Reichardt, B.: Reflections for quantum query algorithms. In: 22nd ACM-SIAM SODA, pp. 560–569 (2011)
5. Lee, T., Mittal, R., Reichardt, B., Spalek, R., Szegedy, M.: Quantum query complexity of state conversion. In: 52nd IEEE FOCS, pp. 344–353 (2011)
6. Belovs, A.: Span programs for functions with constant-sized 1-certificates. In: 44th ACM STOC, pp. 77–84 (2012)
7. Lee, T., Magniez, F., Santha, M.: A learning graph based quantum query algorithm for finding constant-size subgraphs. Chicago Journal of Theoretical Computer Science (2012)
8. Zhu, Y.: Quantum query of subgraph containment with constant-sized certificates. International Journal of Quantum Information 10, 1250019 (2012)
9. Lee, T., Magniez, F., Santha, M.: Improved quantum query algorithms for triangle finding and associativity testing. In: ACM-SIAM SODA, pp. 1486–1502 (2013)
10. Belovs, A.: Learning-graph-based quantum algorithm for k-distinctness. In: 53rd IEEE FOCS, pp. 207–216 (2012)
11. Jeffery, S., Kothari, R., Magniez, F.: Nested quantum walks with quantum data structures. In: 24th ACM-SIAM SODA, pp. 1474–1485 (2013)
12. Magniez, F., Nayak, A., Roland, J., Santha, M.: Search via quantum walk. SIAM Journal on Computing 40, 142–164 (2011)
13. Bollobás, B.: Modern graph theory. Graduate Texts in Mathematics, vol. 184. Springer (1998)
14. Chandra, A.K., Raghavan, P., Ruzzo, W.L., Smolensky, R., Tiwari, P.: The electrical resistance of a graph captures its commute and cover times. Computational Complexity 6, 312–340 (1996)
15. Szegedy, M.: Quantum speed-up of Markov chain based algorithms. In: Proc. of 45th IEEE FOCS, pp. 32–41 (2004)
16. Kitaev, A.: Quantum measurements and the abelian stabilizer problem (1995)
17. Cleve, R., Ekert, A., Macchiavello, C., Mosca, M.: Quantum algorithms revisited. Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences 454, 339–354 (1998)
18. Ambainis, A., Childs, A.M., Reichardt, B.W., Špalek, R., Zhang, S.: Any AND-OR formula of size N can be evaluated in time $N^{1/2+o(1)}$ on a quantum computer. SIAM Journal on Computing 39, 2513–2530 (2010)
19. Belovs, A., Lee, T.: Quantum algorithm for k-distinctness with prior knowledge on the input. Technical Report arXiv:1108.3022, arXiv (2011)
20. Kuperberg, G.: Another subexponential-time quantum algorithm for the dihedral hidden subgroup problem (2011)
21. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proc. of 28th ACM STOC, pp. 212–219 (1996)
22. Childs, A.M., Kothari, R.: Quantum query complexity of minor-closed graph properties. In: 28th STACS, pp. 661–672 (2011)