Engineering the Policy-making Life Cycle

Seventh Framework Programme – Grant Agreement 288147

Prototype of the global level policy reasoning system V2

Document type:	Deliverable
Dissemination Level:	PU
Editor:	Marco Gavanelli
Document Version:	1.1
Contributing Partners:	UNIFE, UNIBO
Contributing WPs:	WP3
Estimated P/M (if applicable):	7
Date of Completion:	30 March 2014
Date of Delivery to EC	31 March 2014
Number of pages:	52

ABSTRACT

This document is the official documentation of the prototype of the global level policy reasoning system Version 2. This tool aims to compute optimal energy plans with respect to chosen objectives, and assess the environmental impact of the energy plan in order to support policy makers. It is based on the modeling techniques proposed in deliverable D3.1 [9], and it is focussed on the Regional Energy plan of the Emilia-Romagna region.

Version 2 has a number of additional features beyond those in version 1, including being able to provide quantitative environmental impact values and an assessment of the impact on the environment, human health and global warming. In addition to this, a new algorithm was developed to implement multi-criteria optimisation.

Authors of this document:

Marco Gavanelli[1], Michela Milano[2], Federico Chesani[2], Elisa Marengo[2], Stefano Bragaglia[2]

[1]: EnDiF, University of Ferrara
email: {marco.gavanelli}@unife.it
[2]: DISI, University of Bologna
email:
{michela.milano,federico.chesani,elisa.marengo,stefano.bragaglia}@unibo.it

Contents

This page has been intentionally left blank.

Acronyms

BOM	Business Object Model
CLP	Constraint Logic Programming
CLP($\mathbb{R}$)	Constraint Logic Programming over the Reals
DSS	Decision Support System
GUI	Graphical User Interface
HTML	Hyper-Textual Markup Language
JSP	JavaServer Page
kTOE	kilo Tonnes of Oil Equivalent
MILP	Mixed-Integer Linear Programming
MVC	Model-View-Controller
NO_X	Nitrogen Oxides
PAH	Polycyclic Aromatic Hydrocarbon compounds
Pb	lead
SOA	Service-Oriented Architecture
UI	User Interface
WS SEI	Web Service's Service Endpoint Interface

1 Executive summary

The Global Level Policy Reasoning System is an IT tool which aims to compute optimal energy plans, and to assess the impact of potential policy decisions on aspects such as the environment, the region and its population.

Deliverable D3.4 is the second version of the Prototype of the global level policy reasoning system. This document is attached to the deliverable, and stands as the documentation of the prototype, extending the documentation included in Deliverable D3.2 [6].

The new features of the global optimizer, with respect to version V1 are:

- While the first version was able to provide the environmental assessment of the plan only based on the co-axial matrices (that contain qualitative information), the new version also provides quantitative information, namely the emissions of pollutants in the air by the various types of power plants (Section 4.3.9).
- Some indicators (aggregated values) were calculated to indicate the effect of the plan on three factors: the environment, human health and global warming. (Section 4.3.10).
- Since during the last Regional Energy Plan, renewable energy sources have grown significantly, the forthcoming plans will probably need to close non-renewable energy plants. For this reason, the optimization software was extended to allow for decommissioning existing plants. This introduced some non-linearities in the constraint model. (Section 3.4).
- A new algorithm was used to implement multi-criteria optimization, with an arbitrary number of objective functions, and with a variety of different objective types. In Version V1, there was a simple predicate that performed optimization of exactly two objectives; *air quality*, and *cost* of the plan. In Version V2, the user can select an arbitrary number of objective functions, chosen from
  - any environmental receptor (not just the *air quality*)

  - the cost
  - the total energy produced (divided into two categories: electric or thermal energy)
  - the total installed power (again, divided into electric or thermal energy)
  - one of the emissions (CO_2, NO_X, particulate matter, etc.)
  - an aggregation of the emissions considering *human toxicity*, *acidification* or *greenhouse effect*
  - any linear combination of the above.
- finally, we developed a new web page to test the results of multi-criteria optimization. Note: this is not the official user interface of the global optimizer, that will be developed in Work Package 7.

2 Introduction

Deliverable D3.1 [9] proposed a set of alternatives for addressing the problem of environmental assessment of a Regional plan (the so-called *Strategic Environmental Assessment*), including Causal Probabilistic Logic Programming, Fuzzy logic, and Constraint Logic Programming over the Reals (CLP($\mathbb{R}$)). That deliverable also contained the evaluation of the pros and cons of the various models, and the selection of CLP($\mathbb{R}$) as the best choice amongst the envisaged ones.

The choice of using CLP($\mathbb{R}$) also allowed us to perform regional planning, so the software was also able to *provide plans*, which are thus already assessed, and not only to assess plans proposed by experts.

This deliverable provides the software, as well as all the necessary documentation for installing it, running it, and modifying it to customize it for other domains. The software is currently tailored to the regional energy plan of the Emilia-Romagna region; however, we explain in detail how one can customize it for

- different types of plan (beside the energy plan)
- other regions, possibly including different parameters for the environmental assessment, or different energy sources, etc.
- technological advancements, that might lower the cost of current power sources, or even provide new energy sources.

In order to make the document self-contained, we first recap the constraint model, already provided in deliverable D3.1.

3 The constraint model

In this section, we first describe the problem of Regional Policy Making, as it is currently addressed in the Emilia-Romagna region of Italy, with particular emphasis on the Regional Energy Plan, together with the available data in input (Section 3.1), then we describe how we model the problem into Constraint Logic Programming (Section 3.2).

3.1 Regional Planning and Input Data

Regional Planning is the result of the main policy making activity of European regions. The European Community provides the European Regional Development Fund, that are meant

to support European Regions to reach objectives of growth in research and development expenditure, creation of the knowledge society and dissemination of sustainable development, etc. Within the European Regional Development Fund, each region has a budget distributed by an Operational Programme (OP): an OP sets out each region's priorities for delivering the funds. On the basis of these funds, the region has to define its priorities: in the field of energy, one example of priority is increasing the use of renewable energy sources. Taking as an example the Emilia-Romagna Regional Energy Plan approved in 2007, some objectives of the policy makers are the production of a given amount of energy (400 additional MW from renewable energy sources), while reducing the current greenhouse gas emission percentage by 6.5% with respect to 2003. In addition, the budget constraint limiting the amount of money allocated to the energy plan by the Regional Operational Programme was 30.5M€ in 2007.

Then, a region should decide which activities to insert in the plan. Activities may be roughly divided into six types: (1) infrastructures and plants; (2) buildings and land use transformations; (3) resource extraction; (4) modifications of hydraulic regime; (5) industrial transformations; (6) environmental management. Also, a magnitude for each activity should be decided describing how much of a given activity is performed.

For each activity, the input data includes an outcome (such as the amount of energy produced or consumed) and a cost. Let N_a the number of activities, we have two vectors $\mathbf{Out} = (out_1, \ldots, out_{N_a})$ and $\mathbf{C} = (c_1, \ldots, c_{N_a})$, where each element is associated to a specific activity and represents the outcome and cost per unit of an activity.

There is a distinction between primary and secondary activities. Primary activities are those that are specifically targeted within a plan, and secondary activities are those that must be performed in order to support primary ones. There are constraints linking activities: for instance if a regional plan decides to build three biomass power plants (primary activities for an energy plan), each of these plants should be equipped with proper infrastructures (secondary activities like streets, sewage or possibly a small village nearby, power lines). Other activities are linked to the *decommissioning* of other activities: e.g. activity *"Reduction of fossil fuel power plants"* is a secondary activity that is positive if one of the fossil fueled power plants has a negative value.

The dependencies between the activities are described in two square $N_a \times N_a$ matrices; $\mathcal{D}^+$ and $\mathcal{D}^-$. $\mathcal{D}^+$ contains elements d_{ij}^+ which represent the magnitude of activity j per unit of activity i, and $\mathcal{D}^-$ contains elements d_{ij}^- that represent the magnitude of activity j per unit reduction of activity i.

Also, each activity has impacts on the environment in terms of positive and negative pressures: example of positive pressure is the increased availability of energy, while a negative pressure is the production of pollutants. Pressures are themselves linked to environmental receptors such as the quality of the air, of surface water. There are constraints on these pressures and receptors: for example the maximum amount of greenhouse gas emissions of the overall plan.

The policy maker also has to take into account impacts on the environment, economy and

society. One of the instruments used in Emilia-Romagna are the so called *coaxial matrices* [3], that are a development of the network method [13].

One matrix $\mathcal{M}$ defines the dependencies between the above mentioned activities contained in a plan and *impacts* (also called *pressures*) on the environment. Each element m_j^i of the matrix $\mathcal{M}$ defines a qualitative dependency between the activity i and the impact j. The dependency can be *high*, *medium*, *low* or *null*.

The second matrix $\mathcal{N}$ defines how the impacts influence environmental receptors. Each element n_j^i of the matrix $\mathcal{N}$ defines a qualitative dependency between the impact i and an environmental receptor j. Again the dependency can be *high*, *medium*, *low* or *null*. Examples of environmental receptors are the quality of surface water and groundwater, quality of landscapes, wildlife wellness and so on.

The matrices currently used in Emilia-Romagna contain 116 activities (including primary and secondary activities), 29 negative impacts, 19 positive impacts and 23 receptors, and have been used to assess 11 types of plans (those for which the assessment is legally required, i.e. Agriculture, Forest, Fishing, Energy, Industry, Transport, Waste, Water, Telecommunication, Tourism, Urban and Environmental plans).

3.2 A CLP model

To design a constraint-based model, variables, constraints and objectives have to be defined. Variables represent decisions that have to be taken. Given a vector of activities $\mathbf{A} = (a_1, \ldots, a_{N_a})$, to each activity we associate a variable G_i that defines its magnitude. The magnitude could be represented either in an absolute way, as the amount of a given activity, or in a relative way, as a percentage with respect of the existing quantity of the same activity. Absolute representation is used in this deliverable.

As stated above, we distinguish primary from secondary activities: let A^P be the set of indexes of primary activities (i.e. those of main importance in a given plan) and A^S the set of indexes of secondary activities (i.e. those supporting the primary activities). The dependencies between primary and secondary activities are considered by the constraint:

$$\forall j \in A^S \quad G_j = \sum_{i \in A^P} K_{ij}$$

where

$$K_{ij} = \begin{cases} d_{ij}^+ \cdot G_i & \text{if} \quad G_i \geq 0 \\ d_{ij}^- \cdot (-G_i) & \text{if} \quad G_i < 0 \end{cases}$$

i.e. we have different coefficients d^+ and d^- for the positive and negative dependency.

Given a budget B_{Plan} available for a given plan, we have a constraint limiting the cost of the overall plan as follows

$$\sum_{i=1}^{N_a} G_i \, c_i \leq B_{Plan}. \tag{1}$$

Also, given an expected outcome out_{Plan} of the plan we have a constraint ensuring the outcome is reached:

$$\sum_{i=1}^{N_a} G_i \, out_i \geq out_{Plan}.$$

For example, in an energy plan the outcome can be to have more energy available in the region, so out_{Plan} could be the increased availability of electrical power (e.g. in megawatts). In such a case, out_i will be the production in MW for each unit of activity a_i.

Concerning impacts of the regional plan, we sum up the contributions of all the activities and obtain the estimate of the impact on each environmental pressure:

$$\forall j \in \{1, \ldots, N_p\} \quad p_j = \sum_{i=1}^{N_a} m_j^i \, G_i. \tag{2}$$

In the same way, given the vector of environmental pressures $\mathbf{P} = (p_1, \ldots, p_{N_p})$, one can estimate the influence on the environmental receptor r_i by means of the matrix $\mathcal{N}$, that relates pressures with receptors:

$$\forall j \in \{1, \ldots, N_r\} \quad r_j = \sum_{i=1}^{N_p} n_j^i p_i. \tag{3}$$

Concerning objectives, there are a number of possibilities suggested by planning experts. From an economic perspective, one can decide to minimize the overall cost of the plan (that is anyway subject to budget constraints). Clearly, in this case the most economic energy sources are preferred, despite their potentially negative environmental effects (though these themselves could also be constrained within the plan). On the other hand, one could maintain a fixed budget and maximize the produced energy. In this case the most efficient energy sources will be pushed forward. Or, the planner could prefer a *green* plan and optimize environmental receptors. For example, one can maximize, say, the air quality, or the quality of the surface water. In this case, the produced plan decisions are less intuitive and the system we propose is particularly useful. The link between decisions on primary and secondary activities and consequences on the environment are extremely complex, and cannot easily be considered manually. Clearly, more complex objectives can be pursued, by properly combining the above mentioned aspects.

3.3 The regional energy plan

We can now describe how to adapt the general model for regional plan described above into the model for designing a regional energy plan. The first step is to identify primary and secondary activities. In the context of a regional energy plan, the environmental and planning experts defined the following distinction. Primary activities are those capable of producing energy, namely renewable and non-renewable power plants. Secondary activities are those supporting the energy production, such as activities for energy transportations (e.g. power lines), and infrastructures supporting the primary activities (e.g. dams, yards).

There are a number of other aspects that need to be taken into account when considering regional energy plans. One important aspect is the energy source diversification: one should

not use a single energy source, but should cover both renewable and non renewable energy sources. This requirement comes from fluctuations of the price and availability of the various resources. For this reason, we have constraints on the minimal proportion F_i of the total power produced by each source i.

In addition, each region has its own geo-physical characteristics. For instance, some regions are particularly windy, while some others are not. This poses constraints on the maximum power that can be produced by a wind generation. Another example is that hydroelectric power plants can only be built with a very careful consideration of environmental impacts, the most obvious being the flooding of vast areas of land. This poses constraints on the maximum power that can be produced by a given energy source i.

Finally, the regional priorities should conform with European guidelines such as the 20-20-20 initiative aimed at achieving three ambitious targets by 2020: reducing its greenhouse gas emissions by 20%, producing a 20% share of its final energy consumption from renewable sources, and improving its energy efficiency by 20%. For this reason, we can impose constraints on the minimum amount of energy L_{ren} produced by renewable energy sources whose set is referred to as ${A^P}_{ren}$. The constraint that we can impose is

$$\sum_{i \in {A^P}_{ren}} G_i out_i \geq L_{ren}.$$

3.4 Decommissioning existing plants

A Regional Energy Plan might prescribe to decommission existing power plants; in this case, the magnitude of the activity can be negative, and the model becomes non linear, in fact:

- the **cost** of a decommissioning is not simply the opposite of the construction cost. In the current model, it is assumed that the decommissioning cost for power plants is 0; in future work, more refined pricing schemes can be adopted.
- the **secondary activities** associated to decommissioning a plant are not the opposite of the activities needed to build a plant. In the current model, it is assumed that assets built by activities for supporting the plant will not be decommissioned, so a negative magnitude for the primary activity does not impose a negative magnitude for secondary activities. This means that a primary activity with negative magnitude will usually have contribution zero onto secondary activities.
  Note, however, that there can be secondary activities that are triggered by a primary activity being *negative*. For example, activity *"Reduction of fossil fuel power plants"* is considered as a secondary activity that is activated by fossil power plants (e.g. *coal power plants*) that have a negative magnitude.
- **emissions and environmental pressures:** a negative magnitude for a type of power plant will give a negative value for emissions, because these are pollutants that are not emitted. The same effect can be felt with the pressures. In this case, the relations between magnitude and pressures/emissions remains linear.

The nonlinear relations are implemented using integer variables, as usual in Integer Linear Programming. It is worth noting that having nonlinear relations means a higher complexity (under the usual assumptions: it is well-known that linear programming is polynomially

solvable, i.e. it belongs to the complexity class *P*, while integer linear programming is in complexity class *NP*, commonly believed to be more complex).

If the bounds are positive, the software will avoid adding integer variables. However, if for some primary activity one of the bounds is negative, for that activity an integer, 0-1 variable will be created. That variable can be considered as a Boolean, and takes value 1 if the corresponding activity is greater or equal to zero, and takes value 0 otherwise (in case a plant is decommissioned).

3.5 Multi-Objective Optimization

In many practical cases, the user is interested in optimizing more than one objective function at the same time. In the case of regional energy planning, the user could be interested in minimizing the cost, maximizing the quality of each receptor, maximizing the produced energy, etc. In case the objectives are conflicting (e.g. minimizing the cost and maximizing the produced energy), there is not a unique optimal solution, but there are various solutions that are not comparable. Usually, the following definition is given.

Definition 1 *Given a set of objective functions $f_1, f_2, \ldots, f_k$ that the user wants to minimize, a solution X* **dominates** *another solution Y if*

$$\forall i, 1 \leq i \leq k, \qquad f_i(X) \leq f_i(Y)$$

i.e. solution X is better (or equal) to Y with respect to all objectives.

Obviously, the user is not interested in solutions that are dominated, but only in non-dominated solutions, i.e. in solutions that are not dominated by any other solutions. The non-dominated solutions form the so-called *non-dominated set* or *Pareto front*.

4 The CLP(R) software

In order to run the Global Optimizer it is necessary to download and install ECL^{i}PS^e and then to download and run the Global Optimizer. Instructions on how to that are reported hereafter in Sections 4.1 and 4.2. In order to try the Global Optimizer without installing it, it is possible to access to the Global Optimization Module Testing Page available online. This service is described in Section 6.1.

4.1 Downloading and installing ECL^{i}PS^e

The Global Optimizer software runs on top of ECL^{i}PS^e [11, 1] version 6 or greater. ECL^{i}PS^e is an open-source Constraint Logic Programming (CLP) environment, available for many platforms and operating systems, and it can be downloaded

- either from the ECL^{i}PS^e web site `http://www.eclipseclp.org/`
- or from SourceForge `http://sourceforge.net/projects/eclipse-clp/`

Using SourceForge, one can go to `http://sourceforge.net/projects/eclipse-clp/files/`, click on *Binaries XXX*, where XXX is the platform one is installing to (it can be PPC MacOSX, x86-64 MacOSX 64bit, x86 MacOSX, Windows, Linux x86_64, Linux 32bit, Sparc Solaris or x86 Solaris).

Then, one should select the version of ECL^{i}PS^e he/she wishes to install, and finally is presented with a list of files as the following:

- `if_osiclpcbc.tgz`
- `tcltk.tgz`
- `eclipse_rt.tgz`
- `eclipse_doc.tgz`
- `eclipse_misc.tgz`
- `eclipse_basic.tgz`
- `UNPACK`
- `README_UNIX`
- `README`

Please download *all* files in a chosen directory.

In particular, the Global Optimizer uses the `eplex` library [12], which requires an external Mixed-Integer Linear Programming (MILP) solver, so when installing ECL^{i}PS^e, one should make sure to also install the libraries for interfacing with external MILP solvers. By default, ECL^{i}PS^e provides an interface to the open-source CLP/CBC solver, simply by installing the optional packet `if_osiclpcbc.tgz`, available on the ECL^{i}PS^e web site.

After downloading all the files, the installation is different according to the platform being used. The file named `'README'` provides detailed instructions for installation. To simplify the installation, the instructions (taken from the `README` files) for Windows and Linux are below.

Windows For the most common Windows install configurations, use the ECL^{i}PS^e Windows Installer. To do so, download and execute the single file

`ECLiPSe<Version>_<Build>.exe`

Run this installer program as a user with "administrator" rights.

The installer contains the ECL^{i}PS^e kernel, basic libraries, and the following optional packages:

- Documentation (html, txt, pdf)
- An interface to COIN-OR's open source solvers CLP and CBC
- An interface to a version of Dash Optimization's XPRESS-MP solver
- 3rd party libraries
- A bundled standalone distribution of GraphViz (to support the visualisation tools)
- A bundled (but not standalone) distribution of Tcl/Tk (to support the TkECLiPSE GUI)

For installation, double click on `ECLiPSe<Version>_<Build>.exe` and the install wizard will guide you through the installation. By choosing to install 3rd party libraries from the wizard, the `eplex` library you need to run the Global Optimizer will be installed.

Then try to launch `tkeclipse` via the Start menu, or look at the documentation, e.g. the tutorial.

Linux After downloading all files in a folder, open a shell in that folder, then execute:

```
chmod a+x UNPACK
```

```
./UNPACK
```

```
./RUNME
```

Now the software is installed; the installation terminates suggesting to add a directory to the PATH environment variable; please follow the instructions of you OS to do so.

After doing that, ECL^{i}PS^e can be run either from the command line, with command

```
eclipse
```

or with the graphical interface, by running

```
tkeclipse
```

4.2 Downloading and installing the Global Optimizer

The up-to-date version of the Global Optimizer can be directly obtained from the public SVN repository

```
https://svn.ing.unibo.it/svn/ai/OptModel/trunk/
```

The software can be downloaded with an SVN client; the following command downloads the directory tree necessary to run the software and generate the documentation:

```
svn co https://svn.ing.unibo.it/svn/ai/OptModel/trunk/
```

Otherwise, one can download the whole package (version 1.1) from

```
http://www.epolicy-project.eu/system/files/PUBLIC/deliverables/D3.4.zip
```

In the following, we will assume a basic knowledge of the Prolog language. For users who are novices to Prolog, an example of using the Global Optimizer is provided in Section 5.

4.3 User manual

The program is compiled by consulting file `modello.ecl`. If you are using ECL^{i}PS^e at command line, after running ECL^{i}PS^e type the following command:

```
?- [modello].
```

From the graphical front-end the program can be compiled by choosing compile from the file menu. Section 5 shows the use of the command line and the graphical front.

4.3.1 Defining limits for energy sources

After compiling the program, the limits for the primary activities should be defined. They are defined by means of `dynamic` predicates, so they can be conveniently provided through the `assert/1` command of Prolog. For each of the possible energy sources, a fact

$$\texttt{min\_max\_source}(Activity, Min, Max).$$

should be provided, where $Activity$ is the name of the energy source, and Min and Max are, respectively, the minimum and maximum number of MW of that energy source that can be installed. It is important to notice that this value considers the new installations, and not the total at the end of the period (i.e. this value should not include any existing power station capacity).

NOTE: the bounds could also be *negative*: in this case, we are decommissioning existing power plants (or not using them).

To simplify the input of the bounds, a number of predicates have been defined for the significant cases of energy plans of the Emilia-Romagna region. These predicates are

- `plan`$(Year)$. $Year$ can currently be either 2013 or 2020. Invoking `plan`(2013), for each energy source the minimum and maximum value are fixed to the value proposed by the technicians of the Emilia-Romagna region in the 2013 Regional energy plan [10], which includes also a forecast for 2020.
  As both the limits are fixed to the same value, this predicate can be used to re-generate the plans provided in the Regional Energy Plan [10], and to provide their environmental assessment, but it is not used for deciding a new plan, nor for generating alternatives.
- `plan`$(Type, Year)$. $Type$ can either be ele (for *Electrical Energy*) or $therm$ (for *Thermal Energy*). The meaning is the same as the previous item, but here we consider separately the plan for electrical and for thermal energy; for example, in the case of electrical energy, all energy sources that do not provide electrical energy are fixed to zero.
- `plan_amplified`$(Year)$. The $Year$ parameter has the same meaning as before. In this case, the asserted limits are a widening of the limits used for `plan`; they are typically amplified by a factor 2 (meaning that the Min is half the value proposed by the technicians for the Regional Energy Plan, while the Max is twice that value). For some of the energy sources limits are included as they were given by the technicians, that consider, for example, subsidies already planned by the Region in other plans (e.g. the Agriculture plan) to foster energy sources (e.g. biomasses).
- `plan_amplified`$(Type, Year)$. Similarly to the previous item the limits are a widening of the limits used for `plan`, but in this case the plans for electrical and thermal energy are considered separately. This is achieved by means of the parameter $Type$ that can either be ele or $therm$.

4.3.2 Computing Plans

To compute plans, the main predicate is `compute_plan`. It accepts a number of parameters, not all of them are required. The predicate has various forms, to ease the insertion of optional parameters:

- `compute_plan`$(Objective, Budget, ElectricalOut, ThermalOut)$
- `compute_plan`$(Objective, Budget, ElectricalOut, ThermalOut, Receptors, Cost)$
- `compute_plan`$(Objective, Budget, ElectricalOut, ThermalOut, Receptors, Cost, AddConstr)$
- `compute_plan`$(Objective, Budget, ElectricalOut, ThermalOut, Receptors, Cost, AddConstr, ObjValue)$

where the parameters are:

Objective (Required, Input) The objective is the function that the user wants to optimize. It is a required input parameter, and it can be in the form $min(Term)$ or $max(Term)$, where $Term$ is a linear combination of (one or more of) the following:

cost : The total cost of the activities that are performed in the plan. This cost includes both the cost for the region (public cost) and the cost incurred by private stakeholders.

energy(electric) or electric : The electric energy produced in a year, in kilo Tonnes of Oil Equivalent (kTOE).

energy(thermal) or thermal : The thermal energy produced in a year, in kTOE.

power(electric) : The total installed power of electric energy, in MWe.

power(thermal) : The total installed power of thermal energy, in MW.

rec(X) : where X is the index of one of the receptors, i.e. it can be a number ranging from 1 to N_r. As an example, currently the matrices used by ARPA for the strategic environmental assessment contain 23 receptors.

emission(X) : X can be

- either a number, and in this case it is supposed to be the index of one of the pollutants considered in the emissions (see Section 4.3.9)
- or a string, corresponding to one of the pollutants considered in the emissions (see Section 4.3.9).

best(I), worst(I), typ(I) or simply I where I is one of the indicators in Section 4.3.10), currently `acidification`, `global_warming` or `human_toxicity`. Due to uncertainties in the interpretation of the available data (see Section 4.3.10), these indicators could hold a range of values. $best(I)$ optimizes the best case from this range (the one in which the indicator is minimal), $worst(I)$ the worst-case, $typ(I)$ the typical case. Using simply the name of the indicator I is a shorthand for $worst(I)$, as the most conservative case should be used as default.

For example, valid objective functions are

- $min(cost)$: finds the plan with minimum cost
- $max(electric)$: finds the plan that provides the maximum possible energy
- $max(rec(9))$: finds the plan with the best possible value for receptor 9 (that, with the current receptors used by ARPA, is the *air quality*)
- $min(emission(3))$: finds the plan with the minimum of emission of pollutant number 3, in the considered set (either ISPRA or INEMAR, see Section 4.3.9)
- $min(emission("NOx"))$: finds the plan with the minimum emission of NOx (assuming NOx is one of the pollutants considered in the adopted database (ISPRA or INEMAR)).
- $max(0.5 * rec(9) - 0.5 * cost)$: linear combination of two parameters; intuitively the solver should try to have a high air quality and a low cost. Of course, deciding the coefficients is not trivial.

Budget (Optional, Input) The budget for the plan, in M€. The $cost$ (intended as in the $Objective$, see previous item) should always satisfy the constraint $Cost \leq Budget$. The budget is optional: in case the user does not want to specify a budget, this parameter can be left as a variable (for example, the unnamed variable '`_`' in Prolog syntax).

ElectricalOut,ThermalOut (Optional, Input) These two input parameters, when specified, provide the energy (in kTOE) that should be produced as Electrical Energy and as Thermal Energy.

Receptors (Optional, Output) Provides in output the list of values representing receptors in the optimal plan. It is an optional parameter, and, if the user is not interested in the value of the receptors, the unnamed variable '_' of Prolog can be used for this parameter.

Cost (Optional, Output) Provides in output the cost of the plan

AddConstr (Optional, Input) This parameter is a list that contains further constraints that should be satisfied. The constraints are linear constraints of the form $Term >= Term$, $Term =< Term$ or $Term =:= Term$, where $Term$ has the same syntax given for the *Objective* function.

This parameter can be used to give minimum/maximum values for $cost$, $electric$ or $thermal$ energy or for each receptor or emission, as well as linear combinations of them; e.g. the following is syntactically accepted: $rec(1) >= 0.3 * cost + 2 * electric$.

In order to simplify the input of these data, other predicates are provided, for the typical values of the Regional Energy Plan with forecasts for 2013 and 2020 of the Emilia-Romagna Region. These predicates are

ele(α) invokes predicate `compute_plan` with the energy requirement for electrical energy as that in the Regional Energy Plan of the Emilia-Romagna region. Can be invoked, e.g. in conjunction with the aforementioned predicates `plan`$(ele, Year)$ or `plan_amplified`$(ele, Year)$. The number α should be a real number between 0 and 1, and the considered objective is $\max((1-\alpha)rec(9) - \alpha cost)$, where $rec(9)$ refers to *air quality*.

term(α) invokes predicate `compute_plan` with the energy requirement for thermal energy. Can be invoked in conjunction with `plan`$(term, Year)$ or `plan_amplified`$(term, Year)$. The meaning of α is the same as for predicate ele.

p(α) invokes predicate `compute_plan` with the energy requirement taken from the Regional Energy Plan of the Emilia-Romagna region, considering both electrical and thermal energy. The meaning of α is the same as for predicate ele.

Examples Assuming the limits for energy sources have been defined (for example with `plan_amplified`(2013)), the following goals can be issued:

- `compute_plan`$(min(cost), \_, 177, 296)$.
  Computes the plan with minimal cost achieving 177kTOE of electrical energy and 296kTOE of thermal energy, without any budget restriction.
- `compute_plan`$(max(electric), 3000000000, \_, \_)$.
  Computes the plan that produces maximal electric energy, provided a budget of 3000000000€.
- `compute_plan`$(max(electric + thermal), \_, \_, \_, \_, \_, [thermal >= 2 * electric])$.
  Computes the plan that maximizes the total energy produced (the sum of electric energy and thermal energy) such that the thermal energy is at least twice the electric energy produced.
- `compute_plan`$(min(emission("NOx")), \_, \_, \_, \_, \_, [rec(1) >= 0])$.
  Computes the plan that minimizes the emission of Nitrogen Oxides (NO_X) such that receptor 1 is not worsened (it takes values greater than or equal to 0).

4.3.3 Output results

Beside the usual result of Prolog predicates, i.e. the binding for output parameters, the `compute_plan` predicate can provide further data in the form of csv files (easily readable through common spreadsheets, like Microsoft Excel, LibreOffice/OpenOffice Calc, GNUmeric, etc.); this data can be printed on standard output, on file or on streams. Predicate `set_print_options`$(NewOptions)$ allows to set the desired output. Specifically, $NewOptions$ is a list of options including the following:

- an empty list will result in no output being produced.
- csv_to_file: print csv-formatted output to the standard files.
- csv_to_stdout: print csv-formatted output to the standard output.
- $csv\_to\_stream(Stream)$: print csv-formatted output to the specified stream.
- $generic\_to\_file$: print tabular-formatted output to the standard files.
- $generic\_to\_stdout$: print tabular-formatted output to the standard output.
- $generic\_to\_stream(Stream)$: print tabular-formatted output to the specified stream.

The outputs generated by the Global Optimizer are in Italian, because the use case being considered is that of the energy policy in the Emilia-Romagna region, and the outputs should be understandable to the policy makers of this region. For the sake of comprehension, in this document the outputs are translated in English. Please note that the output of the Global Optimizer are not the final output of the ePolicy application, and that the final user interface will support the English translation as well.

Below is a list of files that can be generated.

`result.txt` Predicate `compute_plan` provides on standard output the following information, which can also be printed on file `result.txt`:

- value of each activity, including its measuring unit (for those activities that have a measuring unit). The primary activities are shown in boldface, if the host O.S. allows for printing of special characters, e.g. in Linux. Figure 1 contains an excerpt of this section of the output.
- value of each pressure. See Figure 2 for an excerpt of the output.
- value of each receptor (Figure 3).
- various objectives that can be of interest for the user (Figure 4), including:
  - the target formula to be optimised, which was input by the user, and its optimal value
  - the total cost of the plan
  - the total electrical energy.

`energies.csv` This file shows the electrical and thermal energy produced (in kTOE) for each of the energy sources. For example, using as limits those given by `plan_amplified`(2013) and as goal $p(0.5)$, file `energies.csv` contains:

```
====================== Activities ======================
Sewers:  9.41002178337093km
Water Supply Systems:  15.4475217833709km
Sewage treatment and wastewater treatment plants:  89.6 thousands of
equivalent inhabitants
Waste storage:  89.1502178337093 thousands of equivalent inhabitants
```
Thermoelectric Biomass Plants: 219.690468827907MWe
Biomass-based Thermal Plant: 671.311709509186MW
```
Aerial Power Line Supports:  1319.00217833709km
Aerial Power Lines:  1319.00217833709km
Underground Power Lines:  1.31900217833709km
Plants for Electricity Transformation (Substations/Transofrmers):  1.319km
```
Windmill Electrical Generators: 20.0MWe
```
Gas, Oil and Vapor Pipelines:  671.316709509186km
Lighting Systems:  91.1502178337093km
Tunnel Ventilation Systems:  0.03
Houses and Residential Areas:  693.2837563919771000m^3
Squares and Yards:  89.8032178337093ha
Roads:  1.49900217833709km
Paths:  1.49900217833709km
```
Small Hydroelectric Plants: 3.0MWe
```
Artificial Lake for Multiple Uses:  0.031000m^3
Dams, dikes, beams, thresholds:  0.03m^3
```
Photovoltaic Plants: 400.0MWe
Solar Thermal Panels: 37.5MW
Superficial Geothermal Plants: 5.0MW
Thermodynamic Solar Plants: 5.0MWe
```
Construction Sites (artifacts, traffic):  0.03
Fences (industrial areas):  13.0205217833709km
Disposal of Obsolete Facilities:  8.97585928337093(1000m^3)
Caves and Mines:  0.03mc
Excavation and Soil Movements:  16.2600217833709mc
Stores for Excavation Resulting Materials:  0.008m^3
Groundwater Extraction Plants:  896.007178337093m^3
Well Drilling:  0.5km of well
Water Derivation Works:  0.896002178337093km
new Riverbeds (also correction existing riverbeds):  0.0896002178337093km
⋮
```

Figure 1: Excerpt of the section on Activities from a sample output of predicate `compute_plan`.

```
====================== Pressures ======================
1.  Energy transformation from non-renewable sources:  1189.92779218242
2.  Lithoid material consumption:  95.4096260569318
3.  Soil consumption/alteration:  1977.79100492174
4.  Water consumption:  1389.34348000493
5.  Substantial change of water flows:  936.28214607949
6.  Alteration in surface waters flows:  479.893460442804
7.  Alteration in groundwater flow and filtration:  1095.22781273687
8.  Intercept and alteration of nearshore currents:  0.002
9.  Wastewater dumping, water pollution:  2233.41406347513
10.  Spreading of hazardous materials:  1367.6365139551
11.  Waste production:  1663.6967582833
12.  Gas and dust emissions in atmosphere:  2751.36116421547
13.  Odor production:  1056.62681512071
14.  Noise production:  551.343070072278
15.  Vibrations production:  477.916405607828
16.  Electro-magnetic field production:  991.230137020325
17.  Ionizing radiation production:  22.8442548940947
18.  Heat transfer in air:  1189.72108594999
19.  Light pollution:  1455.72971978123
20.  Alteration of landscapes perceptions:  2794.47799974232
21.  Alteration of vegetation cover:  2529.00614416692
22.  Habitat fragmentation.:  1948.4804917872
23.  Attraction of unwanted organisms:  784.882817535685
24.  Introduction of exotic flora:  658.950494963014
25.  Transformation of urban functions:  2909.71710852415
26.  Attraction of non planned infrastructures:  813.181234887592
27.  Risk of serious accidents:  2364.84899965763
28.  Road accidents:  1663.25432469415
29.  Use of explosives:  0.2875
30.  Creation of income, employment opportunities:  4660.3487467519
31.  Enhancement/creation of tangible assets:  5110.7785891288
32.  Improved operation of facilities/services:  6155.81729275774
33.  Creation of access opportunities:  518.268086257177
34.  Improved waste management:  1076.00177010558
35.  Control/reduction of air pollution:  371.91405425207
36.  Control/reduction of green house gas emissions:  1549.77609164354
37.  Control/reduction of water pollution:  621.173707359073
38.  Control/reduction of noise:  242.373602473276
39.  Control/reduction of ionizing radiation:  114.382922618887
40.  Control/reduction of non-ionizing radiation:  401.717789494567
41.  Saving/production of renewable energy:  1782.98106578519
⋮
```

Figure 2: Excerpt of the section on Pressures from a sample output of predicate `compute_plan`.

```
====================== Receptors ======================
1.  Subsidence limitation:  48.7491680507435
2.  Embankment stability:  -3644.56862849131
3.  Stability of coasts or seafloor:  -392.823165376105
4.  Stability of river banks and beds:  -1493.21427445758
5.  Soil quality:  -3609.9402259488
6.  Quality of sea water:  -2094.03972784824
7.  Quality of inland surface water:  -3603.92777322148
8.  Groundwater:  -5096.01723947927
9.  Air quality:  -4100.68627266995
10.  Quality of climate:  -1161.74560627152
11.  Wellness of terrestrial vegetation:  -7143.85439448481
12.  Wellness of wildlife:  -9722.04252109749
13.  Wellness of aquatic plants:  -7513.74158156845
14.  Wellness and health of mankind:  786.159566677553
15.  Quality of sensitive landscapes:  -10598.705514049
16.  Cultural/historical heritage value:  -6819.45183888558
17.  Recreation resources accessibility:  -915.132470840773
18.  Water availability:  -4664.35480988736
19.  Availability of agricultural fertile land:  -4410.00596335353
20.  Lithoid resource availability:  872.635576256134
21.  Energy availability:  165.82269222421
22.  Availability of productive resources:  12735.6272529762
23.  Value of material goods:  9930.7325565569
```

Figure 3: Section on Receptors from a sample output of predicate `compute_plan`.

```
====================== Objectives ======================
Objective:  max((1 - 0.5) * ric(9) - 0.5 * cost) = -1363566163.55179
cost(2727128226.41731)
energy(177.2698956357)
```

Figure 4: Section on Objectives from a sample output of predicate `compute_plan`.

Activity	Quantity	Subsidence limitation	Embankments stability	Stability of coasts or seafloor	Stability of river banks and beds	Soil quality	…
Sewers	9.41002178337093	-2.35250544584273	-5.88126361460683	0.588126361460683	-2.35250544584273	-6.46938997606752	
Water Supply Systems	15.4475217833709	-5.7928206687641	-13.5165815604496	-4.82735055730342	-11.5856413375282	-6.75829078022478	
Sewage and Wastewater Treatment Plants	89.6002178337093	5.60001361460683	-56.0001361460683	0.0	-33.600081687641	-67.200163375282	
Waste Storages	89.1502178337093	-27.8594430730342	-55.7188861460683	16.7156658438205	5.57188861460683	-55.7188861460683	
Thermoelectric Biomass Plants	219.690468827907	68.6532715087209	-54.9226172069768	41.1919629052326	13.7306543017442	-96.1145801122093	
Biomass-based Thermal Plant	671.311709509186	209.784909221621	-167.827927377296	125.870945532972	41.9569818443241	-293.698872910269	
Aerial Power Line Supports	1319.00217833709	0.0	-329.750544584273	-82.4376361460683	-164.875272292137	-412.188180730342	
Aerial Power Lines	1319.00217833709	0.0	-577.063453022478	0.0	-164.875272292137	-494.62581687641	
Underground Power Lines	1.31900217833709	0.0	-0.659501089168547	-0.0824376361460683	-0.247312908438205	-0.659501089168547	
Plants for Electricity Transformation	1.31900217833709	0.0	-0.659501089168547	-0.0824376361460683	-0.247312908438205	-0.659501089168547	
Windmill Electrical Generators	20.0	6.25	-6.25	1.25	-1.25	-7.5	
Gas, Oil and Vapor Pipelines	671.316709509186	0.0	-419.572943443241	-41.9572943443241	-125.871883032972	-335.658354754593	
Lighting Systems	91.1502178337093	0.0	-17.0906658438205	0.0	-5.69688861460683	-17.0906658438205	
.							
.							
.							

Figure 5: Excerpt of the content of the file `activities_receptors.csv`

Source	Electrical Energy	Thermal Energy
Thermoelectric Biomass Plants	132.2536622344	0.0
Biomass-based Thermal Plant	0.0	288.66403508895
Windmill Electrical Generators	2.575	0.0
Big Hydroelectric Plants	0.0	0.0
Small Hydroelectric Plants	0.6706451613	0.0
Photovoltaic Plants	41.27058824	0.0
Solar Thermal Panels	0.0	4.84999999875
Superficial Geothermal Plants	0.0	3.2236842105
Thermodynamic Solar Plants	0.5	0.0

`activities_receptors.csv` This table provides the impact of each of the activities on each of the receptors. This table is mainly meant for the environmental expert, that can see the contributions of the activities on the receptors in a plan. Figure 5 shows an excerpt of this file.

`tab_summary.csv` This file provides a table in the same format used in the Regional Energy Plans [10] of the Emilia-Romagna region. An example is shown in Figure 6. The table has two sections, one for electrical energy and one for thermal energy.

For each energy source, it shows the installed power from the last available data (MW), an estimate for the current year, the objective, bot in terms of MW of installed power and in terms of kTOE of energy produced in a year. Finally, it shows the cost, in millions of Euros.

4.3.4 Computing the Pareto frontier (V1): Cost Vs Air Quality

Since there can be different objectives (possibly conflicting) to be pursued, it is interesting to show the Pareto frontier of the different objectives. This section reports the first experiment performed with the multi-criteria optimization, and it is limited to two fixed objectives: the *Cost* and the receptor *Air Quality*. The new version is presented in Section 4.3.5, and it supersedes the version in this section.

Predicate `plot` computes a number of different plans, including:

Electric Energy Production	Current level at 2009 (MW)	Assessment at 2010 (MW)	Overall Objective at 2013 (MW)	Overall Objective at 2013 (kTOE)	Investments (M€)
Big Hydroelectric Plants	0	0	0.0	0.0	0
Small Hydroelectric Plants	297	300	303.0	67.7351612913	25.2
Photovoltaic Plants	95	230	630.0	65.001176478	1400.0
Thermodynamic Solar Plants	0	0	5.0	0.5	22.5
Windmill Electrical Generators	16	20	40.0	5.15	40.0
Thermoelectric Biomass Plants	371	430	649.690468827907	391.1136622344	768.916640897675
Total	779	980	1627.69046882791	529.5000000037	2256.61664089767

Thermal Energy Production	Current level at 2009 (MW)	Assessment at 2010 (MW)	Overall Objective at 2013 (MW)	Overall Objective at 2013 (kTOE)	Investments (M€)
Solar Thermal Panels	25	25	62.5	8.08333333125	90.0
Superficial Geothermal Plants	23	23	28.0	18.0526315788	34.2
Biomass-based Thermal Plant	100	120	791.311709509186	340.26403508895	346.311585519632
Total	148	168	881.811709509186	366.399999999	470.511585519632

Figure 6: Example of the content of the file `tab_summary.csv`, that is in the same format as the table shown in the Regional Energy Plan of the Emilia-Romagna region.

Label	Air Quality	Cost	Plan	Biomass	Photovoltaic	...
0.996094943839125	-483.501132949337	2256.61664089767				
0.996094942915079	-483.501132949337	2256.61664089768				
0.996094942143827	-483.501132949337	2256.61664089767				
0.996095061302185	-483.501132949337	2256.61664089767				
0.992183685302734	-483.501132949337	2256.61664089768				
1	-483.501132949337	2256.61664089767				
0.9921875	-483.501132949337	2256.61664089767				
0.996101379394531	-483.501132949337	2256.61664089767				
0.99169921875	-483.501132949337	2256.61664089767				
0.5	-483.501132949337	2256.61664089767				
0.996094703674316	-483.501132949337	2256.61664089768				
0.992172241210938	-483.501132949337	2256.61664089767				
0.996094942907803	-483.501132949337	2256.61664089768				
9.5367431640625e-7	-358.236967822676	2381.76198973488				
0	190.706586432088	4045.18597636562				
plan	-212.623837504884		3013.9999996715			
Same air quality	-212.623837504884	2823.00293675325				
Same cost	-149.593240727854	3013.9999996715				
Biomass	-897.397559402556			1030.63892811453		
Photovoltaic	746.964832352661				6013.43146472147	
⋮						

Figure 7: Example of the content of file `pareto_ele.csv`.

- the Pareto graph of two objectives, namely the receptor *Air quality* and the cost of the plan;
- the plan proposed by the Region's technicians;
- two plans that lie on the Pareto front, one with the same cost and the other with the same air quality of the plan of the technicians; in other words, two plans that dominate the plan proposed by the Region's experts;
- the *extreme* solutions (that can be used as a comparison) in which only one type of energy source is used.

Predicate `plot` is invoked as `plot`$(Type, Year)$, where Type can take values ele or $therm$, and $Year$ can currently take values 2013 or 2020. The meaning of the parameters is the same as in predicate `plan_amplified`.

The results are saved in one of the files `pareto_ele.csv` or `pareto_term.csv`, depending on the $Type$ of energy requested.

The content of the file can be thought of as consisting of three sections.

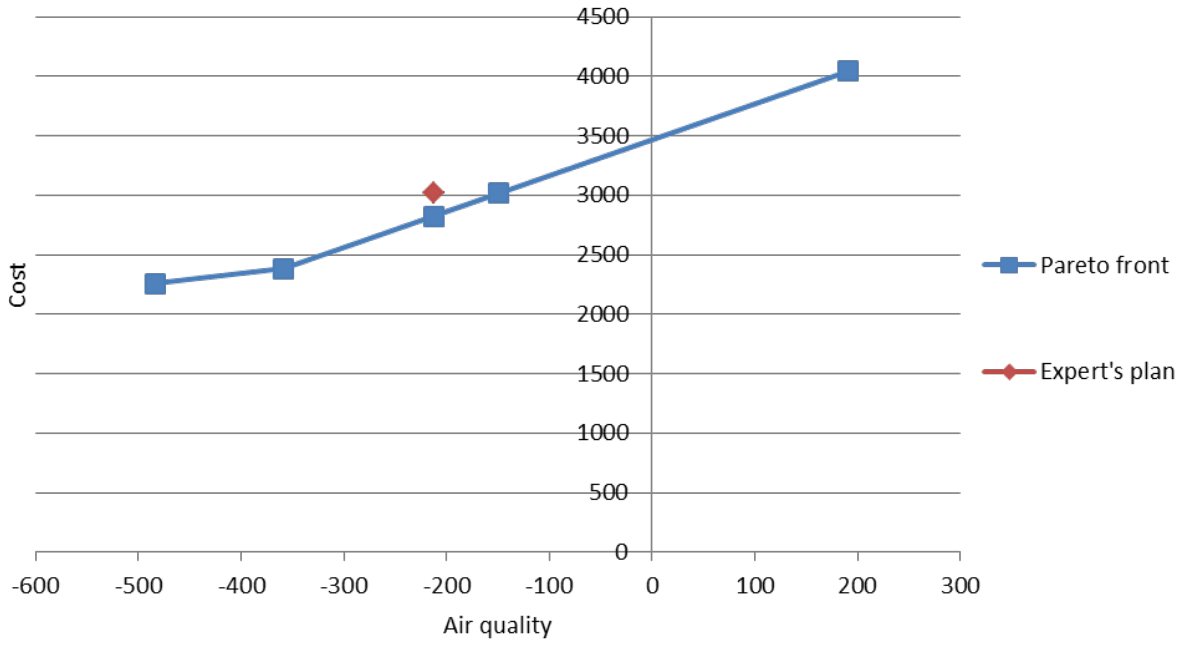


Figure 8: Pareto graph corresponding to the example shown in Figure 7

The first is the Pareto front: for each significant point in the Pareto front we have the two coordinates *air quality* and cost.

The second section contains the comparison of the plan proposed by the technicians with the Pareto front. The third column is the cost of the plan proposed by the experts, while its air quality is reported in the second column. This makes it very simple, with common spreadsheets, to draw a graph like the one in Figure 8, simply by selecting the three columns, and plotting the data as a scatter plot. Note that the plot shows:

- the experts' plan (in red)
- the Pareto front, with highlighted the significant points, where the curve changes its slope
- two plans on the Pareto front that dominate the plan by the experts; more precisely, one has the same air quality as the experts' plan, the other has the same cost.

The third section contains, for each of the envisaged energy sources, a hypothetical plan providing all the required energy through that source. Notice that such plans are provided just for the sake of comparison, and they might not satisfy all the constraints imposed by the user[1]. Note that not all energy sources are reported here; only those sources listed in predicate `sources`$(Type, Sources)$[2]. The data is organized in the file `pareto_ele.csv` as follows. For each of these plans, the air quality is shown in column 2 (labelled air quality), while the cost is in a column containing only one value. Again, this organization allows one to plot the

[1]For example, of course they do not satisfy diversification constraints. Also, one could be interested to plot the impact of a non-renewable source; of course such plan does not satisfy the requirement of having a minimum of energy coming from renewable sources.

[2]This can be useful because some energy sources could be uninteresting for some region, so there is no reason to plot them; on the other hand, the Global Optimizer is designed to be general enough to contain many energy sources.

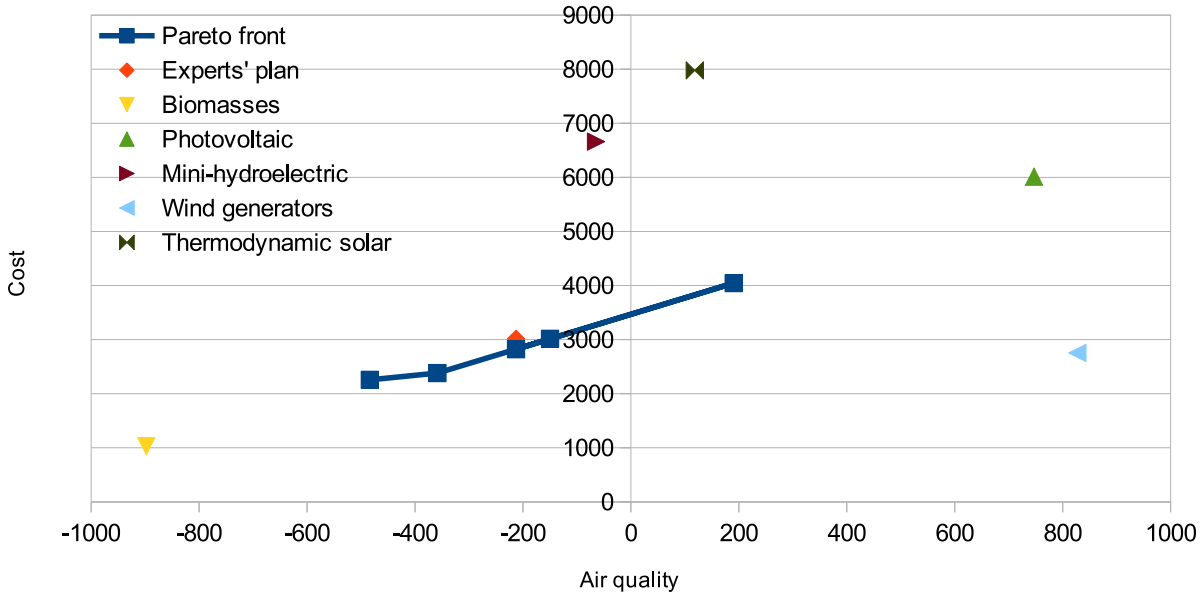


Figure 9: Plot of the single energy sources taken from the example shown in Figure 7

data in a convenient way, by using standard plot types of common spreadsheets; in fact, by selecting that part and choosing a scatter plot, we obtain a graph similar to that in Figure 9. The graph shows in a synoptic way the plans using single energy sources, the Pareto front, and the experts' plan.

For each of these plans, the files described in Section 4.3.3 are saved in a suitable directory. In particular,

- the significant points on the Pareto curve are saved in folders `paretoXX`, where `XX` is a number from 0 to the number of significant points;
- the experts' plan is saved in folder `piano`;
- the plan on the Pareto front with same air quality of the experts' plan is saved in folder `pari_aria`;
- the plan on the Pareto front with same cost of the experts' plan is saved in folder `pari_costo`;
- the plans using only one energy source are saved in a folder named as that source.

4.3.5 General Multi-Objective optimization (V2)

Multi-objective optimization is realized by implementing the *Normalized Normal Constraint* method [8]. By invoking the predicate `nnc` it is possible to compute a number of solutions belonging to the Pareto frontier.

The predicate can be invoked in the following ways:

1. `nnc`$(+ObjectiveList, +NumSolutions)$
2. `nnc`$(+ObjectiveList, +NumSolutions, -NonDomSolutions)$
3. `nnc`$(+ObjectiveList, +NumSolutions, +AdditionalConstraints, -NonDomSolutions)$

where, according to Prolog conventions, the sign '+' stands for an input parameter, while '-' is an output parameter. The parameters are:

ObjectiveList is the list of objectives to optimize. They can be specified in the same way as the objectives for the `compute_plan` predicate (see Section 4.3.2).
For instance, to specify that we are looking for a plan which minimises costs, maximises the air quality and maximises the soil quality, enter the following:

[min(cost),max(rec(9)),max(rec(5))]

NumSolutions is the number of different plans that is desired to obtain as output.

AdditionalConstraints are additional constraints that one wants to be satisfied; this parameter has the same syntax of the $AddConstr$ parameter of the `compute_plan` predicate in Section 4.3.2.

NonDomSolutions is a list containing the computed non-dominated solutions in objective space. Each element of the list represents a non-dominated solution, and is a list containing the values of the various objectives in that solution.

Beside the values provided in the output parameter $NonDomSolutions$, the predicate also prints on standard output the details (in the same way described in Section 4.3.3) of a number of plans.

- The first plans that are provided are the results of the optimization of the objective functions one at a time.
- Then, a number of Pareto solutions of the multi-optimization problem are presented.

Additionally, the file `pareto_multi_obj.csv` is created. It reports the values of the different objectives for the different plans that are computed. The file is in `csv` format (that can be used by most spreadsheet software), and contains one column for each objective in the *ObjectiveList*, plus a first column with a label for each point. The points are classified into *Anchor points* [8], i.e. points that optimize one of the objectives, and *Pareto points*, that are intermediate points. For example, the same Pareto front taken as example in the previous section, and shown in Figure 8 can be computed with the goal

```
plan_amplified(ele,2013),
nnc([max(rec(9)), min(cost)], 17,
  [electric >= 177.2698956357, cost =< 4045000000.0], Sol).
```

The content of file `pareto_multi_obj.csv` is shown in Table 1 and plotted in Figure 10 (this particular plot was obtained using LibreOffice's scatter plot; to show the labels, in LibreOffice from the menu `Insert`, select `Data labels`, then tick `Show category`).

Note that the number of solutions returned in $NonDomSolutions$ could be different from the requested number of solutions $NumSolutions$. This may happen due to several reasons.

One example is when there is only one objective function. In this case, the multi-objective optimization becomes a single objective optimization, and thus there is only one non-dominated solution (which is the optimum for the given objective).

Another example is when the given objectives are not in contrast with each other: in this case the optimum for one objective is also the optimum for the other objectives, so there is only one optimum.

Run	max(rec(9))	min(cost)
	Anchor Points	
max(rec(9))	190.492367677081	4045000000
min(cost)	-483.783908535372	2256616640
	Pareto Points	
1	155.519977914558	3939048027
2	120.547588152037	3833096055
3	85.5751983895147	3727144082
4	50.6028086269928	3621192110
5	15.6304188644707	3515240137
6	-19.3419708980512	3409288165
7	-54.3143606605728	3303336192
⋮	⋮	⋮

Table 1: Excerpt of the file `pareto_multi_obj.csv` for a Pareto front *Air Quality* Vs. *Cost* computed by the general multi-objective optimization (V2)

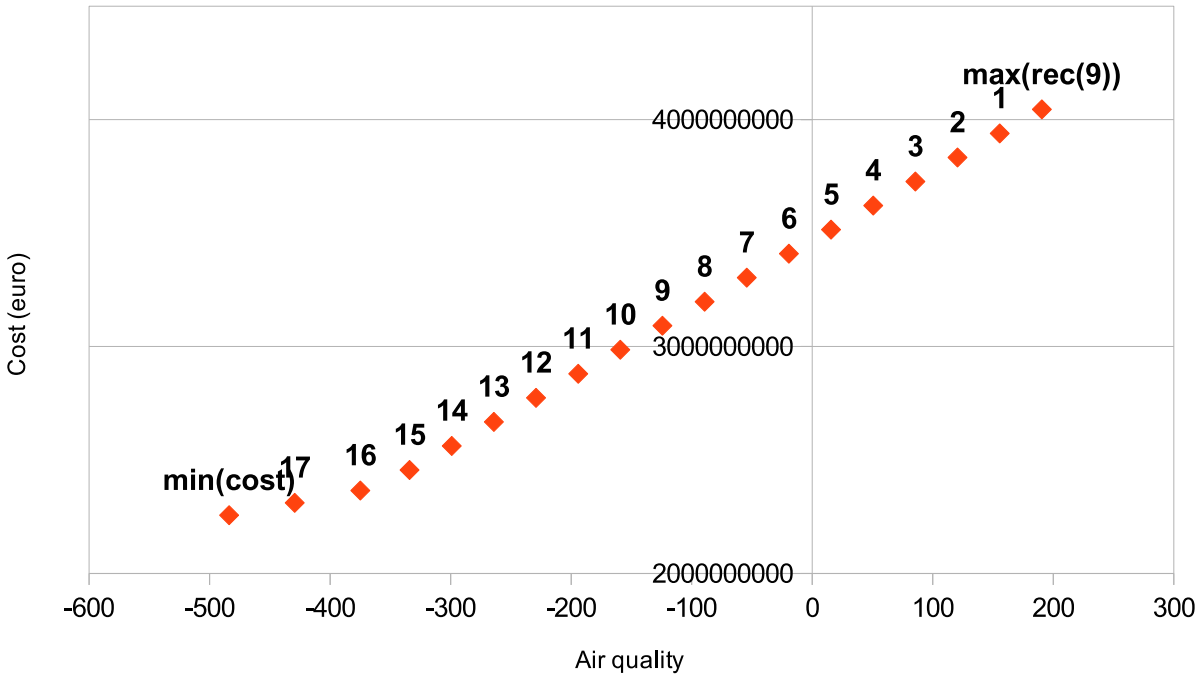


Figure 10: Scatter plot of the Pareto front *Air Quality* Vs. *Cost* computed by the general multi-objective optimization (V2)

Another case is when one of the objective is actually a constant (within the given search space). Suppose we have two objectives: $max(thermal)$ and $min(cost)$. In principle, these two objectives are in contrast, so there should be a series of non-dominated solutions. However, the user imposed a fixed value for thermal energy (for example, because (s)he is interested only in electric energy, or because the devised Region does not have sources of thermal energy), or or by adding in the additional constraints that $thermal = 100$. In this case, the thermal energy is fixed, so there is only one *"real"* objective to optimize. In this case, the list $NonDomSolutions$ will contain only *one* element (as there is only one optimal solution, the optimal value for $min(cost)$). Moreover, this element will contain only one coordinate, because the `nnc` predicate removes the constant objectives from optimization.

In other cases, the number of $NonDomSolutions$ could be less than $NumSolutions$ due to how the Normalized Normal Constraint method [8] works: this algorithm generates a number of candidate solutions, and then removes those solutions that are dominated by other generated solutions.

4.3.6 What if there is no solution?

In some cases, the user might provide some combinations of the input parameters for which there is no solution; in such cases the output of constraint solver is simply a failure, without any explanation of what is the cause of the failure.

In order to provide some hints on the input parameters, we provide a predicate `why_not` that can be used to find solutions.

Predicate `why_not` has three parameters: Budget, electrical energy and thermal energy. Given three values (which can also be variables), it unassigns one of the three parameters, and computes its minimum and maximum possible values, assuming that the other two parameters take the values in input from the user.

For example, the goal `why_not`$(2727128226, 177, 90)$ prints as output:

```
electrical=177,thermal=90, --> cost in 2477534862.33179 .. 4529697710.15843
cost=2727128226,thermal=90, --> electrical in 105.2162334013 .. 219.930058550931
cost=2727128226,electrical=177, --> thermal in 89.77368420925 .. 298.045674763611
```

meaning that if the requested electrical energy is 177kTOE and the thermal energy is 90kTOE, then the cost can range between 2477534862.33179 and 4529697710.15843€.

4.3.7 Configuring the program for other types of plans or other regions

The data used in the global optimizer was provided by ARPA Emilia-Romagna, by ENEA, and by the experts in the Emilia-Romagna region.

The data and the format required by the Global Optimizer is described below. This can be used to tailor the application for other needs, for instance, to include new pressures or receptors, to update the coefficient to include new technologies for producing energy, for preparing a different type of plan, or for any customization for other regions, etc.

The Global Optimizer stores the input data in various files, as Prolog facts. The name of

the files in which each input is defined is listed below. For different configurations of the program it is possible to modify these files.

- **Coaxial matrices.** As explained in Section 3, the Coaxial Matrices link possible Activities with environmental Pressures, and Pressures to environmental Receptors.
  Each Activity, Pressure, and Receptor has a unique identifier, that is given as a Prolog atom. For example, *'wind generators'* could be an activity, *'water consumption'* is a possible pressure, and *'water quality'* is a possible receptor.
  Pressures are declared in the file `press.pl` with a predicate

$$\texttt{pressures}(List\ of\ pressures). \tag{4}$$

  The program expects the pressures to be always in the same order as given in this list. Predicate

$$\texttt{opere\_press}(Activity\ Type, Activity, List\ of\ Impacts\ Activity/Pressure).$$

  defines an Activity, and also its impact on the various pressures. This set of predicates is defined in the file `activity_press.pl`. The *A*ctivity Type is currently not used, though it could be used in future extensions.
  The `List of Impacts Activity/Pressure` has the same length as list *List of pressures* of Eq. (4), and encodes the content of matrix $\mathcal{M}$ described in Section 3. Each element of the list can be a qualitative value with the following code:
  - **a** for High impact
  - **m** for Medium impact
  - **b** for Low impact
  - or it can be a Prolog variable if the activity has no impact on the given pressure.

  Predicate

$$\texttt{ricet\_press}(Receptor, List\_of\_impacts\_Pressure/Receptor).$$

  declares a Receptor, as well as the impact that each pressure has on such receptor. These predicates are defined in the file `rec_press.pl`. The list $List\_of\_impacts\_Pressure/Receptor$ has the same length as the list of pressures (Eq. 4); each element can take one of these values:
  - **a** for a High positive impact
  - **m** for a Medium positive impact
  - **b** for a Low positive impact
  - **l** for a Low negative impact
  - **i** for an Intermediate negative impact
  - **h** for a High negative impact
  - or it can be a Prolog variable if the pressure has no effect on that receptor.

  The qualitative values are converted into numerical coefficients through predicate `coefficient`, that is currently defined as follows:

```
coefficient(X,0):- var(X),!.
coefficient(a,0.75).
```

```
coefficient(m,0.5).
coefficient(b,0.25).
coefficient(h,-0.75).
coefficient(i,-0.5).
coefficient(l,-0.25).
```

This choice makes the addition of more qualitative values very easy, or even the use of quantitative values (it is just a matter of suitably redefining predicate `coefficient`).

- **Primary-secondary activities.** The definition of the relation between primary and secondary activities is declared in the file `activity_settings.pl` through the following predicates.
  Predicate

  $$\texttt{prim\_sec}(Primary, Secondary, Quantity).$$

  defines a positive relation between a $Primary$ and a $Secondary$ activity. Both $Primary$ and $Secondary$ should be activities defined in predicate `opere_press`, and $Quantity$ is a real number providing how much of the Secondary activity is necessary in order to implement one unit of the $Primary$ activity. It is used only when $Primary$ has a positive value.
  Predicate

  $$\texttt{prim\_sec\_neg}(Primary, Secondary, Quantity).$$

  defines a *negative* relation: it means that if we have a decommissioning of 1 unit of the $Primary$ activity (e.g. we decommission a power plant for 1MW) we are building $Quantity$ units of the $Secondary$ activity. It is used only when $Primary$ has a negative value.
- **Outcomes**. Some of the activities provide an *outcome* that is important for the given plan type. For an energy plan, the outcome is the produced energy. Predicate

  $$\texttt{outcome}(List\ of\ outcomes).$$

  provides for each activity the outcome. The $List\ of\ outcomes$ (which is specified in file `activity_settings.pl`) has the same number of elements as the activities, and assumes the values are given in the same order as given by predicate `opere_press`.
  In the Regional Energy Plan of the Emilia-Romagna region, this value is the electrical energy in kTOE, that one unit of the activity can provide in a year. In the Regional Energy Plan of the Emilia-Romagna region, the activities corresponding to power plants are measured in MW of installed power, so the outcome depends mainly on the availability of the energy source during the year (e.g. solar power is available only during the day, so on average the energy produced in a year is lower than a biomass plant having the same installed power).
  For the energy plan, we also have a predicate

  $$\texttt{outcome\_termico}(List\ of\ outcomes).$$

  that provides the same information for thermal energy (and which is specified in file `activity_settings.pl` as well).

- **Costs.** The cost of each activity is declared in the file `cost_activities.pl` by means of the predicate

  $$\texttt{cost\_activity}(Activity, Unit\ manag.\ cost, Manag.\ cost\ measuring\ unit, Initial\ invest.\ cost, Invest.\ cost\ unit, Years).$$

  note, however, that not all the provided data is used in the current implementation, and they are provided in this more general form for possible future developments. In detail,
  - $Activity$ is the activity
  - $Unit\ manag.\ cost$ is the yearly management cost of the activity (per unit of the activity). This data is not currently used.
  - $Manag.\ cost\ measuring\ unit$ is the measuring unit for the $Unit\ manag.\ cost$ (for a power plant, it is usually €/$(MW \cdot year)$). Currently not used.
  - $Initial\ invest.\ cost$ is the cost of installation of the activity, i.e. the initial investment.
  - $Invest.\ cost\ unit$ is the measuring unit for $Initial\ invest.\ cost$. For a power plant, it is usually in €/MW.
  - $Years$ is the expected lifetime of the activity. Currently, it is not used.
- **Current levels.** As we have seen, the tables provided in the regional energy plan (see Figure 6) contain information about the current installed power for each of the energy sources. The current levels are declared in predicates `current_level`$(Activity, Installed)$ and `current_level2010`$(Activity, Installed)$, respectively for the last available data and for the expected to-date level. Both are declared in file `activity_settings.pl`.
- **Renewable sources.** Predicate `renewable`$(Activity)$ declares which activities provide energy from renewable sources. This datum is not currently used, because in the 2013 regional energy plans of the Emilia-Romagna region only renewable sources were considered. However, for future extensions distinguishing renewable from non-renewable sources could be used, to enforce policies requiring a minimum share of energy coming from renewable sources. The file containing these predicates is `activity_settings.pl`.
- **Minimum/maximum shares.** In order to ensure that all the required energy does not come from a single energy source, one can state the minimum and maximum percentage of energy that should be provided for (one or more) sources. This can be done in file `activity_settings.pl` through predicate

  $$\texttt{min\_max\_source\_perc}(Activity, Min, Max).$$

  that declares that the given $Activity$ should not have a magnitude less than Min% (nor more than Max%) of the total of the magnitudes of all the activities for which there exists a fact `min_max_source_perc`. For example, declaring

  $$\texttt{min\_max\_source\_perc}(photovoltaic, 10, 100).$$
  $$\texttt{min\_max\_source\_perc}(wind, 0, 100).$$
  $$\texttt{min\_max\_source\_perc}(nuclear, 0, 100).$$

  one declares that at least 10% of the power provided by the energy sources *photo-voltaic*, *wind* and *nuclear* should come from photo-voltaic (independently from the fact that the system could consider other energy sources as well).

4.3.8 Adding new energy sources

This section recaps the information needed to add new energy sources and it is not strictly necessary, since all the needed input is already listed in Section 4.3.7; however we provide it as a to-do list to simplify the work for those that want to tailor the Global Optimizer for other regions, or for other energy sources.

To add a new activity (in particular, energy sources):

- Add a row to the matrix $\mathcal{M}$ (file `activity_press.pl`), by adding a fact `opere_press`$(Activity, Impacts)$.
- Add a fact `renewable`$(Activity)$, if it is a renewable energy source (the file containing this information is `activity_settings.pl`).
- Add an element to the lists `outcome` and `outcome_termico` (file `activity_settings.pl`), containing the energy (in kTOE) that a plant of 1MW of that source can provide in a year, considering electrical energy (`outcome`) and thermal energy (`outcome_termico`).
- Add the secondary activities in predicates `prim_sec` and `prim_sec_neg` (file `activity_settings.pl`). Note that, in order to be considered primary, an activity must have at least one secondary activity (even if with a zero coefficient).
- Add minimum and maximum allowed values in predicate `min_max_source`, and/or in predicate `min_max_source_perc` (file `activity_settings.pl`).
- Add the current level of installation (file `activity_settings.pl`) in terms of last-known values (predicate `current_level`) and foreseen (predicate `current_level2010`).
- Add the energy source to predicate `sources`$(Type, SourcesList)$, if the source should be plotted as a single source in the graph in Figure 9 (file `activity_settings.pl`).

4.3.9 Emissions

The Global Optimizer can also compute emissions of the energy sources which emit pollutants. In order to do so, the Global Optimizer uses Emission Factors (EF), that have been taken from two data sources: INEMAR [4] and ISPRA [7].

INEMAR Emission Factors are very fine-grained: the Emission Factors are given considering the engine or boiler type, the type of fuel, and the size of the power plant. We consider here only those for biomasses.

ISPRA Emission Factors consider, instead, only the type of main activity (e.g. biomass power plant, oil-based power plant, natural gas power plant, etc.).

In order to choose one, file `modello.ecl` consults one of the two files `emission_inemar.pl` or `emission_ispra.pl`. The files provide the following predicates:

- `emissions(PollutantNamesList)`, which provides the names of the pollutants considered in the given source.
- `unit_emissions(UnitsList)`, which provides the unit of measurement of the Emission Factor per GJ of input (thermal) energy
- `activity_emission(MainActivity, Fuel, Engine/Boiler Type, MinPower, MaxPower, ListEmissions).` where
  - `MainActivity` is the (primary) activity (e.g. biomasses, oil power plant, natural gas power plant, etc.). It is a required data: it is a Prolog ground term.

- `Fuel` is the type of fuel.
- `Engine/Boiler Type` is the type of engine or boiler used in the power plant. In the `emission_XXX.pl` files it is assumed to be a primary key of the database: each type must occur only once.
- `MinPower` and `MaxPower` are the minimum and maximum size of an individual plant of that type, in MW of input (thermal) power
- `ListEmissions` is a list, of the same length of `PollutantNamesList` and `UnitsList`, of the emission factors

Note that in the case of the INEMAR emission factors, the Global Optimizer currently chooses one or more types of plant, fuel, size, etc. In the future, constraints or objective functions could be used to prefer one type to the others.

The emissions are computed considering the formula:

$$Emission(g/year) = P^e(MWe) \cdot EF(g/GJ) \cdot \frac{1MW}{\eta MWe} \cdot \frac{1GW}{1000MW} \cdot \frac{3600s}{1h} \cdot T(h/year)$$

where

- P^e is the output (electrical) power of the plant
- EF is the emission factor
- η is the efficiency of the power plant.
  It is currently considered equal for all types of power plant; its value is declared in predicate `efficiency`, and it is currently fixed to 39% (taken from law prescriptions [2]).
- T is the number of hours the plant is run every year.
  Declared in predicate `hours_per_year` and currently equal to 8000.

4.3.10 Indicators

With the computation of emissions (Section 4.3.9), the Decision Support System (DSS) provides new quantitative information, and lets the user find plans that are optimal with respect to objective functions that include emissions; for example, the user might require the plan that minimizes the emission of NO_X or that of CO_2, or even a weighted sum of the two. However, although useful, these might be too fine-grained for the environmental expert, not to mention for a policy maker: indeed, a policy maker could know that NO_X are toxic for humans, but how does that compare with the emission of heavy metal compounds? Instead, the policy maker knows that CO_2 is not harmful for human health, but it is responsible for the greenhouse effect; are there other emissions that worsen global warming?

The European Commission [5] published a set of indicators quantifying the effect of various substances on *human toxicity*, *global warming* and *acidification*. For example, Annex 1 of the aforementioned document [5] contains 100 chemical substances together with their human toxicity factor, defined as the toxicity of the substance compared to that of lead (Pb). The following annexes contain global warming potentials, relative to CO_2, and acidification potentials, relative to SO_2. By using the weights in the tables, one can provide, e.g. the total human toxicity of a plan (in kg of *equivalent emitted lead*), the total global warming effect of the plan (in kg of *equivalent* CO_2) and the acidification of the plan (in kg of equivalent SO_2).

Moreover, a policy maker may want to optimize on these indicators, and find the plan that minimizes human toxicity, or the greenhouse effect, or any weighted sum of the two.

However, the tables provided by the EC do not always have the same granularity of the information available for emissions. For example, for each plant type we know the emissions of NO_X; unluckily, in [5] we do not have an aggregated value for the toxicity of all the nitrogen oxides, but we have the single toxicity values of NO and NO_2, and they are quite different (respectively, 95 and 300 times that of lead). Even more complicated is for Polycyclic Aromatic Hydrocarbon compounds (PAH), which include many compounds, e.g. Benzo-a-pyrene (toxicity 0.05 times that of Pb) and Naphthalene (500 times Pb). Our environmental expert suggested that we provided as output, for each indicator, three cases: best, worst, and average, considering respectively the highest toxicity in the compound class, the lowest and an average. Instead, when one of the indicators is in the objective function (e.g. one wants to find the plan with minimum human toxicity), we should optimize the worst case to be more conservative.

5 Example

Assuming the ECL^{i}PS^e installation was successful, one can run it either from the command-line or through the graphic front-end `tkeclipse`. In this example outputs are reported in Italian, as they are generated by the Global Optimizer and in accordance with the use case we are considering (the Emilia-Romagna region). The final user interface will support the English version.

From the command-line. If the ECL^{i}PS^e directory is in your PATH environment variable (and that there are no other homonyms applications), then executing `eclipse` should run the ECL^{i}PS^e complier.

Go to the directory where you downloaded the Global Optimizer, then run `eclipse`:

```
$ eclipse
ECLiPSe Constraint Logic Programming System [kernel]
Kernel and basic libraries copyright Cisco Systems, Inc.
and subject to the Cisco-style Mozilla Public Licence 1.1
(see legal/cmpl.txt or http://eclipseclp.org/licence)
Source available at www.sourceforge.org/projects/eclipse-clp
GMP library copyright Free Software Foundation, see legal/lgpl.txt
For other libraries see their individual copyright notices
Version 6.0 #201 (x86_64_linux), Mon Feb 18 22:23 2013
[eclipse 1]:
```

Now, from the ECL^{i}PS^e shell, you can compile the Global Optimizer with `[modello].`

```
[eclipse 1]:  [modello].
source_processor.eco loaded in 0.01 seconds
hash.eco   loaded in 0.00 seconds
compiler_common.eco loaded in 0.01 seconds
```

```
compiler_normalise.eco loaded in 0.00 seconds
compiler_map.eco loaded in 0.00 seconds
compiler_analysis.eco loaded in 0.00 seconds
compiler_peephole.eco loaded in 0.01 seconds
compiler_codegen.eco loaded in 0.00 seconds
compiler_varclass.eco loaded in 0.01 seconds
compiler_indexing.eco loaded in 0.00 seconds
compiler_regassign.eco loaded in 0.01 seconds
asm.eco    loaded in 0.01 seconds
module_options.eco loaded in 0.01 seconds
ecl_compiler.eco loaded in 0.07 seconds
linearize.eco loaded in 0.00 seconds
constraint_pools.eco loaded in 0.00 seconds
loading OSI clpcbc ... done
empty_language.eco loaded in 0.01 seconds
eplex_standalone.eco loaded in 0.04 seconds
eplex.eco  loaded in 0.05 seconds
matrix_util.eco loaded in 0.00 seconds
~/Software/ECLiPSe/lib/listut.pl compiled 17024 bytes in 0.02 seconds
util.eco   loaded in 0.00 seconds
var_name.eco loaded in 0.00 seconds
modello.ecl compiled 604624 bytes in 0.48 seconds

Yes (0.55s cpu)
[eclipse 2]:
```

Now, you can define sensible ranges for activities by running `plan_amplified(2013).`

```
[eclipse 2]: plan_amplified(2013).

Yes (0.00s cpu, solution 1, maybe more) ?
```

Finally, you can run the Global Optimizer with `p(0.5).`

```
[eclipse 3]: p(0.5).
====================== Opere ======================
Opere fognarie: 9.41002178337093km
Impianti adduzione idrica (p.e. acqedotti): 15.4475217833709km
Depuratori e impianti trattamento reflui: 89.6002178337093kab.eq.
Stoccaggio rifiuti: 89.1502178337093kab.eq.
Centrali termoelettriche a biomassa: 219.690468827907MWe
...
====================== Pressioni ======================
1. Trasformazione d'energia da fonti finite: 1189.92779218242
2. Consumo di materiali litoidi: 95.4096260569318
3. Consumo, alterazione di suolo: 1977.79100492174
```

```
4. Consumo di acqua: 1389.34348000493
5. Variaz. consistente di portate idriche: 936.28214607949
...
====================== Ricettori ======================
1. Limitaz.subsidenza e stabilita' falde: 48.7491680507435
2. Stabilita' di versanti e scarpate: -3644.56862849131
3. Stabilita' di litorali o fondali mare: -392.823165376105
4. Stabilita' di rive o alvei fluviali: -1493.21427445758
5. Qualita' pedologica di suoli: -3609.9402259488
...
====================== Obiettivi ======================
Obiettivo: max((1 - 0.5) * ric(9) - 0.5 * costo) = -1363566163.55179
costo(2727128226.41731)
energia(177.2698956357)

Yes (0.17s cpu, solution 1, maybe more) ?
```

To exit from ECL^{i}PS^e, write `halt`.

```
[eclipse 4]: halt.
marco@marco-Latitude-E6530 ~/OptModel $
```

From the the graphic front-end. If the ECL^{i}PS^e directory is in your PATH environment variable (and that there are no other homonyms applications), then executing `tkeclipse` should open a window like the one depicted in Figure 11. Alternatively, if you are running Windows, you can launch `tkeclipse` from your Start menu.

To compile the Global Optimizer open the file menu, browse to the directory where you downloaded the Global Optimizer and choose the file `modello.ecl`.

In the "Output and Error Messages" part of the window you should obtain the following result.

```
source_processor.eco loaded in 0.00 seconds
hash.eco   loaded in 0.00 seconds
compiler_common.eco loaded in 0.01 seconds
compiler_normalise.eco loaded in 0.00 seconds
compiler_map.eco loaded in 0.00 seconds
compiler_analysis.eco loaded in 0.01 seconds
compiler_peephole.eco loaded in 0.00 seconds
compiler_codegen.eco loaded in 0.02 seconds
compiler_varclass.eco loaded in 0.00 seconds
compiler_indexing.eco loaded in 0.01 seconds
compiler_regassign.eco loaded in 0.00 seconds
asm.eco    loaded in 0.01 seconds
module_options.eco loaded in 0.01 seconds
```

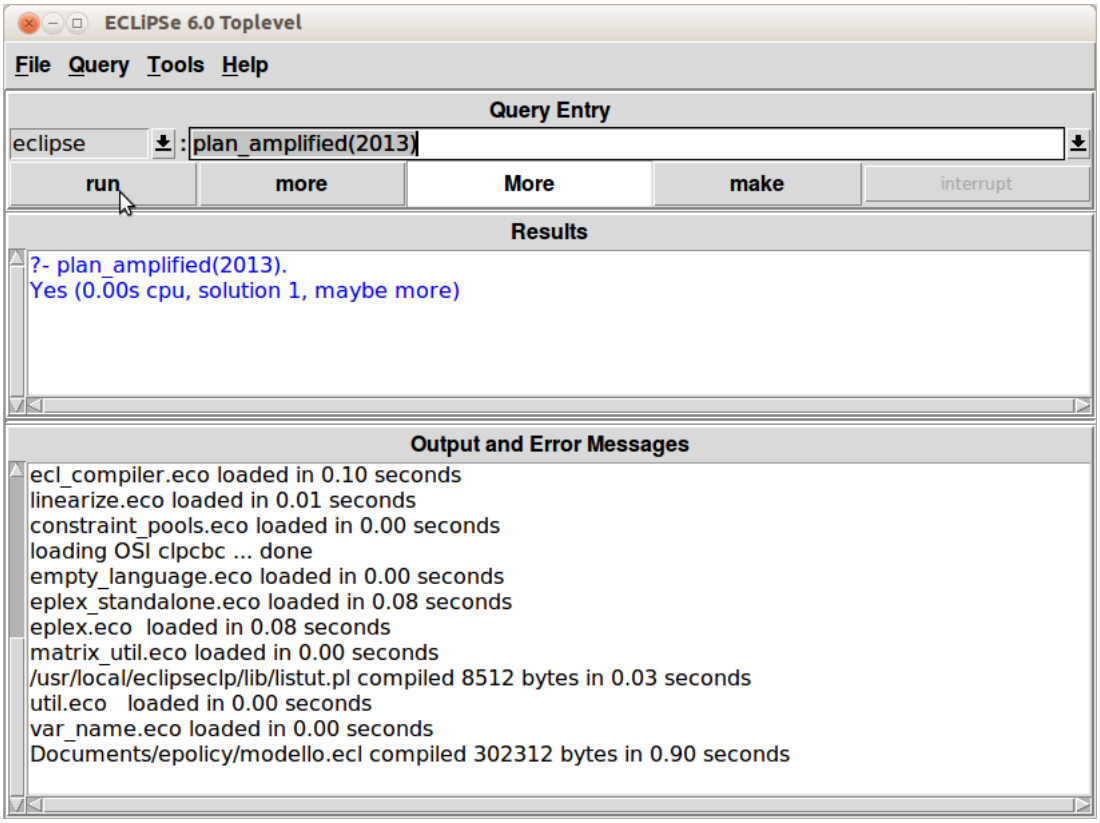


Figure 11: Graphical front-end for running ECL^{i}PS^e.

```
ecl_compiler.eco loaded in 0.07 seconds
linearize.eco loaded in 0.00 seconds
constraint_pools.eco loaded in 0.01 seconds
loading OSI clpcbc ... done
empty_language.eco loaded in 0.00 seconds
eplex_standalone.eco loaded in 0.06 seconds
eplex.eco  loaded in 0.07 seconds
matrix_util.eco loaded in 0.00 seconds
/usr/local/eclipseclp/lib/listut.pl compiled 8512 bytes in 0.03 seconds
util.eco   loaded in 0.00 seconds
var_name.eco loaded in 0.01 seconds
modello.ecl compiled 302312 bytes in 0.74 seconds
```

For defining sensible ranges for the activities by running the amplified plan, type `plan_amplified(2013)` under "Query Entry" and then click on run. In "Results" you should obtain:

```
?- plan_amplified(2013).
Yes (0.00s cpu, solution 1, maybe more)
```

Finally, you can run the Global Optimizer by typing `p(0.5)` in "Query Entry". Then, click on run. A new window like the one in Figure 12 should appear reporting the generated output.

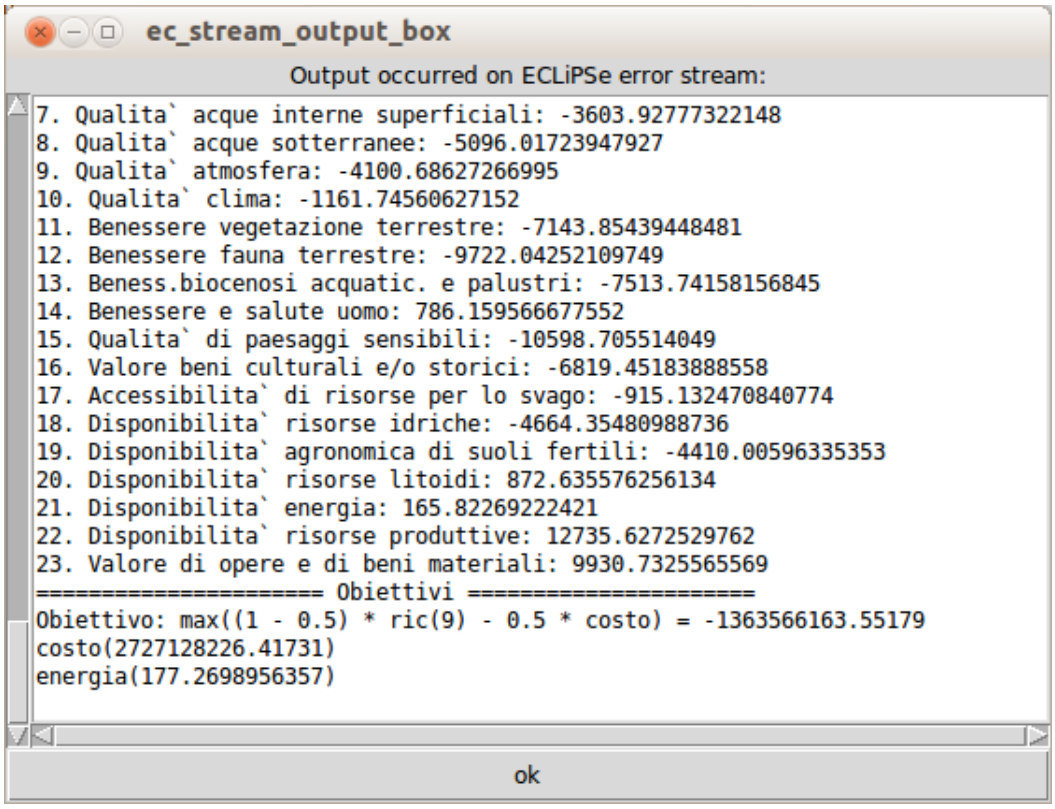


Figure 12: Output window when running the graphical front-end of ECL^{i}PS^e.

6 Global Optimization Module Testing Page

In this section, we describe two testing pages that can be used to experiment with the features of the Global Optimization Module. The services available at the provided links are for **testing** purposes only. Notice that they do not represent the Graphical User Interface (GUI) of the ePolicy prototype. The ePolicy GUI will be based on advanced visualization techniques developed within Work Package 7 (WP7).

6.1 Single objective

The single-objective Global Optimizer can be invoked through the *Global Optimization Module Testing Page* available at

`http://globalopt.epolicy-project.eu/GlobalOptClient/ePolicy.jsp`

The service allows for the generation of a plan by invoking the predicate `compute_plan`, described in Section 4.3.2.

As explained in Section 4.3, one should define limits for the energy sources, then invoke the Global Optimizer. In order to simplify the definition of limits, the test interface provides a button *Set values taken from the Emilia-Romagna use case*. After that, one should define further input (detailed below), and finally *Invoke the Global Optimizer*.

Let us now enter into the details of the expected inputs and of the generated outputs.

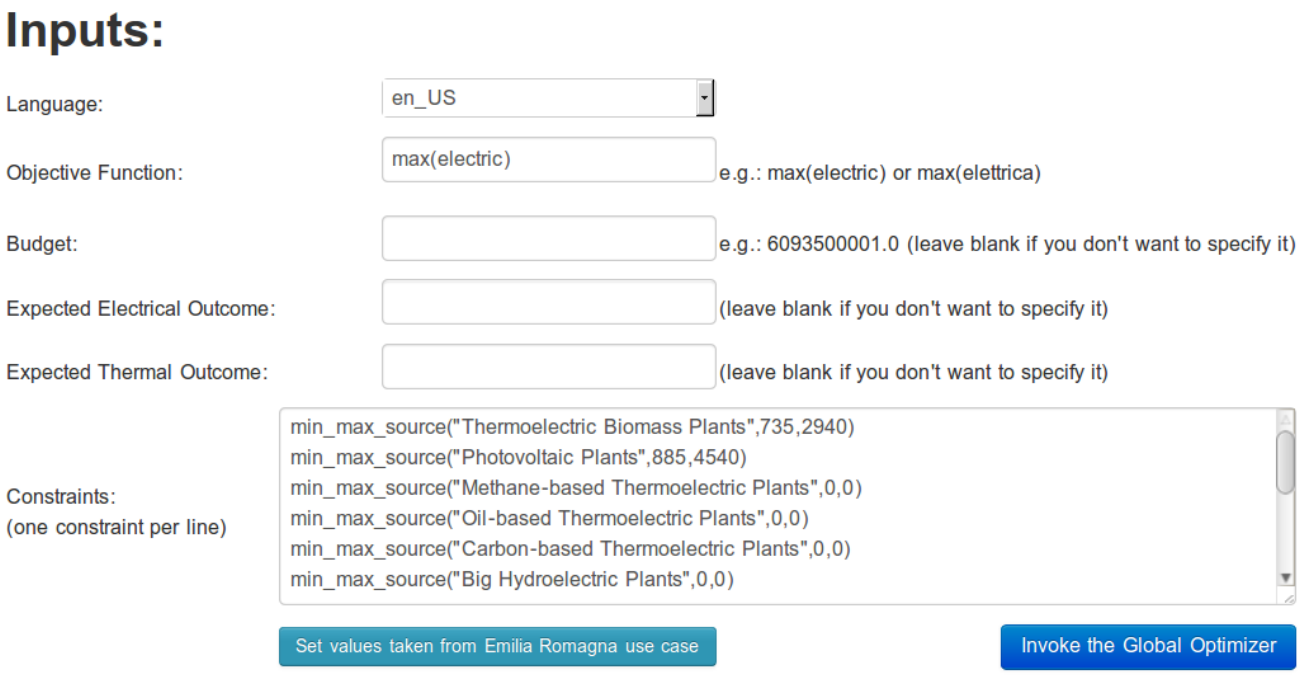


Figure 13: Screenshot of the *Input* section of the Global Optimization Module Testing Page.

6.1.1 Inputs

Figure 13 reports a screenshot of the Input section in the Global Optimization Module Testing Page. Specifically, these are:

Language. the user can specify the desired language. Currently the supported languages are English and Italian. By choosing one of them, inputs are expected to be specified in the selected language. Also outputs will be produced accordingly.

Objective function. This is a **required** input and it follows the same syntax given in Section 4.3.2. Briefly, the *objective function* represents the function that the user wants to optimize and it can be in the form $min(Term)$ or $max(Term)$, where $Term$ is a linear combination of (one or more) of the following: cost (in Italian, *costo*), electric (it: *elettrica*), thermal (it: *termica*), rec(X) (it: $ric(X)$) where X is the index of a receptor.
For example, $max(0.5 * rec(9) - cost + electric)$ is an acceptable objective function.

Budget. This input is optional and represents the budget for the plan, in M€. If specified, it represents an upper limit for the cost of the plan (the total cost of the plan cannot exceed the budget).

Expected Electric Outcome. This represents the total electrical energy that should be produced (in kTOE). This input is optional.

Expected Thermal Outcome. Similar to *Expected Electric Outcome*, but for thermal energy.

Constraints. In this area the user can specify a set of constraints the plan must satisfy (new constraints must be specified on new lines). No syntactical checks are performed on the way constraints are specified. Among the constraints that can be specified there are the limits for the energy sources, which can be given as a set of predicates of the form

$$\texttt{min\_max\_source}(Activity, Min, Max)$$

Alternatively, the limits corresponding to `plan_amplified`*(ele,2020)* can be loaded by pressing the button "Set values taken from the Emilia Romagna use case".

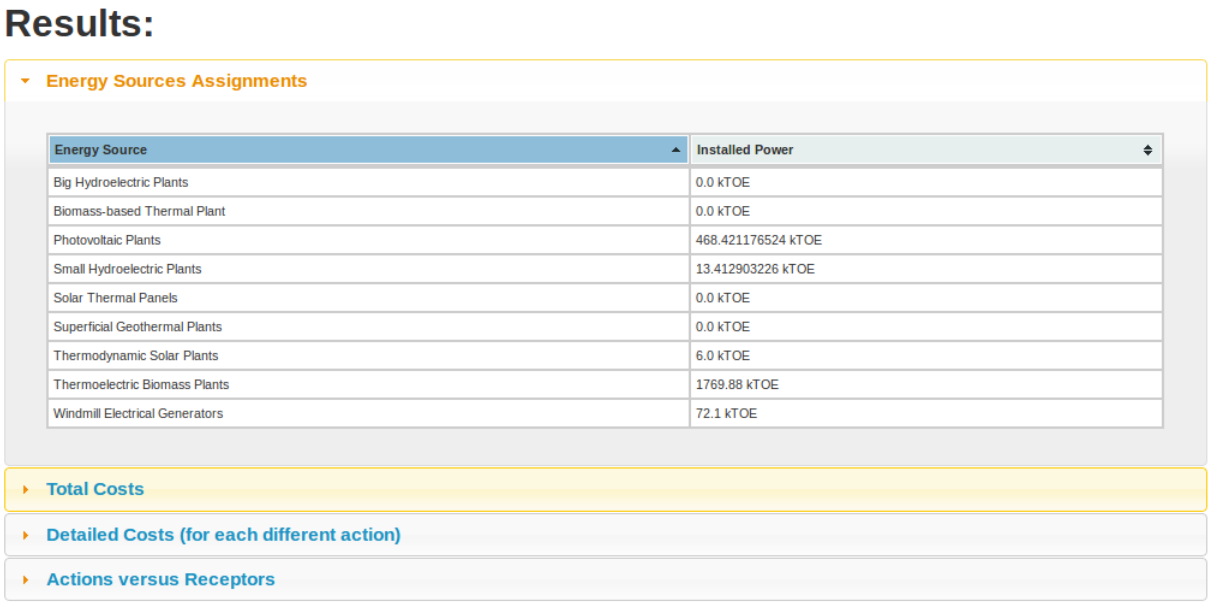


Figure 14: Visualisation of the results in the Global Optimization Module Testing Page.

Once inputs are defined it is possible to compute the plan by pressing the button "Invoke the Global Optimizer".

6.1.2 Outputs

The output produced by the Global Optimizer are organized in four parts described below.

Energy Sources Assignments. Reports the list of Energy sources and, for each of them, the produced energy (in kTOE).

Total Costs. Reports the total costs for primary and secondary activities for producing the quantity of energy from each source reported in the table "Energy Sources Assignments".

Detailed Costs. For each activity the table reports the quantity for that activity and the corresponding cost. At the top of the table it is possible to select the option "hide zero-value quantities" which makes the table more compact.

Actions versus Receptors. This table provides the impact of each activity on each of the receptors. At the top of the table it is possible to select how to visualize the results. Possible alternatives are: (i) showing the values; (ii) coloring cells (using green, red and white respectively for values grater, lower and equal to zero); (iii) showing both colors and values.

Emissions and Pollutants. Provides the total emissions of the power plants.

For each table it is possible to order the rows according to the values of a certain column by clicking on the header of the corresponding column. In this way rows can be ordered in ascending or descending order.

6.2 Testing page for the multi-criteria optimization component

In order to make the optimisation CLP program more user-friendly, we have decided to deploy it as a stateless Web service and access it by means of a stateful Web application. This

is also a convenient choice to ease combining the service with our partners' services and to feed data to the global visualisation interface.

From an architectural and implementation point of view, the CLP program is first embedded inside a Java wrapper (see Figure 15). This component provides a plethora of Java classes that represent the Business Object Model (BOM) of this domain. Any query addressed to this component and all the returned results are expressed in terms of these objects. Ultimately, the purpose of the wrapper is to encode the requests in CLP terms and then to decode the results back. Then we use the Apache CXF framework to define a Web Service's Service Endpoint Interface (WS SEI) – an interface containing the signature of the method to call the service – and later to implement such a service taking advantage of the wrapper.

The Web application that stands as a GUI for the Web service is a standard Java servlet (see Figure 16). Like any Java servlet, it follows the Model-View-Controller (MVC) pattern: any *request* made by users through a browser is intercepted by the servlet which acts a *controller*. The requests are forwarded to the BOM objects inside the *model*; these objects appropriately interact with our Web service and persistence layer to produce some results. The controller then uses the JavaServer Pages (JSPs) – HTML pages with Java tags to incorporate the results – to generate the *view* that becomes the *response* to display in the user's browser. Both the Web service and the Web application are finally deployed to an application server (Apache Tomcat in our case) to make the software available.

The Web application can be accessed at this link:

`http://globalopt.epolicy-project.eu/Pareto/.`

This interface has been implemented as a Web application because of all the advantages that such technological choice involves: the data is centralised and secure, there is no need to install the software, the application is also accessible from smartphones and tablets, it is generally always available and it can truly reach anybody. Moreover the disadvantages arising from the adoption of this delivery platform are not crucial for our domain.

In the following subsections, we present a brief architectural outline of our application to explain how it operates and we describe in detail the User Interface (UI) and how to successfully access the plans optimiser through it.

6.3 Overview of the Architecture

Nowadays, one of the most convenient ways to exploit a software system is to expose it as a Web service. The Service-Oriented Architectures (SOAs), in fact, allow an application to be easily deployed and make it concurrently available to many users, providing infrastructures that are capable of being properly scaled-up with the increase in the complexity of the domain or the number of simultaneous users. Most of all, they provide ways to reuse the code for third-party Web services or other kinds of applications, by means of appropriate choreographies or orchestrations of services.

Among the objectives of the ePolicy project there is the desire to exploit the optimal planner as a stand-alone application as well as part of a more complex system also comprising of the

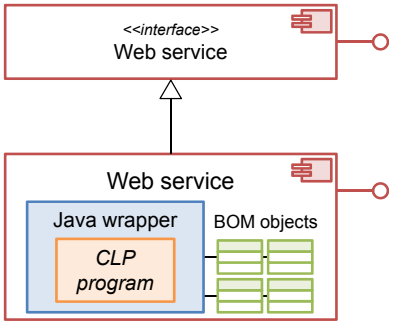


Figure 15: Architecture of the software stack to deploy the CLP optimiser as a Web service.

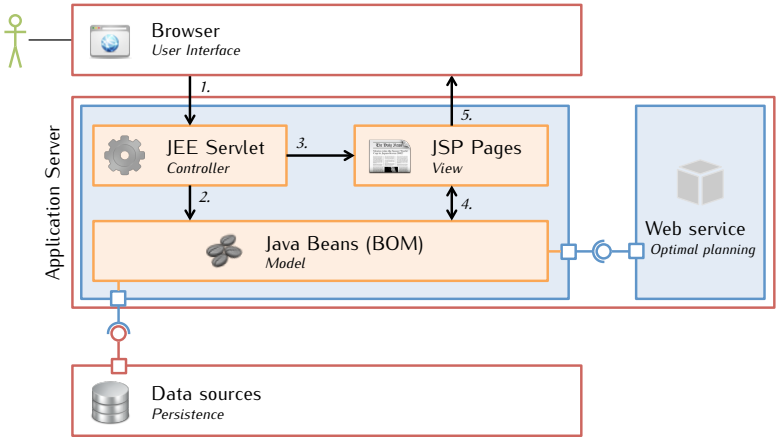


Figure 16: Architectural outline of the Web application to assist the optimal planning of environmental policies.

contributions of our partners. Therefore, the web service approach successfully allows the component to be used in both of these scenarios. So we can focus on the UI when developing the Web application, since the main service is already available.

As you can see in Figure 15, the core of the software stack that implements the Web service is the CLP program itself. The optimal planner is wrapped inside a Java application whose main purpose is simply to execute the Prolog program any time the application is run. In particular, the application starts an instance of ECL^{i}PS^e, passing any input argument and returning the results. Such an operation is accomplished by a plethora of Java classes that are used to represent the BOM, and converted to the proper assertion and prints in Prolog.

6.3.1 Workflow and User Interface

This Web application is designed to serve two kinds of users: the *environmental expert* and the *policy maker*. The former has the task of inputting the geographical and environmental parameters that describe the target for the optimisation of the plan; the latter, however, is expected to provide the specific targets for the optimisation, including amount of energy to be generated, environmental aspects to consider and any constraints on the budget that the optimisation service would then use to return plausible scenarios.

Our initial intention was to dedicate a part of the application to address the needs of the environmental expert and grant access to the other part to both the kinds of users. The part

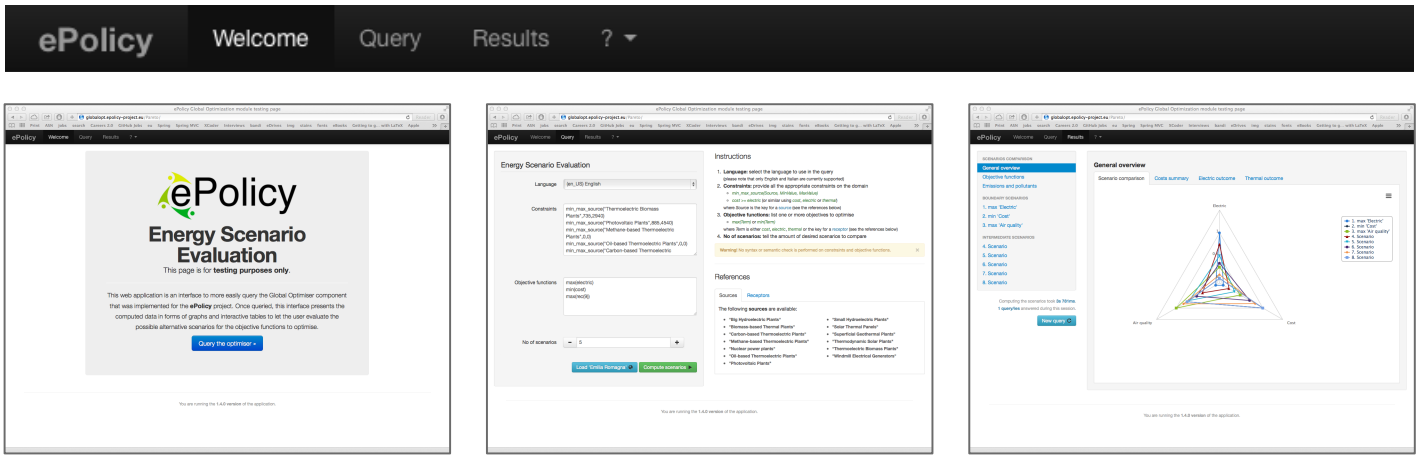


Figure 17: Workflow of the Web application: the navigation toolbar (top) with its three main tasks (*welcome*, *query* and *results*) and their corresponding pages (bottom).

for the environmental expert was planned as an articulated combination of fields by means of which the user could provide and refine a realistic model of the environmental dynamics that take place on the chosen territory. The shared part of the interface, instead, was aimed to ease the querying of the models and allow the graphical presentation of the results. The policy makers would have used it to explore alternative outcomes and decide an effective policy, while the environmental expert would have used it to simply determine whether the model is sufficiently accurate.

Our stakeholders in the environmental domain, however, have pointed out that such an interface would represent a significant deviation from their traditional workflow. In fact, they are used to developing their models in spreadsheets where they can reliably test their data using specific computations. For this reason, we decided to drop the development of the interface for the experts in favour of a simple storage facility into which experts' environmental models can be uploaded.

Once ready, this part will be augmented with a module for the access control and role management (there are frameworks that do so automatically) and a link to upload models will be added for experts.

The workflow that a generic user follows in this Web application is rather simple. It consists only of three main parts: *(i)* a *Welcome* page to introduce the software to the users and guide them to the following page, *(ii)* a *Query* page where the users can request results, and *(iii)* a *Results* page where they can review the results through a neat and simple interface. As shown in Figure 17, the navigation bar on top of the Web application's pages includes a reference to each of these thematic areas. Notice that the currently active page is rendered in a slightly different style to let the users always know where they are in the workflow and how to progress from there.

Since the welcome page is rather trivial, we will only describe the query and the results pages to explain how to user the Web application.

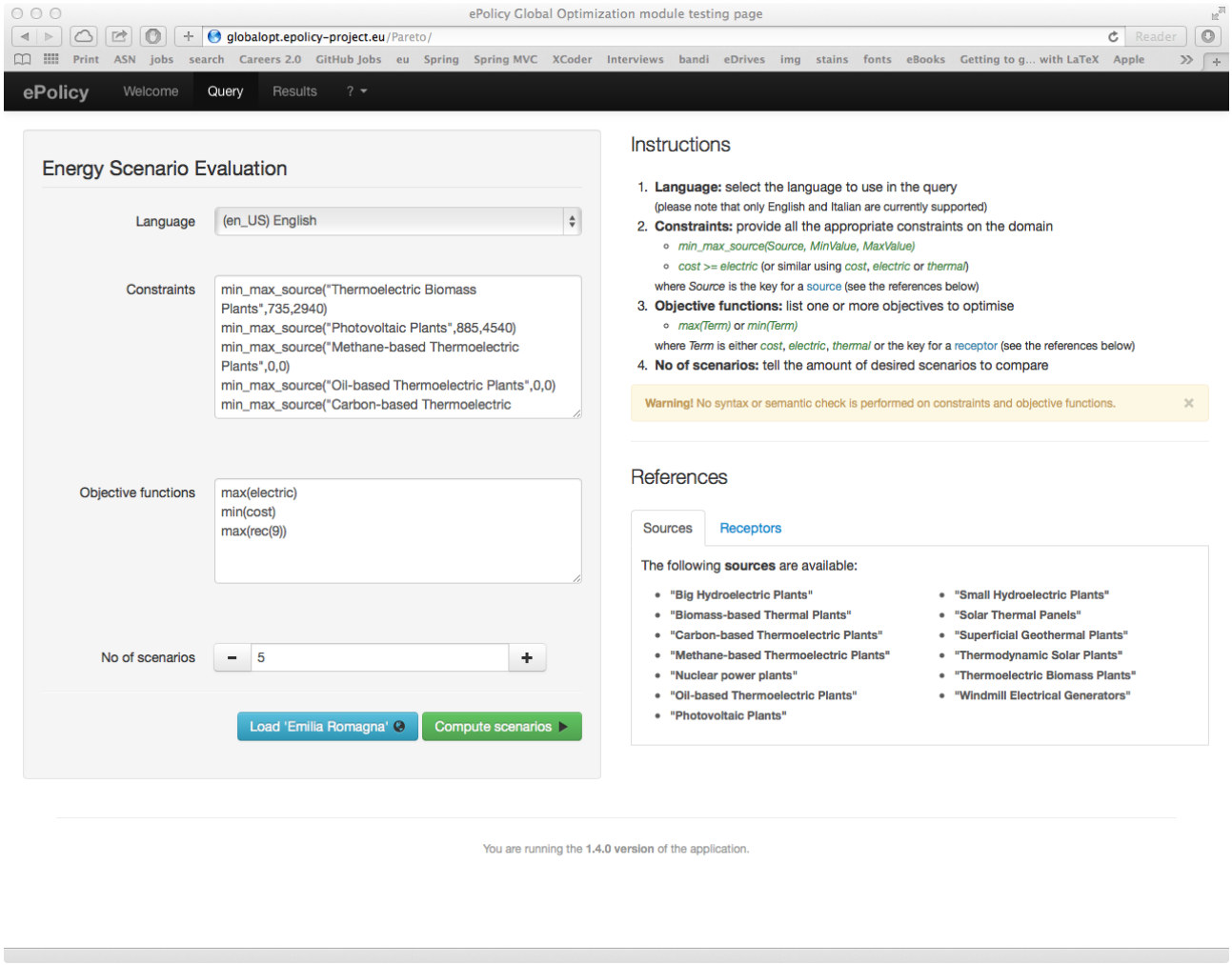


Figure 18: The page of the Web application where the user can provide the additional constraints to compute the desired set of optimal scenarios.

6.3.2 Querying the Web Service

This page is divided into two parts: the panel on the left contains a few input fields while the text on the right instructs the user how to fill them.

The following list introduces all the controls in this page and explains their purpose:

Language. The first control is a *drop-down control* where the user may choose the language in which to express the results. At the moment, English and Italian are supported, to assist both international and local stakeholders and users.

Constraints. This *textarea control* gathers all the additional constraints that the user wants to impose on the environmental model. The generic constraint is expressed in the form:

```
min_max_source( <source>, <min_value>, <max_value> )
```

where `<source>` is the label of an energy source and `<min_value>` and `<max_value>` are respectively the imposed lower and upper bound for that energy source. This control also accepts inequalities in the following form as valid input:

```
<word>  >=  <word>
```

where `<word>` is either `cost`, `electric` or `thermal`.

Objective functions. This *textarea control* contains a reference to the objective functions that the user wants to optimise. It accepts statements in the format:

`max( <term> )` or `min( <term> )`

where the `<term>` is either one of the above words or the label of a receptor.

Number of scenarios. This *integer input control* is used to express how many intermediate scenarios must be considered in the solution.

Load 'Emilia Romagna'. This *button control* fills the above input fields with a default set of constraints, objective functions and values to address our case study on the Electric Regional Energy plan of the Emilia-Romagna region. This data has been used to produce the results shown in Section 6.3.3.

Compute scenario. This *button control* validates the values entered in the input fields. If this is successful, it delegates the Web service to compute the results and the user interface moves on to the *Results* page.

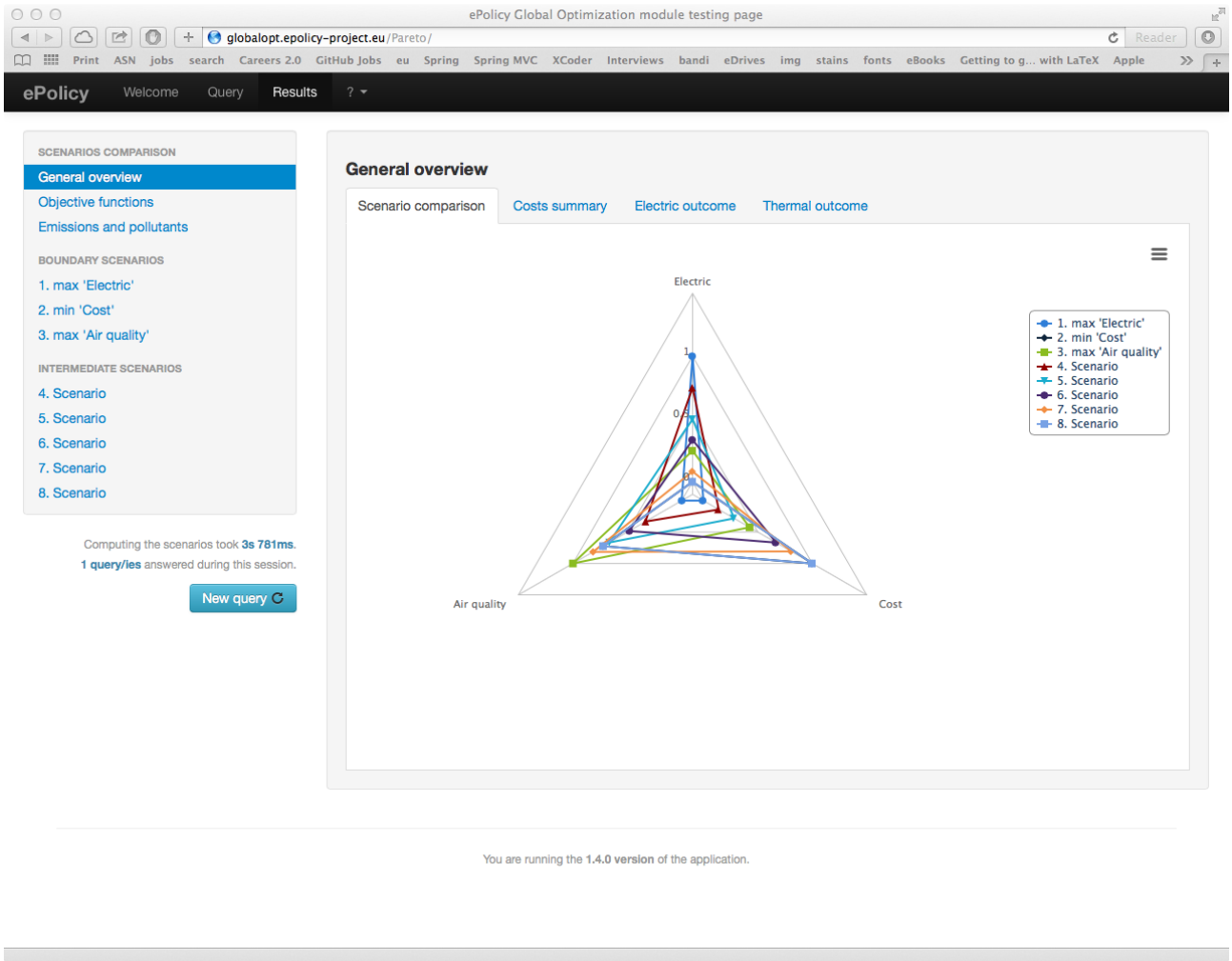


Figure 19: Scenario comparison.

6.3.3 Interpreting the Results

The results show the various scenarios that are computed in the Pareto front. To simplify the interpretation of the results, the scenarios are divided into *Boundary Scenarios* and *Intermediate Scenarios*. Boundary scenarios are those that optimize one of the objective functions; so

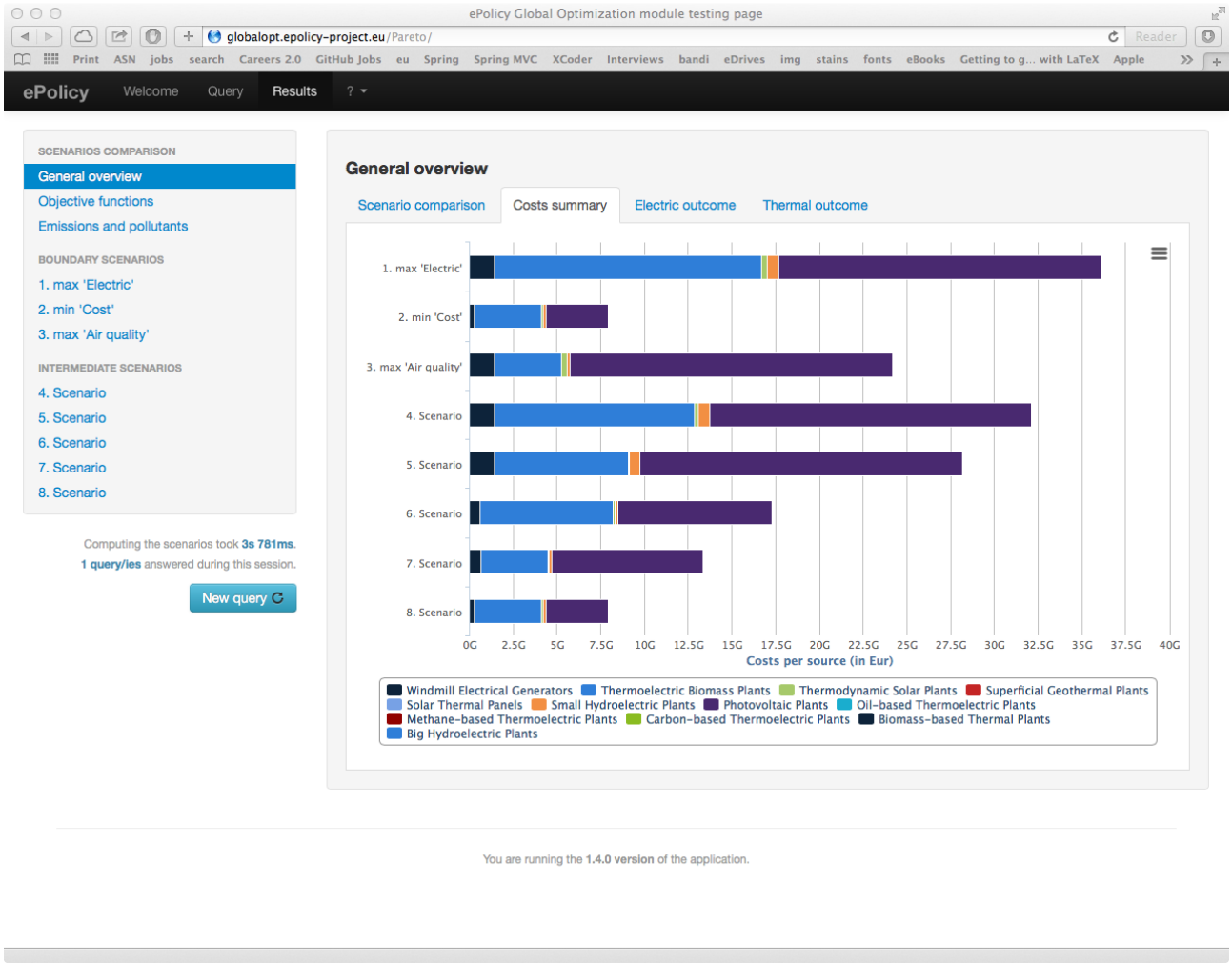


Figure 20: Costs summary.

if we have three objective functions, we will have three boundary scenarios. These scenarios are the “extreme points" that the user can consider; in the example in Figure 10 we have two boundary scenarios: one optimizing the first objective (minimizing the cost) and one optimizing the second (maximizing receptor 9, i.e., the air quality). However, since the two objectives are conflicting, the solution minimizing the cost has a very bad air quality and, vice versa, the scenario with the best air quality has a very high cost. Instead of taking these extreme solutions, the user might be interested in balancing the objectives, and selecting one scenario that has intermediate values for both objectives: these are the *intermediate scenarios*, and are labeled with numbers from 1 to 17 in Figure 10.

The page which presents the results computed by the Web service consists of two *master-slave* panels. The left side-panel is the master and by clicking on any of its entries in one of its three sections, the view in the main panel (the slave) changes accordingly. Each view hosted in the main panel has many tabs and by selecting one of them, an appropriate graph or table is shown.

The following list includes all the graph and tables that are available for each view associated with the entries of the side-panel:

Scenarios comparison. This section provides some entries and views to compare all the gen-

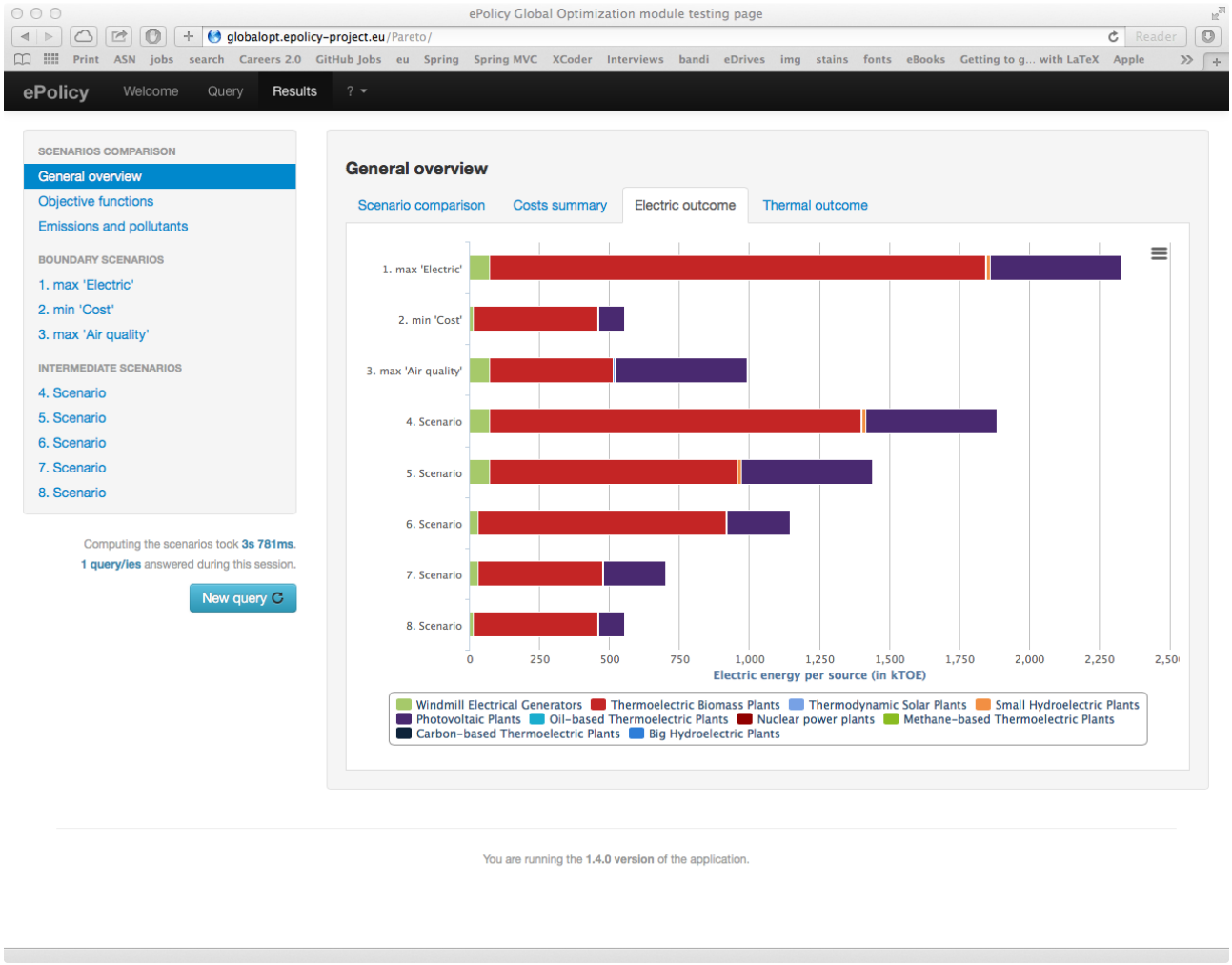


Figure 21: Electric outcome.

erated scenarios with each other. The following entries are available:

a) *General overview.* The view associated with this entry has the following four tabs:
   - ***Scenario comparison.*** The *spiderweb chart* (Figure 19) in this tab has an axis for each objective function. Along each axis, the optimal values are far from the origin. Each scenario is represented by a polygon where each vertex is on a different axis. Generally speaking, a bigger polygon implies a better scenario.
   - ***Costs summary.*** This *stacked bar chart* (Figure 20) has bars representing the costs for each scenario. Each bar is divided into segments that are proportional to the costs per source.
   - ***Electric outcome.*** This *stacked bar chart* (Figure 21) is similar to the above and summarises the amount of electric energy per source, per scenario.
   - ***Thermal outcome.*** This *stacked bar chart* (Figure 22) is similar to the above and summarises the amount of thermal energy per source, per scenario. Notice that it is empty because the default values taken when pressing *load Emilia Romagna* are for the Electric energy plan, so no thermal power plants are included.

b) *Objective functions.* This view has a tab for each objective functions optimized:

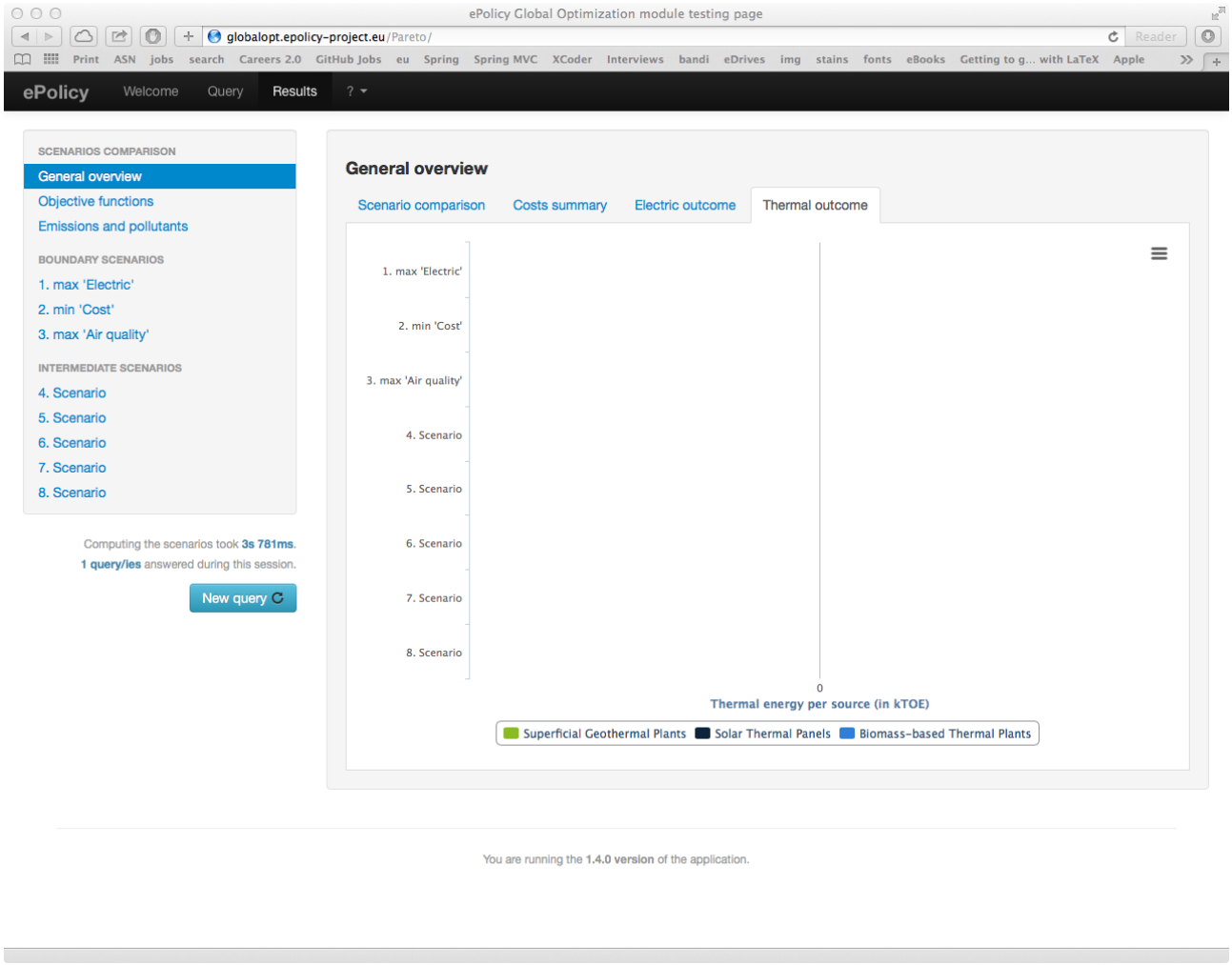


Figure 22: Thermal outcome.

- ***Objective function.*** This *bar chart* compares the value within the current objective function of each scenario. Figure 23 refers to the *Air quality* receptor.

c) *Emissions and pollutants.* This view contains the following three tabs:

- ***Heavy metals.*** This *basic column chart* (Figure 24) shows the amount of this kind of pollutants for each scenario. It is blank because the Emilia-Romagna Regional Energy Plan 2013 included only renewable energy sources, which do not emit heavy metals, according to our data sources.
- ***Greenhouse gases.*** This *basic column chart* (Figure 25) presents the amount of greenhouse gases per scenario.
- ***Other pollutants.*** Similarly, this *basic column chart* (Figure 26) presents the amount of other pollutants per scenario.

Boundary scenarios. This section has an entry for each objective function. Each objective function gives a boundary scenario, the best result for the given function:

a) *Boundary scenario.* For each scenario, the following tabs are available:

- ***Receptors.*** This composite view uses seven *"VU-meter charts"* (Figure 27), showing the receptors in normalized values, i.e. suitably scaled so that each receptor can vary from 0 to 1. Notice that the normalisation requires that the minimum and maximum value for each receptor is computed beforehand.

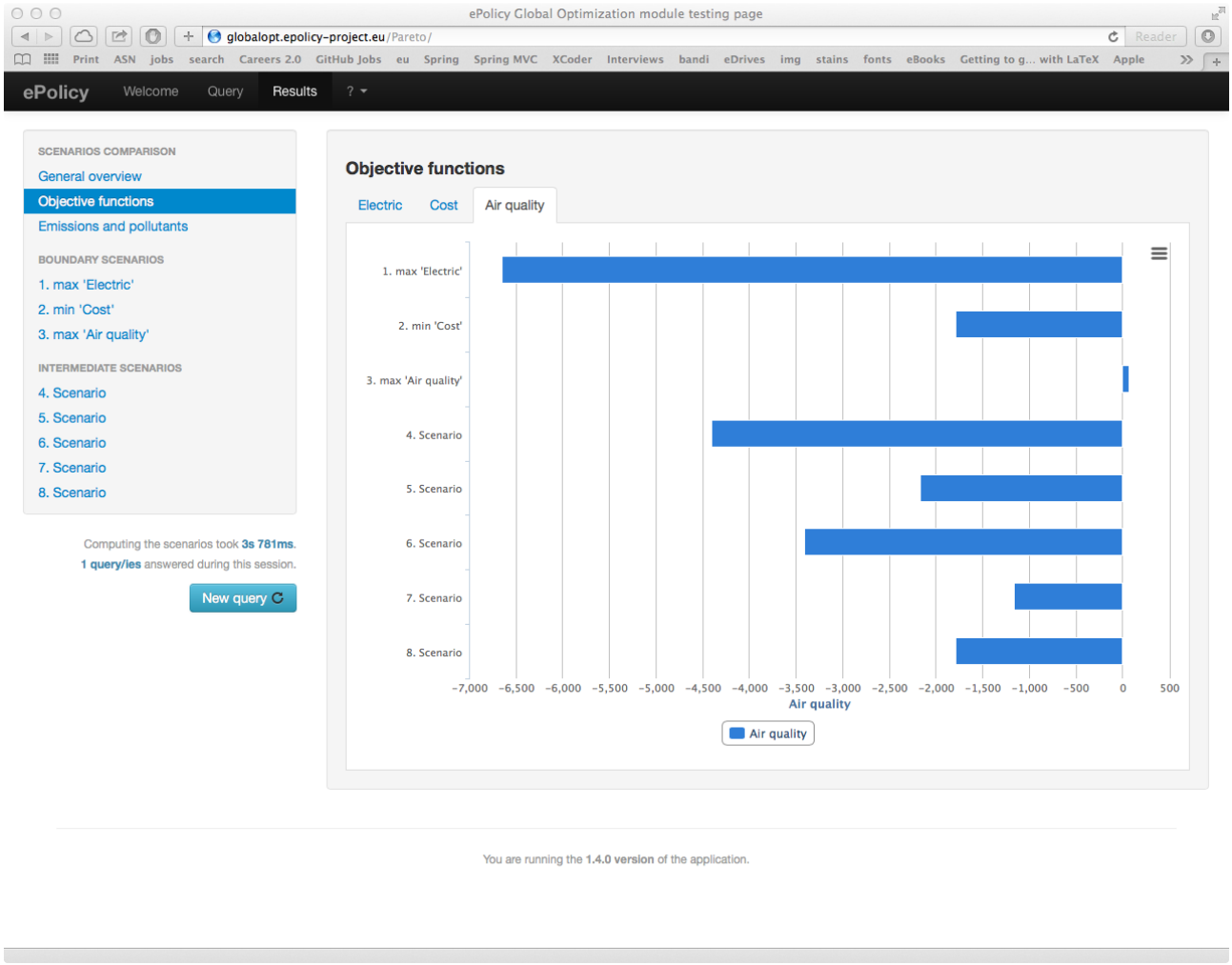


Figure 23: Comparison between scenarios with respect to the *Air quality* receptor.

The top part shows the 3 receptors with the best normalised value, while the bottom one the 3 with the worst normalised value. The main chart allows the user to select any receptor and appraise its normalised value. Notice also that a needle in the green (red) area does not necessarily imply an improvement (worsening) with respect to the given receptor.

- ***Energy sources.*** This interactive *tabular* view (Figure 28) summarises the amount of energy per source produced by this scenario.
- ***Costs per action.*** This interactive *tabular* view (Figure 28) shows the costs per energy source to be spent in primary and secondary works for this scenario.
- ***Detailed costs.*** This interactive *tabular* view (Figure 28) lists the costs per action and amount of goods for this scenario.
- ***Emission and pollutants.*** This interactive *tabular* view (Figure 28) lists the amount of pollutants and emissions of this scenario.

Intermediate scenarios. This section has as many entries as the number of requested scenarios. These scenarios are sampled on the Pareto frontier delimited by the above boundary scenarios as evenly as possible. Each entry exactly includes the same views of a *boundary scenario* (see above).

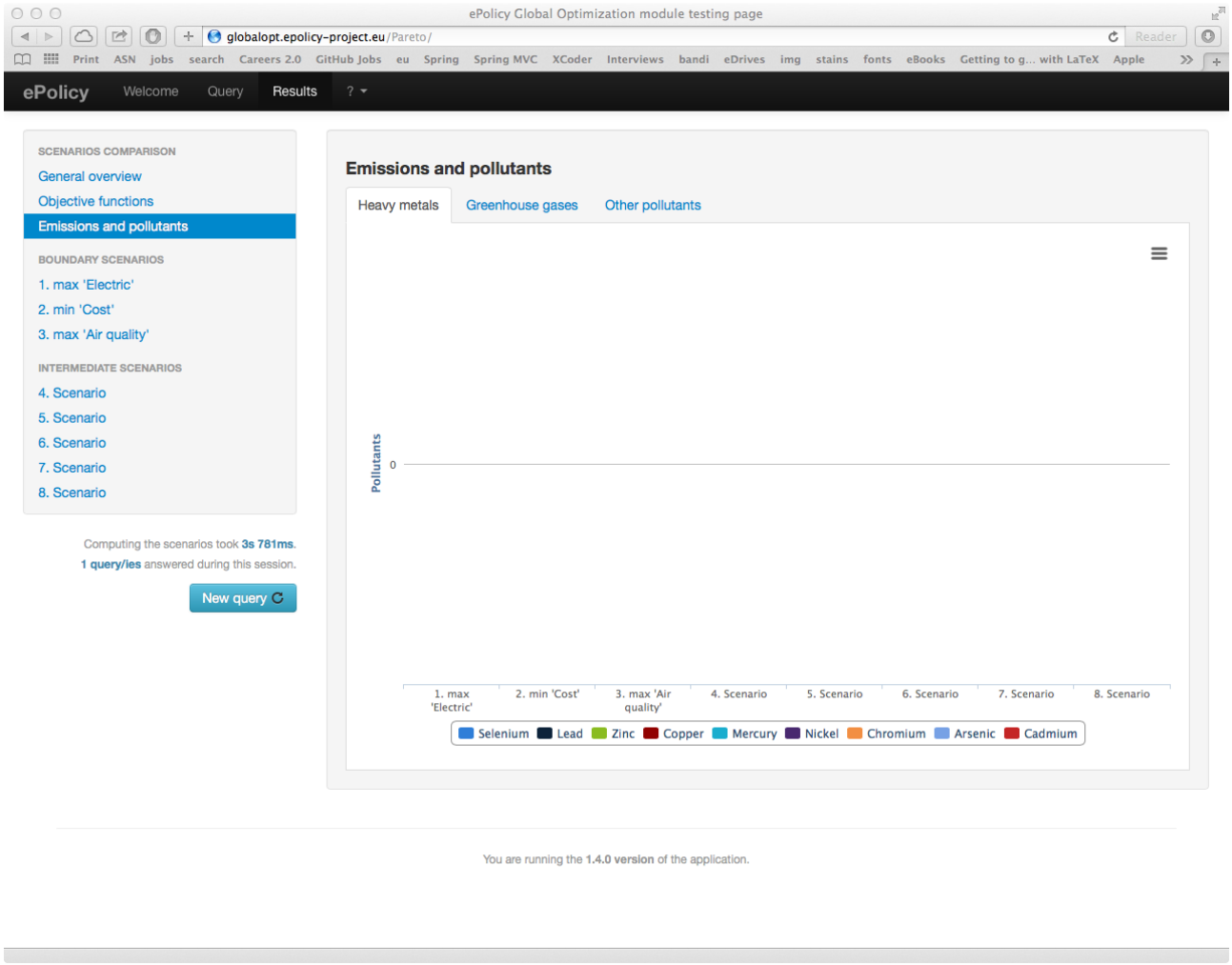


Figure 24: The *Emission and pollutants* views for *Scenarios comparison*: Heavy metals.

7 Conclusions

This document is attached to Deliverable D3.4, which is the prototype of the global level policy reasoning system Version 2. It builds upon Deliverable D3.2, which was the version 1 of the same prototype.

The new prototype provides new features, listed also in the Executive Summary (Section 1), namely:

1. The environmental assessment now is no longer based only on qualitative information, but it also includes the emissions in quantitative form.
2. Emissions are also grouped into the following Indicators: Human Toxicity, Global Warming, and Acidification.
3. The constraint model has been extended to allow the to user compute plans in which the use of some energy sources is reduced. From a computational viewpoint, the model is no longer linear, and it becomes an instance of integer linear programming.
4. A new algorithm was used to implement multi-criteria optimization. While in version V1 multi-criteria was on two fixed objectives (Cost vs. Air Quality), in this version it can be any number of objectives, taken from a wide list.
5. A new test web page (`http://globalopt.epolicy-project.eu/Pareto/`) was developed to test the new multi-criteria optimization.

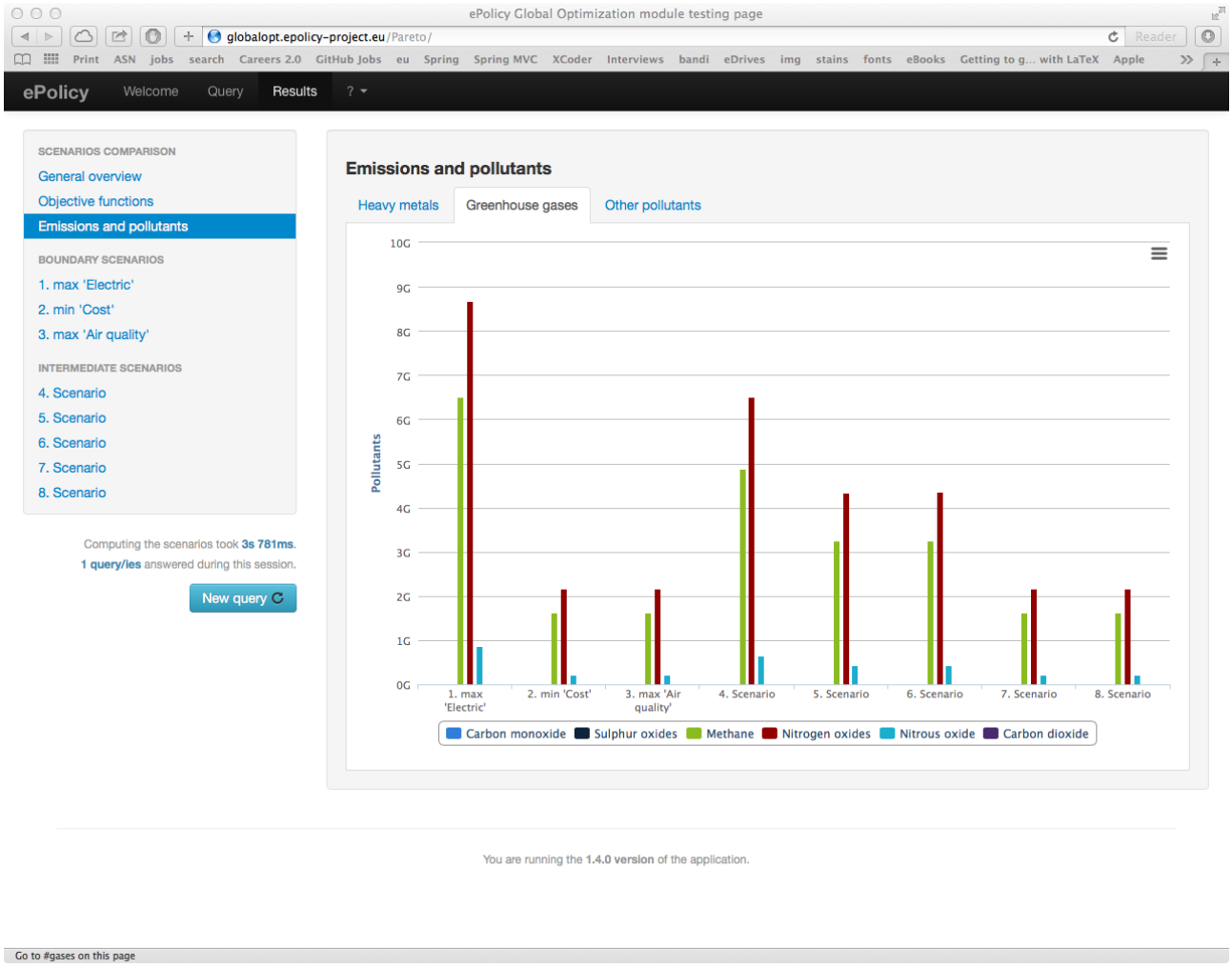


Figure 25: The *Emission and pollutants* views for *Scenarios comparison*: Greenhouse gases.

References

[1] K. Apt and M. Wallace. *Constraint logic programming using ECL^{i}PS^e*. Cambridge University Press, 2007.

[2] Autorità per l'Energia Elettrica e il Gas. Aggiornamento del fattore di conversione dei kWh in tonnellate equivalenti di petrolio connesso al meccanismo dei titoli di efficienza energetica. Gazzetta Ufficiale n. 100 del 29.4.08 - SO n.107, 2008.

[3] Paolo Cagnoli. *VAS valutazione ambientale strategica*. Dario Flaccovio, 2010.

[4] Stefano Caserini, Anna Fraccaroli, Anna Maria Monguzzi, Marco Moretti, Angelo Giudici, and Giovanni Volpi. The INEMAR database: a tool for regional atmospheric emission inventory. In A.E. Rizzoli and A.J. Jakeman, editors, *iEMSs 2002 International Congress: "Integrated Assessment and Decision Support". Proceedings of the 1st biennial meeting of the International Environmental Modelling and Software Society*, Lugano, Switzerland, June 2002.

[5] European Commission. Integrated pollution prevention and control reference document on economics and cross-media effects, July 2006. Available at `http://eippcb.jrc.es/reference/ecm.html`.

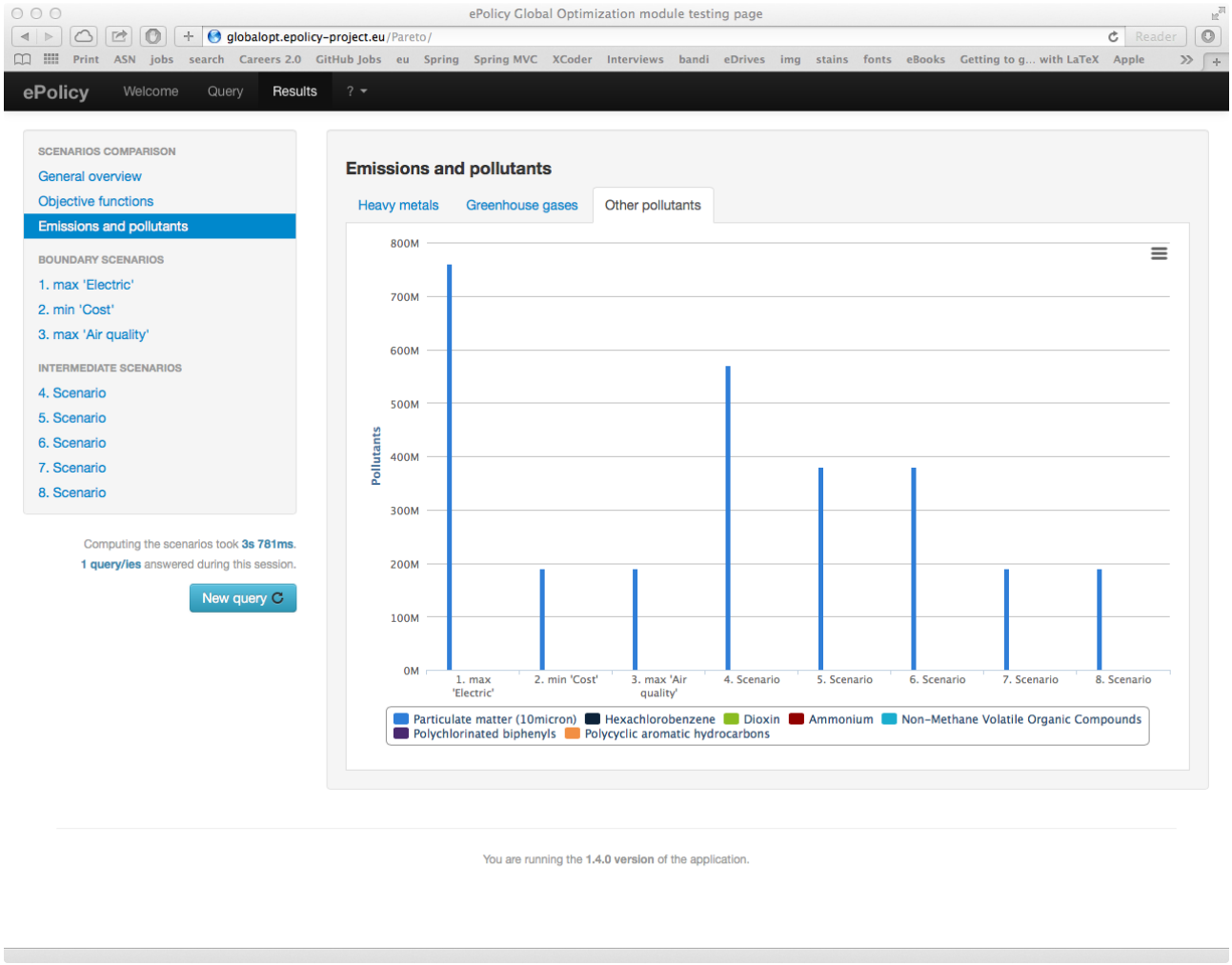


Figure 26: The *Emission and pollutants* views for *Scenarios comparison*: Other pollutants.

[6] Marco Gavanelli, Michela Milano, Federico Chesani, and Elisa Marengo. Prototype of the global level policy reasoning system v1. Deliverable 3.2, ePolicy project, May 2013. `http://epolicy-project.eu/sites/default/files/public/D3.2_0.zip`.

[7] ISPRA. Inventario nazionale delle emissioni in atmosfera. Available at `http://www.sinanet.isprambiente.it/it/sia-ispra/serie-storiche-emissioni/fattori-di-emissione-per-le-sorgenti-di-combustione-stazionarie-in-italia`.

[8] A. Messac, A. Ismail-Yahaya, and C. A. Mattson. The normalized normal constraint method for generating the Pareto frontier. *Structural and Multidisciplinary Optimization*, 25(2):86–98, 2003.

[9] Michela Milano, Attilio Raimondi, Marco Gavanelli, and Fabrizio Riguzzi. Design of the global regional model and its instantiation on the energy plan. Deliverable 3.1, ePolicy project, October 2012. `http://www.epolicy-project.eu/system/files/PUBLIC/deliverables/D3.1.pdf`.

[10] Regione Emilia-Romagna. Il secondo piano triennale di attuazione del piano energetico regionale 2011-2013, July 2011. `http://bur.regione.emilia-romagna.it/dettaglio-inserzione?i=f1fc8c3c7f211ad02579c95ad8dc317a`.

[11] Joachim Schimpf and Kish Shen. ECL^{i}PS^{e} - from LP to CLP. *Theory and Practice of Logic Programming*, 12(1-2):127–156, 2012.

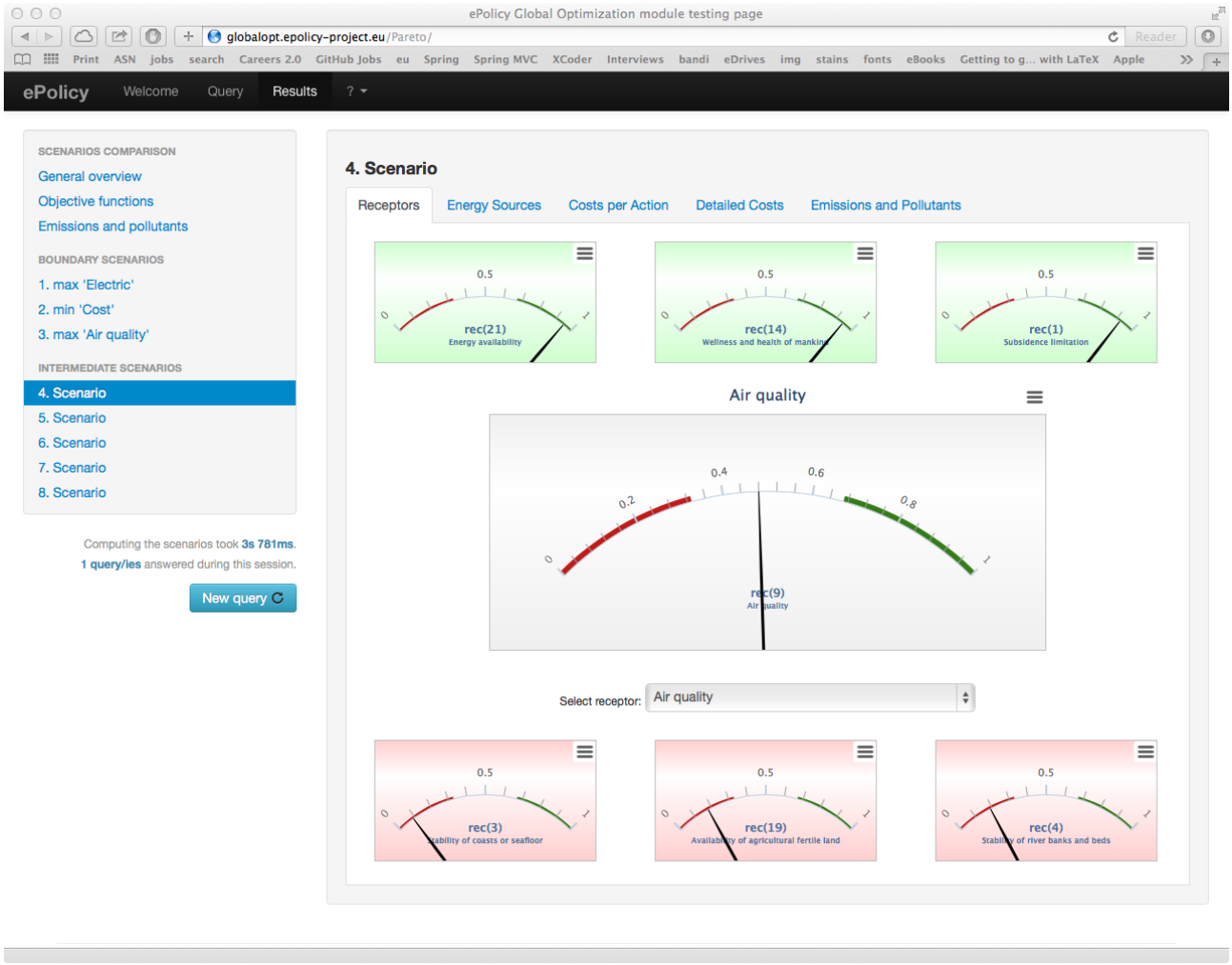
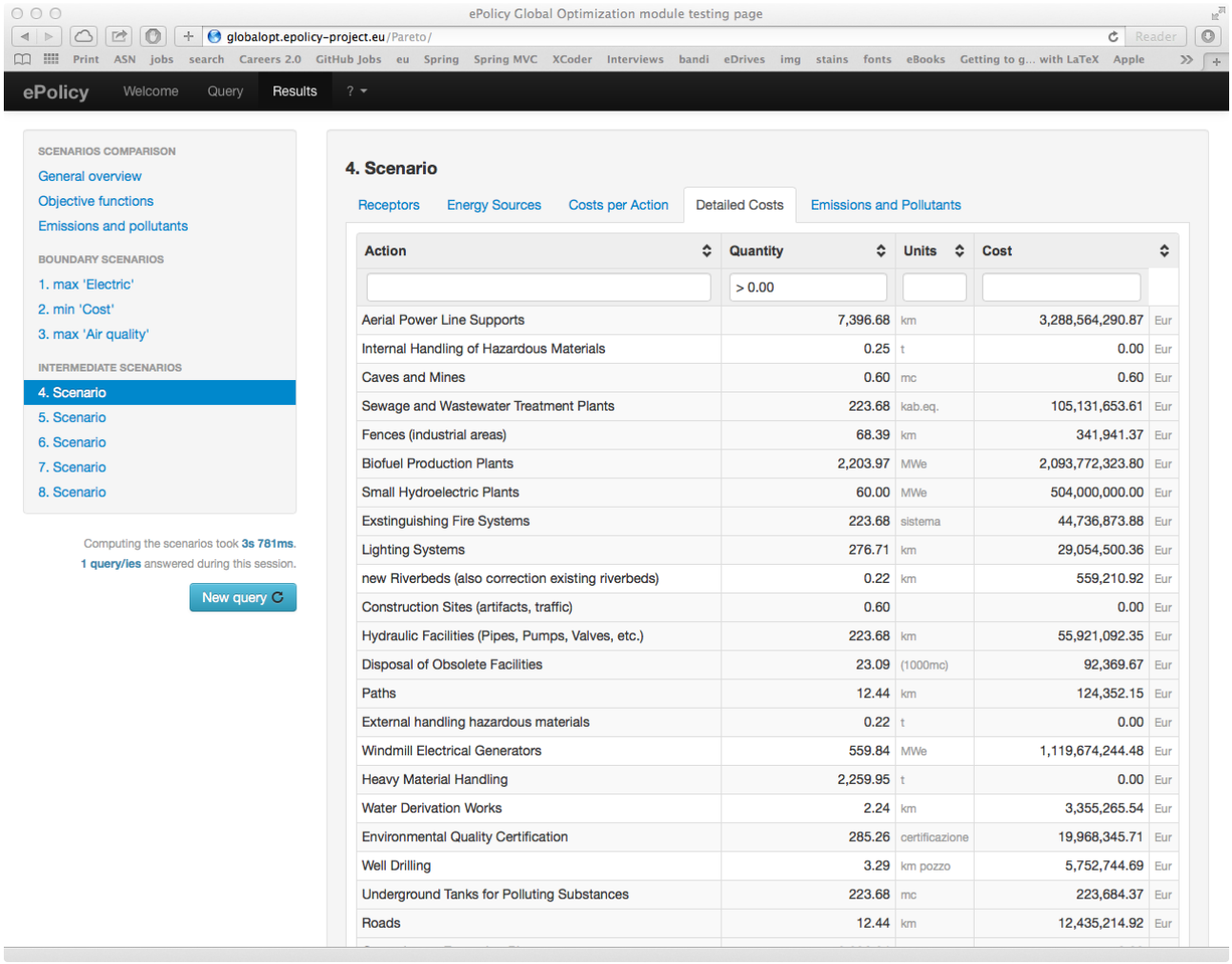


Figure 27: The *Receptors* view associated with each scenario.

[12] Kish Shen and Joachim Schimpf. Eplex: Harnessing mathematical programming solvers for constraint logic programming. In P. van Beek, editor, *Principles and Practice of Constraint Programming - CP 2005*, volume 3709 of *Lecture Notes in Computer Science*, pages 622–636, Berlin/Heidelberg, 2005. Springer-Verlag.

[13] Jens C. Sorensen and Mitchell L. Moss. Procedures and programs to assist in the impact statement process. Technical report, Univ. of California, Berkely, 1973.



Action	Quantity	Units	Cost	
	> 0.00			
Aerial Power Line Supports	7,396.68	km	3,288,564,290.87	Eur
Internal Handling of Hazardous Materials	0.25	t	0.00	Eur
Caves and Mines	0.60	mc	0.60	Eur
Sewage and Wastewater Treatment Plants	223.68	kab.eq.	105,131,653.61	Eur
Fences (industrial areas)	68.39	km	341,941.37	Eur
Biofuel Production Plants	2,203.97	MWe	2,093,772,323.80	Eur
Small Hydroelectric Plants	60.00	MWe	504,000,000.00	Eur
Exstinguishing Fire Systems	223.68	sistema	44,736,873.88	Eur
Lighting Systems	276.71	km	29,054,500.36	Eur
new Riverbeds (also correction existing riverbeds)	0.22	km	559,210.92	Eur
Construction Sites (artifacts, traffic)	0.60		0.00	Eur
Hydraulic Facilities (Pipes, Pumps, Valves, etc.)	223.68	km	55,921,092.35	Eur
Disposal of Obsolete Facilities	23.09	(1000mc)	92,369.67	Eur
Paths	12.44	km	124,352.15	Eur
External handling hazardous materials	0.22	t	0.00	Eur
Windmill Electrical Generators	559.84	MWe	1,119,674,244.48	Eur
Heavy Material Handling	2,259.95	t	0.00	Eur
Water Derivation Works	2.24	km	3,355,265.54	Eur
Environmental Quality Certification	285.26	certificazione	19,968,345.71	Eur
Well Drilling	3.29	km pozzo	5,752,744.69	Eur
Underground Tanks for Polluting Substances	223.68	mc	223,684.37	Eur
Roads	12.44	km	12,435,214.92	Eur

Figure 28: A tabular view associated with each scenario.