Grapheur: A Software Architecture for Reactive and Interactive Optimization

Mauro Brunato and Roberto Battiti

Abstract This paper proposes a flexible software architecture for interactive multi-objective optimization, with a user interface for visualizing the results and facilitating the solution analysis and decision making process.

The architecture is modular, it allows for problem-specific extensions, and it is applicable as a post-processing tool for all optimization schemes with a number of different potential solutions. When the architecture is tightly coupled to a specific problem-solving or optimization method, effective interactive schemes where the final decision maker is in the loop can be developed.

An application to Reactive Search Optimization is presented. Visualization and optimization are connected through user interaction: the user is in the loop and the system rapidly reacts to user inputs, like specifying a focus of analysis, or preferences for exploring and intensifying the search in interesting areas.

The novelty of the visualization approach consists of using recent online graph drawing techniques, with sampling and mental map preserving schemes, in the framework of stochastic local search optimization.

Anecdotal results to demonstrate the effectiveness of the approach are shown for some relevant optimization tasks.

1 Introduction

The development of effective and flexible software architectures for integrating problem-solving and optimization schemes into the overall business organization, modeling, and decision making process is a subject which has been explored for many years. The computational power available even to medium and small businesses and the development of flexible and rapidly deployable optimization schemes, often based on stochastic local search and related heuristics, promise a

Dipartimento di Ingegneria e Scienza dell'Informazione, Università di Trento, via Sommarive 14, I-38100 Trento, Italy.

new progress in the adoption of formal optimization schemes by large and small businesses.

As an example of recent developments, Microsoft introduced the Solver Foundation framework (`http://www.solverfoundation.com/`) to eliminate the divide between "the world of business planning and management by the front office staff and the semi-autonomous world of business IT management by MIS personnel drifting slowly away from the ever-changing and evolving business objectives."

Our architecture is highly modular, it allows for problem-specific extensions, and it is applicable as a post-processing tool for all optimization schemes producing a number of different potential solutions. When the architecture is tightly coupled to a specific problem-solving or optimization method, effective interactive schemes where the final decision maker (DM for short) is in the loop can be developed.

The effectiveness of visualization is well known in the operations research and mathematical programming community [16]. Solving problems involves more than just developing clever algorithms, and giving the optimal solution(s) with minimum f values. Validating and understanding models, algorithms, and data requires appropriate representations. Visualization is crucial to explore optimal solutions, and to summarize the results. After all, the purpose of mathematical programming is *insight*, not numbers [11].

The last years have witnessed impressive developments in the area of stochastic local search [14] techniques for combinatorial optimization and in methods at the boundary between optimization and machine learning, a.k.a. intelligent optimization and Reactive Search Optimization (RSO) [4], autonomous search [13], instance-aware problem-solving [15].

In many cases, the crucial issue is not that of delivering a single solution, but that of critically analyzing *a mass of tentative solutions*, which can easily grow up to thousands or millions. The set of solutions is often characterized by a rich structure which can be mined to extract useful information. A natural structure is caused by the fact that different solutions are chained along a *search trajectory*. A second structural element has to do with the *fitness landscape*: the space of possible configurations is partitioned into attraction basins leading to locally optimal solutions under the local search dynamics. Solutions in different local minima regions often differ by important structural changes: *clustering solutions according to a similarity measure* consistent with the local search neighborhood structure is crucial to navigate among significantly different solutions. One deals with a graph with nodes given by tentative solutions, and edges given by neighborhood relationships along the trajectory, or by similarity relationships of two solutions.

The emphasis on providing a flexible product for the final decision maker is another driver. The final user does not want to be distracted by technical details but he wants to concentrate on using his expertise for and intelligent and informed choice among the large number of possibilities. This can be accomplished if *the user remains in the loop*: he is analyzing preliminary results, and giving feedback to the system which can be used for directing the search to the most relevant and promising regions.

The contribution described in this paper has a clear focus on using state-of-the-art techniques for visualizing stochastic local search, and in particular Reactive Search Optimization results, both at the end of a run, but also while a run is in action, to permit user input while optimizing. The visualization techniques focus on presenting online dynamic graphs, where nodes are candidate solutions and edges (depending on a user-defined threshold) signal similarity, on clustering of solutions (with emphasis on prototypes for the different areas), and on the evolution of the search trajectory.

The intended user is not necessarily an expert in algorithmic details, but a user who needs a fast, responsive, intuitive, mental-map preserving way of navigating among a see of solutions. Abstraction through clustering and the possibility to give feedback to the search process are important ways to keep the user in the loop and encourage him to express in a simple way implicit preferences, not formalized *a priori* in the abstract objective function $\boldsymbol{f}$.

In the following sections, Sec. 2 presents a short review of the multi-objective optimization context and of some existing visualization techniques and Sec. 3 illustrates the software architecture. Sec. 4 explains the used dimensionality-reduction techniques, Sec. 4.2 illustrates the clustering methods, and Sec. 5 discusses the user interaction and dynamic layout.

2 Visualizing multi-objective optimization problems and reactive search optimization

Reactive Search Optimization (RSO) [4] advocates *learning for optimizing*, by inserting a machine learning component into a solution process so that algorithm selection, adaptation, integration, are done in an automated way, and a comprehensive solution is delivered to the final user. The diversity of tasks and the dynamicity which are intrinsic in the real world can be dealt with in an effective manner. The interaction with the final user is simplified and made human: no complex technical questions are asked about parameters but the focus is kept on the specific instance and user preferences. In fact, the user wants to maintain control of the problem definition, including of course hard and soft constraints, preferences, weights. This is the part which cannot be automated, while he is happy to delegate issues related to algorithm choices and tuning.

A first application of the above ideas is studied in the traditional context of multi-objective optimization problem (MOOPs). The incomplete knowledge about the problem to be solved is formalized by assuming the knowledge of a set of desirable objectives, and ignorance of their detailed combination.

In detail, a MOOP can be stated as:

$$\text{maximize } \boldsymbol{f}(\boldsymbol{x}) = \{f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})\} \tag{1}$$

$$\text{subject to } \quad \boldsymbol{x} \in \Omega \tag{2}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is a vector of n decision variables; $\Omega \subset \mathbb{R}^n$ is the *feasible region* and is typically specified as a set of constraints on the decision variables; $\boldsymbol{f} : \Omega \to \mathbb{R}^m$ is made of m objective functions which need to be jointly maximized. Objective vectors are images of decision vectors and can be written as $\boldsymbol{z} = \boldsymbol{f}(\boldsymbol{x}) = \{f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})\}$. The problem is ill-posed whenever objective functions are conflicting, a situation which typically occurs in the real world. In these cases, an objective vector is considered optimal if none of its components can be improved without worsening at least one of the others. An objective vector $\boldsymbol{z}$ is said to *dominate* $\boldsymbol{z}'$, denoted as $\boldsymbol{z} \succ \boldsymbol{z}'$, if $z_k \geq z'_k$ for all k and there exists at least one h such that $z_h > z'_h$. A point $\hat{\boldsymbol{x}}$ is Pareto optimal if there is no other $\boldsymbol{x} \in \Omega$ such that $\boldsymbol{x}$ dominates $\hat{\boldsymbol{x}}$. The set of Pareto optimal points is called *Pareto set* (PS). The corresponding set of Pareto optimal objective vectors is called *Pareto front* (PF).

In particular, we focus on *interactive* multi-objective optimization. According to [20], through interactive MOO the DM is building a conviction of what is possible and confronting this knowledge with her preferences that also evolve.

The assumptions about knowledge and ignorance of the problem to be solved can be of course generalized. For example, in many cases the decision maker wants to actually *see* the proposed solution (not only the objective values) to decide. This is a clear evidence that objectives are not the entire story and additional opportunities for learning the real preferences exist. A recent proposal of an evolutionary multi-objective optimization algorithm adapting to the decision maker is presented in [5].

In this work, after a brief presentation of the software architecture, we focus on the interactive visualization component. Visualization is here investigated to provide a flexible decision support environment. Crucial decisions depend on factors and priorities which are not always easy to describe *before* starting the solution process. Feedback from the user in the preliminary exploration phase can be incorporated so that a better tuning of the final solutions takes the DM preferences into account. When solving complex real-world problems many criteria are present, some explicit in the current modeling through f, but some implicit in the decision-maker mind and experience. Therefore the user must be kept in the loop for subsequent refinements, and the need arises to present candidate solutions and provide *on demand* detailed focus on selected portions of the search space.

Recent developments in the literature of visualization of graphs and networks consider issues of scalability required when dealing with very large graphs. For example, *sampling with a bias* according to a specific focus and aiming at preserving some relevant graph properties is considered in [22]. The fast drawing of *online dynamic graphs* is studied in [9]. The issue is of particular relevance when the user is in the loop while an optimization engine is at work: if the graph layout is changing rapidly the focus of attention may be lost unless techniques for preserving a mental map are used.

Previous work in the area of visualization for optimization includes [21] which discusses the visualization of evolutionary algorithms also through multidimensional scaling, and [1] which deals with human-guided simple search, combining information visualization and heuristic search. In their approach, the computer is responsible only for finding local minima by simple hill-climbing and the user iden-

tifies promising regions of the search space, and intervenes to help it escape non-optimal local minima.

A case study of visualization and optimization for a business strategy is considered in [12]. The visualization capability implicitly allows the user to better formulate the objective function for large optimization runs. As the perceived benefit of different choices does not have an a priori mathematical formulation, the user's intuition and pattern recognition skills are brought into the solution, while simultaneously taking advantage of the strength of algorithmic approaches to quickly and accurately find an optimal solution to a well-defined problem.

N-to-2-space mapping for visualization of search algorithm performance by using space-filling curves is discussed in [17]. The main purpose is to discover weaknesses and tune an algorithm. Visualization of Pareto-Sets in Evolutionary Multi-Objective Optimization is investigated in [19]: finding a mapping which maintains most of the dominance relationships is itself a challenging multi-objective optimization task.

After a presentation of the overall software architecture, we will concentrate on the basic issues to be addressed by appropriate visualization techniques: dimensionality reduction, focussed reduction in the number of nodes and edges, dynamic visualization and exploration.

3 The software architecture

Interfacing a generic optimization algorithm with a user who needs to take decisions is a delicate issue that must consider many nuances of both parties. While optimization systems operate at their best with well-defined, deterministic objectives and can generate a plethora of different solutions on arbitrarily high-dimensional spaces, a decision maker is pursuing conflicting goals with stochastic outcomes, tradeoff policies that rely on his intuition, and is able to compare small sets of solutions represented on a two- or at most three-dimensional graph.

A recent survey on the topic of multi-objective decision making [6] identifies a number of research and application challenges related to this context: (1) uncertainty in the input and in the evaluation of criteria, (2) high dimensionality of the search and objective spaces, (3) not well ordered preference schemes, (4) the representation and use of domain knowledge, (5) the existence of throughput constraints in the process, and (6) the possible distribution of the process to multiple interconnected computers.

The architecture we propose, called *Grapheur*, is shown in Fig. 1. It is based on a three-tier model that embeds most of these issues by identifying the core functionalities related to the interface between the optimization algorithms (bottom) and the user (top). The architecture is independent from the optimization package in use, as long as an appropriate stub can be designed in order to interact with it. These stubs, which can be a data filter in the simpler off-line case, are collected within the back-

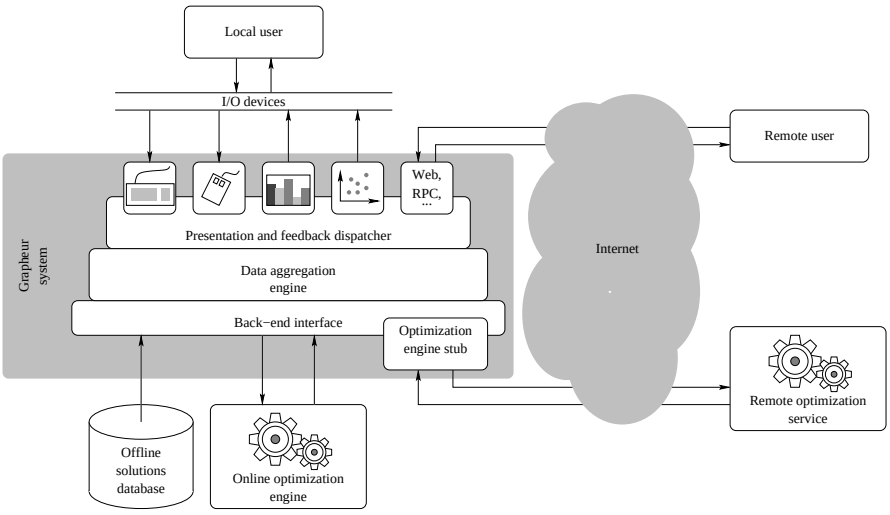

Fig. 1 The *Grapheur* system architecture. The core component is the Data aggregation engine, which handles data and user feedback in a consistent way.

end interface. In this regard, we identify two main classes of optimization services (bottom of Fig. 1):

- Off-line algorithms that do not interact with the user, either because they lack an interface or because they provide a complete enumeration of solutions, or at least a representative set covering the space of possible solutions. Historic or *a posteriori* data analysis also fall in this category. Data produced by these algorithms is treated as a database which needs to be browsed by the user in a convenient way.
- Online algorithms that can be driven in their optimization process either by online commands that dynamically modify their behavior, or by re-running them with different parameters. By designing the appropriate stubs, remote optimization services, possibly leveraging cloud or grid computing infrastructures, can also be used.

The business intelligence of the *Grapheur* system is located in the Data aggregation engine. Its main role is to provide an algorithm- and domain-independent view on the provided solutions by applying dimensionality reduction and clustering techniques, as described in Sec. 4.

The Presentation and feedback dispatcher lies on top of the core component. Its purpose is to make data meaningful to the user by displaying it both in standard graphs and in user- and domain-specific ways by taking into account the user interaction environment. Some of its functions are:

- Domain-specific data displays: beside common graphing abilities, domain-specific charts can also be provided; for example, in Fig. 2 (top) a radio coverage chart is associated to each solution.

- *Cognitive sugar* allowing the user to rapidly identify places and solutions in the visualization space; see for instance flags (placed by the user) in Fig. 3, and ellipsoids that identify clusters of related solutions. The term *cognitive sugar* is chosen to remind the term "syntactic sugar" known in programming language syntax: although not strictly necessary, it makes the visualization "sweeter" for humans to use.
- Reaction to user feedback, whose purpose can be either to alter the current view by geometric transforms or selection mechanisms, or to ask for new solutions, for instance by zooming to a small portion of the view to see more solutions within stricter quality bounds.
- Environment-dependent display, ranging from dynamic 3D presentations with complex user input to static web pages with limited interaction for remote users accessing the system through a web browser.

The architecture is suitable for local use as well as for different kinds of remote operation. For example, a user can run a local copy of the *Grapheur* architecture, in order to have a dynamic high-level view of the current dataset, while delegating the optimization algorithm to a dedicated server accessed via a network stub. On the other hand, a user may access the whole system remotely as an online service with a more limited set of interaction mechanisms.

4 Reducing the dimensionality and the dataset size

When the number of inputs is large, and this is the standard case for combinatorial optimization with hundreds or millions of variables, reducing the dimensionality to the human accessible two or three dimensions is the standard issue to be addressed.

Of course, no optimal solution is available and the objective is to project the input coordinates x from n to 2 o 3 dimensions while maintaining some of the desired properties and information contained in the original data.

In our case, we concentrate on mapping the solution input coordinates, while a suitable color-coding is used to represent objective function values.

A natural way to reason about solutions is through basic concepts of similarity of solutions. The human questions a decision maker poses are of the kind:

- Are the various solutions radically different or similar?
- If solutions are different, can the possibilities be condensed into some relevant examples (prototypes) of the different solutions?
- Given that one prefers some types of solutions (e.g., because of some non-explicit criteria), can the system produce more solutions close to certain preferred prototypes?

To define computable procedures, the above issues require the definition of a metric $\delta(\boldsymbol{x}, \boldsymbol{y})$ to quantify the distance between two solutions. This metric can be given by the Euclidean distance in the solution parameter space, or by the Hamming

distance if solutions can be represented as binary strings. However, they might need ad-hoc, domain-dependent definitions.

To present anecdotal evidence of the different methods, in the following we present examples related to the Wi-Fi access point placement application [3]. The problem is to install a number of wireless stations (access points, AP) in an area with multiple and conflicting objectives, such as:

- the area in which at least one AP signal is above a minimum threshold (i.e, the network coverage) must be maximized;
- the area in which the sum of electromagnetic power is above a given threshold must be minimized (health protection);
- the deployment cost (number of APs, type of mount, distance from existing cables) must be minimized;
- other uses of the system (e.g., radiolocalization) must possibly be taken into account.

The objective function for the placement of N APs in a two-dimensional space with m objective criteria is in the form

$$\boldsymbol{f} : \mathbb{R}^{2N} \to \mathbb{R}^m,$$

where the configuration $\boldsymbol{x} \in \mathbb{R}^{2N}$ encodes the coordinates of the i-th AP in (x_{2i-1}, x_{2i}), $i = 1, \ldots, N$. While the most natural domain for the problem is $\mathbb{R}^{2N}$, in order to define a metric between solutions we need to consider symmetries: since the APs are indistinguishable, the objective function $\boldsymbol{f}$ is invariant by permutations of AP indices, therefore if $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^{2N}$ are two solutions, a good metric is given by

$$\delta(\boldsymbol{x}, \boldsymbol{x}') = \left(\min_{\pi \in \mathit{\Pi}_N} \sum_{i=1}^{N} \| (x_{2i-1}, x_{2i}), (x'_{2\pi_i - 1}, x'_{2\pi_i}) \|^2 \right)^{\frac{1}{2}},$$

where $\mathit{\Pi}_N$ is the set of permutations on $\{1, \ldots, N\}$.

4.1 Multi-dimensional scaling

After the metric is defined, visualizing the distance relationships among the myriads of solutions on a two or three-dimensional space is not a simple task. The methods that we consider are force-directed because they allow for dynamic interaction with the user.

Hook's Law-based models — The analogy is that of a physical system where nodes are connected by springs, which is relaxed to a (local) minimum energy state. In practice, "forces" in these methods influence displacements and not accelerations and the dynamics is often simplified in a pragmatic manner to obtain acceptable layouts in reasonable CPU times.
If $\delta_{i,j}$ are the original distances and $d_{i,j}$ the Euclidean distances among the projected

points, in general one aims at minimizing the differences between the original and the projected distances. The detailed definition differ by the way in which the errors are weighted. For example,

$$E_1 = \sum_{i<j} (\delta_{i,j} - d_{i,j})^2 \tag{3}$$

penalizes all errors in the same (quadratic) manner, while

$$E_2 = \sum_{i<j} \frac{(\delta_{i,j} - d_{i,j})^2}{\delta_{i,j}^2} \tag{4}$$

penalizes relative errors. Large errors on large distances count as small errors on small distances, which is in some cases consistent with human judgment.
The minimization is usually executed by gradient descent of the above error functions. Gradient descent has well known weaknesses for ill-conditioned problems, and faster and more robust methods can also be considered, for example the BFGS quasi-Newton used in [2].

Kamada-Kawai model [18] — A popular method is that of Kamada and Kawai (KK), inspired by the seminal Eades' work [8]. The KK energy is:

$$E_{\text{KK}} = \sum_{i<j} (\delta_{i,j} - d_{i,j})^2 \tag{5}$$

where $\delta_{i,j}$ is given by the *length of the shortest path* between two nodes. This definition encourages not only placing at a given distance directly connected nodes, but also placing at the double distance nodes that are two hops away, etc., therefore helping to create a more globally consistent display. Only one vertex is moved at each iteration, in a spirit similar to the stochastic gradient descent technique.

Fruchterman-Reingold model [10] — Another popular approach has been proposed by Fruchterman and Reingold (FR), who define the FR energy as

$$E_{\text{FR}} = \frac{1}{3k} \sum_{i<j} d_{i,j}^3 - k^2 \sum_{i<j} \log d_{i,j}. \tag{6}$$

After partial derivatives are calculated, the FR energy implies an attractive force proportional to d^2 between directly connected nodes from the first term, and a repulsive force between all nodes from the second term, a force proportional to $1/d$. In the above, k is the desired optimal distance between two vertices: a two-vertices system reaches the minimum energy value at distance k. Multiple attractions and repulsions complicate matters so that an empirical formula $k = C\sqrt{area/no.of vertices}$ with an adjustable C parameter is suggested. The model is usually presented in terms of forces, used to calculate displacements. To encourage convergence, the displacement of each vertex is limited to some maximum value, which decreases over time. To speed the algorithm up, repulsive forces are calculated only among reasonably

close nodes (within distance $2k$, with a grid-like spatial database: the candidates to be included in the repulsive force calculation are immediately identified in time $\Theta(\|V\|)$, avoiding examination of all couples in time $\Theta(\|V\|^2)$.

All previously proposed methods are closely related. In fact the underlying unifying paradigm is that of gradient-descent of a specific energy function. Some techniques intertwine energy and optimization method very closely, while separating the two helps in identifying the goals and the related quality measure (the energy function) from the methods to reach the goals, which can present the typical problems related to local minima trapping, dependence on initial conditions, etc.

Coming back to our objective of representing solutions distances, energy functions can be used to encourage a fair representation of all distances. In many cases some coarser way to reason about similar solutions is actually preferred. A possibility is to allow the DM to specify a distance threshold δ_T, i.e., so that he will consider as practically indistinguishable solutions within that distance. Based on the threshold, simple binary edges are introduced in the solution graph. Two nodes are connected if and only if their distance is less than δ_T. The above dimensionality reduction techniques can then be applied with these zero-one distances, as well as the FR technique.

4.2 Clustering

A human person has a limited amount of short-term memory to be used in problem solving activities. The precise estimate is of course fuzzy, but experts agree that it becomes difficult for a standard decision maker to analyze more than five to seven different options at the same time.

When the starting number of solutions is much bigger, ways must be developed to condense the relevant information by identifying clusters of similar solutions. After clusters are found, a representative solution for each cluster can be presented to the DM.

The cluster visualization process can proceed in a top-down manner, by first concentrating on macroscopic differences, and then progressively focussing onto finer and fined details.

The *Grapheur* visualization architecture follows the above requirements by clustering dataset elements. The clustering techniques available in the first release are:

- Hierarchical clustering: the user can browse through different cluster resolution levels, from one single cluster encompassing the whole dataset, and whose shape hints at the overall data distribution, to as many as the user wishes, with smooth animated transitions between levels that help the user perceive the underlying hierarchical relationship.
- k-means clustering, where either the user or the system decide the number of clusters.
- Quasi cliques [7] are a recently proposed method for identifying islands of connected components in graphs, where some parameters define the required con-

nectivity strength. Unlike the previously described clustering methods, they do not partition the dataset, but they allow for overlapping subsets and are able to disregard isolated elements.

In order to graphically represent a cluster, we operate in two ways. First, if a display property (such as color or shape) of the data items is available, it is used to encode the cluster. Second, the cluster itself is represented by means of its inertial ellipsoid, whose surface is the locus of points having unit distance from the cluster's average position according to the Mahalanobis metric given by the cluster's dispersion. In order to display the cluster ellipsoid using OpenGL primitives upon which the *Grapheur* system is based, let $\mathcal{C} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ be a cluster, and let $\boldsymbol{p}_i = (p_{i1}, p_{i2}, p_{i3})$ be the (3D) coordinates where item $\boldsymbol{x}_i$ is mapped by the dimensionality reduction procedure. The center $\bar{\boldsymbol{p}}$ of the cluster in the user space is therefore the mean value:

$$\bar{\boldsymbol{p}} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{p}_i.$$

Let the covariance components be defined as

$$S_{ij} = \frac{1}{n} \sum_{k=1}^{n} (p_{ki} - \bar{p}_i)(p_{kj} - \bar{p}_j), \qquad i, j = 1, 2, 3.$$

According to the OpenGL internal format, where points are represented as row vectors of homogeneous coordinates in $\mathbb{R}^4$ with the infinite plane represented as $(x, y, z, 0)$, the projective coordinate transformation mapping the unit sphere into the desired ellipsoid is represented by the following matrix:

$$T_{\mathcal{C}} = \begin{pmatrix} S_{11} & S_{12} & S_{13} & 0 \\ S_{21} & S_{22} & S_{23} & 0 \\ S_{31} & S_{32} & S_{33} & 0 \\ \bar{p}_1 & \bar{p}_2 & \bar{p}_3 & 1 \end{pmatrix}. \tag{7}$$

When moving between hierarchical clustering levels, cluster $\mathcal{C}$ will split into several clusters $\mathcal{C}_1, \ldots, \mathcal{C}_l$. To preserve the proper mental image, a parametric transition from ellipsoid $T_{\mathcal{C}}$ to its l offsprings $T_{\mathcal{C}_1}, \ldots, T_{\mathcal{C}_l}$ will be animated and the ellipsoids

$$T^{\lambda}_{\mathcal{C}_i} = (1 - \lambda) T_{\mathcal{C}} + \lambda T_{\mathcal{C}_i}, \qquad i = 1, \ldots, l$$

will be drawn with parameter λ uniformly varying from 0 to 1 in a given time interval (currently 1s). This will effectively show the original ellipsoid *morphing* into its offsprings.

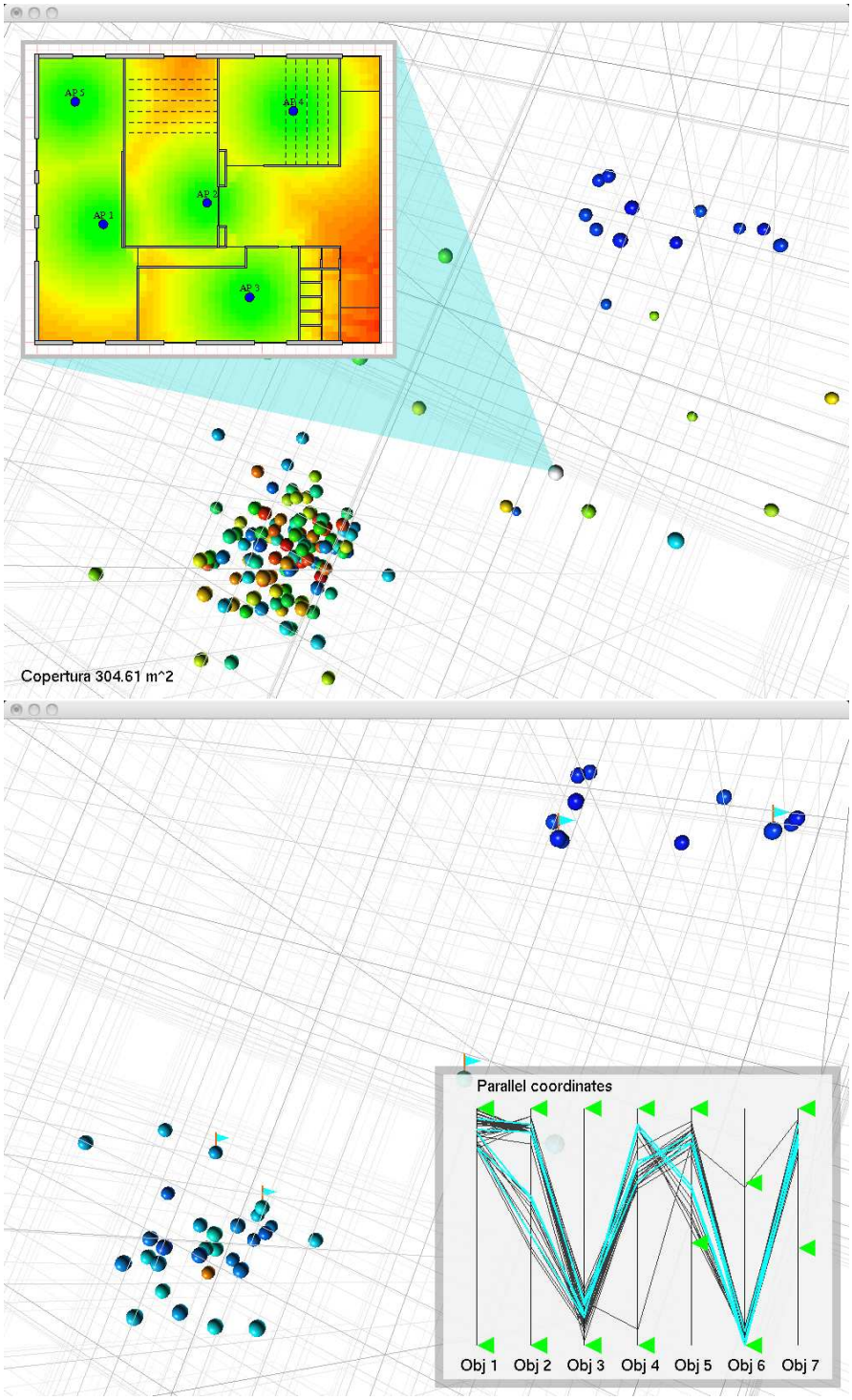

Fig. 2 Visualization of Wi-Fi coverage simulation results in a 3D fashion with different types of standard and problem-dependent charts. Top: a solution is highlighted and its corresponding coverage chart is displayed; bottom: parallel-coordinates interactive chart

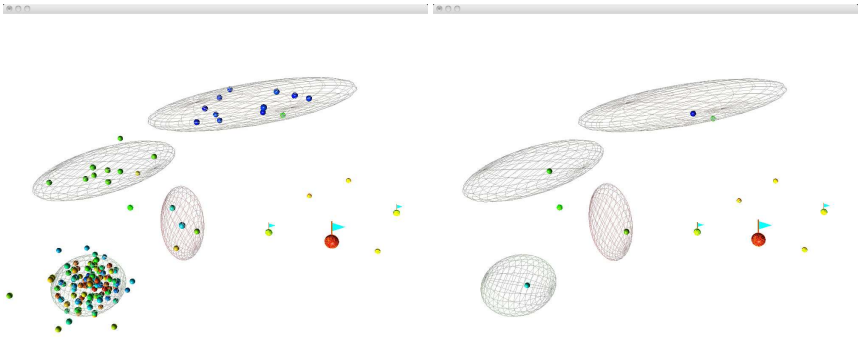

Fig. 3 Clusters of solutions identified by the data aggregation engine are displayed together with individuals (left) or with representative solutions (right).

5 User interaction and dynamic layout

Figures 2 and 3 contain four screenshots that exemplify some of the data aggregation and display functionalities of the *Grapheur* system. The AP placement application described in Sec. 4 has been used to generate a dataset of about 150 solutions, each representing the two-dimensional position of 5 access points, amounting at 10 coordinates, which are reduced to 3 by the Fruchterman-Reingold model (6). The rendering of the 3D mapping is obtained using the OpenGL library.

Solutions are represented by spheres, while their color encodes the main objective (coverage area, from red to blue). In the example, nearby solutions may correspond to dramatically different coverage areas because moving an AP by few centimeters from one side of the wall to the other can significantly change their effectiveness.

In Fig. 2 (top), a raw display of solutions is provided. The user is free to navigate through solutions by means of arrow keys or by other control devices, therefore zooming on particularly interesting areas. The user has clicked on one such solution, triggering the display of domain-sensitive data, in this case a map showing the corresponding position of the APs and the radio signal intensity.

While solution selection in the parameters space is done by navigating through the 3D user space, selecting solutions according to objective function values can be done by displaying ancillary charts, such as the parallel coordinates chart shown in Fig. 2 (bottom). In this figure, the user has already operated a range restriction on some objectives by moving the green handles, and consequently the number of displayed spheres has been reduced. Moreover, the user has placed some “flags” on interesting solutions, whose corresponding parallel coordinate profile is shown in cyan.

Clusters are represented as wire-frame ellipsoids defined by Eq. 7 that enclose the region of 3D space in which most solutions are displayed. Fig. 3 shows a number such clusters, together with the whole set of solutions on the left panel, while the

right panel shows how the user's cognitive burden can be reduced by only displaying the most representative solution of each cluster. Note that some clusters are actually singletons.

A useful visual hint for navigation is an absolute three-dimensional grid, shown in Fig. 2 that allows the user to move within the 3D space while maintaining a coherent mental map of his actions. A significant source of confusion in 3D dynamic representations is in fact the possibility of two equivalent mental representations, either user-centered (the fixed user moves the dataset to a convenient position) or data-centered (the user moves his own point of view within the dataset). The existence of an absolute grid allows the user to figure out that he is moving within the solution space, therefore making the correct assumptions about the effects of his actions. In the presence of other wire-frame objects, such as cluster ellipsoids, the grid may confuse the user, who can disable it.

6 Conclusions

We discussed the realization of a comprehensive software architecture for interactive optimization, with a particular emphasis on supporting flexible visualization.

The main advantages of the *Grapheur* framework are:

- Post-processing by clustering and statistical analysis. One starts from a set of solutions, or a set of possible choices, visualizes them and discovers relationships and hidden structure through clustering and dimensionality-reduction techniques.
- Seamless navigation in solution space, also with help by *cognitive sugar* items.
- Capacity to analyze solutions adapted to the human cognitive abilities. This requires that the tool goes into the background, becoming virtually invisible to the user, and that the human capability of forming mental maps and compact representations comes to the foreground.
- Versatility and neutrality: *Grapheur* can be used with different solution generator software and can be easily customized for different problems and usage contexts.
- Advanced human-computer interface including 3D-enabled mouse and joysticks.

The preliminary tests of the software environment in the concrete context of designing a wireless network by placing access points have shown the effectiveness of the approach in rapidly reaching a design preferred by the decision maker. Additional tests are under way in different application areas and with final users of different capabilities.

References

1. D. Anderson, E. Anderson, N. Lesh, J. Marks, K. Perlin, D. Ratajczak, and K. Ryall. Human-guided simple search: combining information visualization and heuristic search. In *Proceedings of the 1999 workshop on new paradigms in information visualization and manipulation*

in conjunction with the eighth ACM internation conference on Information and knowledge management, pages 21–25. ACM New York, NY, USA, 1999.
2. R. Battiti. First-and second-order methods for learning: Between steepest descent and newton's method. *Neural Computation*, 4:141–166, 1992.
3. Roberto Battiti, Mauro Brunato, and Andrea Delai. Optimal wireless access point placement for location-dependent services. Technical Report DIT-03-052, Università di Trento, 2003.
4. Roberto Battiti, Mauro Brunato, and Franco Mascia. *Reactive Search and Intelligent Optimization*. Operations research/Computer Science Interfaces. Springer Verlag, 2008.
5. Roberto Battiti and Andrea Passerini. Brain-computer evolutionary multi-objective optimization (BC-EMO): A genetic algorithm adapting to the decision maker. Technical Report DISI-09-060, University of Trento, Oct 2009 2009.
6. Piero P. Bonissone, Raj Subbu, and John Lizzi. Multicriteria Decision Making (MCDM): A framework for research and applications. *IEEE Computational Intelligence Magazine*, 4(3):48–61, August 2009.
7. Mauro Brunato, Holger H. Hoos, and Roberto Battiti. On effectively finding maximal quasi-cliques in graphs. In *Proc. 2nd Learning and Intelligent Optimization Workshop (LION2007 II, Trento, Italy, December 2007)*, volume LNCS 5313. Springer, December 2008.
8. P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42(149160):194–202, 1984.
9. Yaniv Frishman and Ayellet Tal. Online dynamic graph drawing. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):727–740, July 2008.
10. T.M.J. Fruchterman and E.M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991.
11. A.M. Geoffrion. The purpose of mathematical programming is insight, not numbers. *Interfaces*, 7(1):81–92, 1976.
12. D.L. Gresh and E.I. Kelton. Visualization, optimization, business strategy: a case study. *Visualization, 2003. VIS 2003. IEEE*, pages 531–538, Oct. 2003.
13. Y. Hamadi, E. Monfroy, and F. Saubion. Special issue on autonomous searcch. *Constraint Programming Letters*, 4, 2008.
14. H. H. Hoos and T. Stuetzle. *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann, 2005.
15. Frank Hutter and Youssef Hamadi. Parameter adjustment based on performance prediction: Towards an instance-aware problem solver. Technical Report MSR-TR-2005-125, Microsoft Research, Cambridge, UK, December 2005.
16. C.V. Jones. Feature Article–Visualization and Optimization. *INFORMS Journal on Computing*, 6(3):221, 1994.
17. M. Kadluczka and P.C. Nelson. N-to-2-space mapping for visualization of search algorithm performance. *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*, pages 508–513, Nov. 2004.
18. T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information processing letters*, 31(1):7–15, 1989.
19. M. Koppen and K. Yoshida. Visualization of pareto-sets in evolutionary multi-objective optimization. *Hybrid Intelligent Systems, 2007. HIS 2007. 7th International Conference on*, pages 156–161, Sept. 2007.
20. K. Miettinen, F. Ruiz, and A.P. Wierzbicki. Introduction to Multiobjective Optimization: Interactive Approaches. In *Multiobjective Optimization: Interactive and Evolutionary Approaches*, pages 27–57. Springer-Verlag Berlin, Heidelberg, 2008.
21. H. Pohlheim. Visualization of evolutionary algorithms-set of standard techniques and multidimensional visualization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 533–540. Morgan Kaufmann, 1999.
22. Davood Rafiei and Stephen Curial. Effectively visualizing large networks through sampling. In *WWW2005 Proceedings*, 2005.