VISUAL INTERACTION

For Solving Complex Optimization Problems

Alexander Hinneburg
University of Halle, Germany
hinneburg@informatik.uni-halle.de

Daniel A. Keim
AT&T Research Labs, USA and
University of Constance, Germany
keim@research.att.com

Abstract Many real world problems can be described as complex optimization problems. Some of them can be easily formalized and are amenable to an automated solution using some (usually heuristic) optimization algorithm. Other complex problems can not be solved satisfactory by automated algorithms. The reason is that the problems and the corresponding optimization goals can either not be fully formalized or that they vary depending on the user and the task at hand. In both cases, there is no chance to obtain a fully automatic solution of the problem. The only possibility is to make the user an integral part of the process. In this article, we therefore propose an interactive optimization system based on visualization techniques to guide the optimization process of heuristic optimization algorithms. To show the usefulness of our ideas, we provide two example applications: First, we apply the idea in the framework of similarity search in multimedia databases. Since it is difficult to specify the search task, we use visualization techniques to allow an interactive specification. As basis for the automated optimization we use a genetic algorithm. Instead of having an a-priori fully-specified fitness function, however, we let the user interactively determine the fitness of intermediate results based on visualizations of the data. In this way, an optimization with user-dependent and changing optimization goals is possible. The second example is a typical complex optimization problem – the time tabling problem. In most instantiations of the problem, it is not possible to completely specify all constraints, especially the potentially very large number of dependencies and soft constraints. In this application example, we also use visualization techniques in combination with automated optimization to improve the obtained solutions.

Introduction

Many tasks in the areas of resource management, planning, scheduling, data mining, information retrieval, graph drawing and many others can be described as optimizations problems. Examples of such tasks include similarity search in multimedia data bases, time tabling, mining for interesting association rules, searching for clusterings in high dimensional data or finding a graph drawing with some esthetic properties. All these problems have in common that the quality criterion for the optimization process can hardly be formalized since the problem involves a large number of user- and task-dependent constraints. The main problem is to communicate these complex constraints and dependencies to the computer. One can try to deal with the problem by allowing the user to specify additional constraints or parameters for the optimization process. Often, however, this is difficult since it is not intuitive and very time-consuming. In addition, there are a large number of cases where the users can not formally specify their requirements. Examples are image similarity search, where the user might want to focus on a detail of the query image like texture, foreground, background, or some object in the image; or graph drawing, where the users only have a vague idea what the esthetic properties of the graph should be and what compromises they are willing to accept.

The Process of Solving Complex Optimization Problems Let us consider the process of solving complex optimization problem via the computer in more detail. For obtaining a computerized solution, it is necessary to

1 formalize the problem,

2 communicate the constraints for the desired solution to the system.

3 develop an algorithmic solution of the problem, and

Even for rather simple problems, this can be difficult since it is unlikely that the user is able to completely specify the problem and the constraints. For more complex problems, it may even be impossible to formalize the problem and communicate all necessary constraints to the system.

Problems of this type occur in many complex real world applications. The communication problems between humans and computers may result from

- an internal representation of the problem which is difficult to understand for the user

- preconditions of the algorithm which are unknown to the user
- insufficient means for communicating vagueness, alternatives, willingness to compromise, etc.
- algorithms that are not designed to learn the users needs but deal with the user in an input-output fashion instead of a dialog.

Most of these problems can not be solved by better optimization algorithms but need a shift in the interaction paradigm. Our solution to this problem is to use visual interaction to improve the communication between the representation of the problem in the computer and the users and their goals. In this paper, we propose a framework for visual interaction which provides the possibility of a *non-verbal* and *non-parametric* specification of the problem and the constraints. The goal is to allow the user to deal with problems that have implicit user- and task-dependent constraints. Our approach employs well-known automated optimization algorithms and combines them with visualization and interaction techniques to allow an appropriate communication with the user – without the need for an explicit and complete specification of the problem and the constraints.

Related Work Our approach tightly combines research results from three areas: automated optimization, visualization, and interaction. In the area of automated optimization, there exist a large number of optimization algorithms such as simulated annealing Aarts and Korts, 1989, neural nets Rojas, 1996, greedy and genetic algorithms Goldberg, 1989, as well as constraint programming Bartak, 1999. Our approach does not depend on the particular optimization algorithm used, but employs the optimization algorithm as an exchangeable component and combines it with visualization techniques to overcome their problems. In the area of visualization, there are a large number of relevant approaches which allow the visualization of abstract information. Examples for information visualization techniques include geometric projection techniques, iconic techniques, hierarchical techniques, graph-based techniques, pixel-oriented techniques, and combinations hereof. For an overview of information visualization techniques the interested reader is referred to Keim, 2000. The visual interaction approach proposed in this paper uses multiple visualization paradigms to visualize the intermediate results of the optimization algorithms and to allow the user to interact with the optimization process. Our work is closely related to the work on interactive optimization by Karl Sims Sims, 1993 and Joe Marks et al. Anderson et al., 2000; Lesh et al., 2000. Both approaches also try to combine visualization and interaction to achieve better and/or faster solutions

to complex optimization problems. The approaches by Sims and Marks et al., however, use different visualization methods and the application areas are also different.

The rest of this paper is structured as follows: First, we introduce the basic idea of our visual interaction paradigm and discuss the contribution and new aspects of this work (see section 1). In sections 2 and 3, we demonstrate the usefulness of our ideas using two application examples in the areas of image similarity search and time tabling.

1. The Visual Interaction Paradigm

Most visualization systems combine visualization and interaction techniques, ranging from simple slider and drag & drop interfaces to more advanced techniques such as Fish Eye Views. These techniques have in common that the visualizations transform data from an internal representation into a easily recognizable visual representation. The visual representation, however, does not contain more information about the data or problem than the internal representation. With the paradigm of visual interaction we want to go beyond a simple transformation of the internal data. We want to use the medium of visualization in combination with interaction to capture additional facts about the data and the problem, which are hard or costly to capture and communicate otherwise. These constraints are usually not represented by the formal problem description (see unspecified portion of the problem in figure 1). A typical example are complex optimization problems. Because of the complexity the problem and the optimization goals, the formal specification of the problem is usually simplified, and captures only the necessary requirements but not the sufficient ones.

Let us briefly exemplify this situation by discussing the two application examples considered in this paper. In image similarity search, for each image in the data base a feature vector is derived, which describes some properties of the image. It is required that similar images have similar vectors. This requirement is a necessary condition for image similarity search. The other direction that similar vectors correspond to similar images is a sufficient condition which is hard to verify theoretically and to guarantee generally. To bridge the gap between the internal representation (feature vector) and the users mind (image similarity), in the easiest case we can, for example, show thumbnails and allow an interaction process to let the computer learn the users' notion of image similarity by varying the weighting of the features. This approach basically corresponds to the very effective relevance feedback approach in information retrieval Rui et al., 1998. Note, however, that the visual-

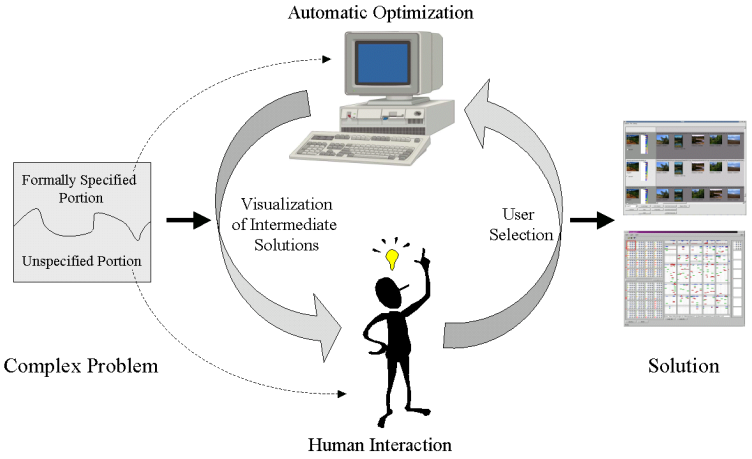


Figure 1. Visual Interaction Paradigm

ization does not necessarily need to show the image thumbnails but can also visualize important properties of the images as done in the *HD-Eye* system Hinneburg et al., 1999.

The second application – time tabling – is an optimization problem with a number of different constraints. The constraints ensure that solutions have few conflicts (for example: temporal conflicts, room allocation conflicts, etc.). The constraints are considered as the necessary preconditions. The sufficient constraints describing, for example, the preferences of professors and students, however, are almost impossible to be formalized due to the very large number of such constraints. As a consequence, only few schools and universities use computers for time tabling since automatic time tabling usually requires an intensive post processing with human involvement Burke et al., 1995. In our approach, we use visualization to represent the time tables and to point out the conflicts. By interacting with the system, the user gets an impression of what is possible with the available resources and how conflicts may be resolved. By directly interacting with the time table, the user can incorporate additional knowledge into the time table, which is not specified by the conflict constraints. It is also possible that a group instead of a single user guides the time table refinement.

In both application examples, the pattern is the same: The formalization of the problem does not fit the users needs because important constraints are not available to the computer. Without visual interaction, the experienced user can try to vary the input and learn the translation into the internal representation or do the post processing

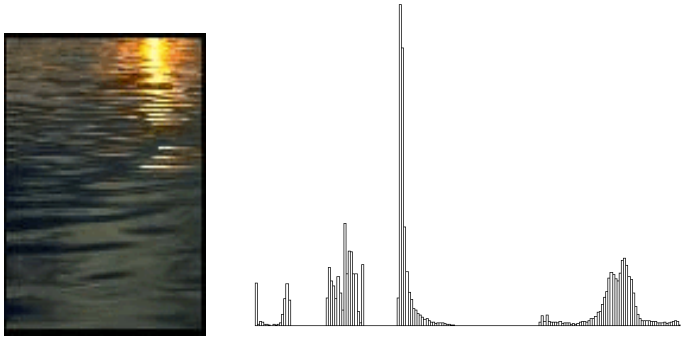
manually. Visual interaction, on the other hand, supports an easier and more effective communication with the computer, because facts, rules, relations, possibilities, and preferences can be communicated indirectly to the computer via the visualizations (see figure 1).

The paradigm of visual interaction can be used to circumvent situations, where the communication of constraints is hardly possible, or very time and space consuming. In the case of image similarity the communication is hard to be formalized since image similarity is not amenable to formal specification but can be easily expressed visually. A specification in non-visual form leads to a partial specification which neglects some facts about the intended type of similarity. In the case of time tabling, the communication using visual interaction circumvents very costly specifications, which are not realistic to be performed in practical applications. All constraints and preferences of professors and students could, for example, be specified and weights could be assigned to each of them. This specification, however, would be infeasible due to its time and space constraints since the number of configurations grows exponential in the number of rooms, lectures, professors, etc. The other point is that usually not all preferences are predecided by the users but are typically subject to a discussion of preliminary solutions which usually result in a compromise.

2. Visual Interaction for Image Similarity Search

Similarity Search in image databases is an example for a problem where the optimization goal is difficult to formalize. There is no single answer to the problem of finding the image which is most similar to a given query image. Depending on the user and the task, different images can be considered similar. One user, for example, may be interested in the foreground while an other user may search for images with a similar background. The interest of the user may relate to different objects in the query image, their arrangement, or even some patterns or color distribution of the images. Note that there is no single notion of similarity even for the same user since different features of the images may be of interest depending on the context and the search task.

Only few of the current similarity search systems for image databases support a flexible and adaptive similarity search. Most of them are either based on simple keyword search Corbis, 2000 or on some kind of color histograms Seidl and Kriegel, 1997; Cinque et al., 1999. More advanced systems include a number of different features such as keyword search, color distribution, object arrangement, etc. An example of an advanced system is the QBIC system Ashley et al., 1995 which requires



(a) Example Image (b) Corresponding Color Histogram

Figure 2. Image and its Color Histogram of an Image

a complex preprocessing (e.g., object recognition) but allows the user to perform a rather sophisticated similarity search. However, it is still difficult for the user to communicate his/her intended similarity to the system. Other recent approaches recognize the need of human feedback in image similarity search. The approach described in Böhm et al., 2001 focuses on an efficient support of an interactively refined image search and the approach of Inc., 2000 allows to refine the search by searching for similar images.

2.1 General Definition for Similarity Search

In Hinneburg et al., 2000, we proposed a generalized search algorithm which allows a more flexible and effective similarity search. The basic idea is to use the structure of the data to automatically determine a number of potentially meaningful similar images. The data representation used for the search are high-resolution color histograms of the images (cf. figure 2 for an example of an image and its color histogram). The procedure is based on a general optimization algorithm which tries to identify the relevant colors for a given query image based on the properties of the data distribution.

The problem may be formally described as a nearest neighbor search in high dimensional space. The color histograms can be seen as points in some high-dimensional data space and the search as a nearest neighbor search in some interesting projections of the high-dimensional data. Let $D = \{x_1, \ldots, x_n\},\ x \in \mathbb{R}^d$ be a database of d-dimensional color histograms, $x_q \in \mathbb{R}^d$ the query point, $p : \mathbb{R}^d \rightarrow \mathbb{R}^{d'},\ d' \leq d$ a projection, $C(p, x_q, D) \rightarrow \mathbb{R},\ C \geq 0$ a quality function which rates the quality of

the projection p with respect to the query point x_q. Then, the problem may be formally defined as

Definition 1 (Generalized Similarity Search)
A meaningful nearest neighbor for a given query point $x_q \in \mathbb{R}^d$ is the nearest neighbor $x_{NN} \in D$ in the subspace

$$p_{best} = \left\{ p | \underset{p:\mathbb{R}^d \to \mathbb{R}^{d'}, d' \leq d}{MAX} \{ C(p, x_q, D) \} \right\}.$$

The main problem is to find useful projections of the data, which can be seen as a translation of the users' intended similarity into a formal representation. A difficulty is that the space of all general projections P is infinite and even the space of all axes-parallel projections is exponential in d. It is interesting, however, that the data distribution contains highly relevant information, which can be used to restrict the search space. In other words, it can be observed that there is only a limited number of interesting projections for a given query. The optimization algorithm tries to determine the interesting projections. Since the "interestingness", however, depends on the user and the task, there is no general optimization function (or in case of the genetic algorithm: fitness function) but visual interaction is needed during the optimization process.

2.2 Application of Visual Interaction in Image Similarity Search

The main goal of the *VisOpt* system is to translate the intended similarity of the user into the formal representation used by the optimization algorithm. More precisely, the *VisOpt* system needs to find a projection of the feature space, where the metric on the projected feature vectors is adapted to the intended similarity. The *VisOpt* system uses the framework of genetic algorithms. Genetic algorithm are iterative heuristics and useful for the optimization of problems with many local extrema. In contrast to hill climbing methods a genetic algorithm scans many parts of the solution space at the same time. In each step, solutions from different parts compete for selection via a fitness function. The selected solutions with a high fitness are combined using crossover and mutation operations, and are the basis for the next iteration of the recombination process. Often it is difficult to define the fitness function. In our approach, we do not have to formally define the fitness function, but use visual interaction to determine the fitness of different projections depending on the users' input. The population size depends on the space requirements of a single visualization (in our case the result of a nearest

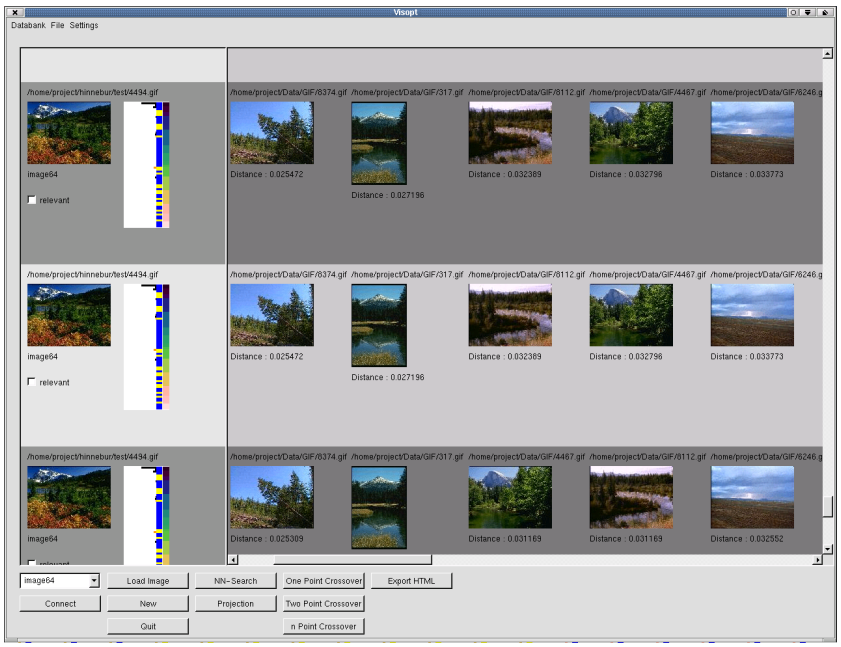


Figure 3. Screen Shot of the *VisOpt* system

neighbor search) with respect to the available screen size. Ideally the whole population should be visualized in one overview plot and detail on demand techniques should be used to access a single visualization. The population can be initialized using heuristics or - if none are available - at random. In our case we use the projections represented by a binary vector as input to the genetic algorithm. The genetic algorithm iterates until a satisfactory result is found or the number of iterations exceeds a given maximum. In each iteration the best (fittest) items of population gets selected for crossover and mutation to form the next generation. To find the fittest projections the *VisOpt* system presents the results of the nearest neighbor queries based on different projections to the user (see figure 3). The user selects the projections which are most relevant according to his/her current search task, thereby guiding the optimization process and providing the necessary input for the genetic optimization algorithm. The algorithm then tries to improve the intermediate results by combining the selected projections using crossover and mutation operations. The results are again shown to the user and the process is repeated until the desired results are obtained.

To further improve the search process, for advanced users the projection (binary vector) is visualized as a bar in the beginning of each

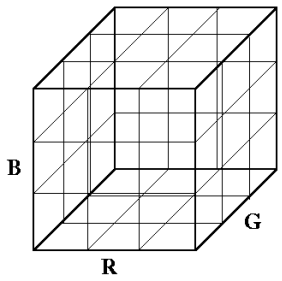


Figure 4. Structure of the Color Histogram

row (see figure 3), helping the user to get an intuitive understanding of what the color histogram bins mean for her/his notion of similarity in the given data base. This information may help the user to specify more complex crossover and mutation operators.

2.3 Experimental Evaluation

We use the *VisOpt* system to demonstrate the power of our proposed framework. First, we show that our method produces better results than traditional image similarity search approaches. Second, we demonstrate the power of our framework to easily adapt to the users' requirements.

In the first example, we compare the effectiveness of our method with k-nearest neighbor search based on color histograms. We use multidimensional histograms which consist of the normalized bin counts in the three dimensional RGB color space (see figure 4). Since the three dimensional structure is linearized to a feature vector the correlation relations among the dimensions is complex. We used feature vectors with dimensionality $d = 512 = 8 \times 8 \times 8$. To demonstrate the improved effectiveness we show a query image containing a sunset at sea. Figure 5(a) shows the result of the k-nearest neighbor query based on the full histogram (without projections) and (b) is the result obtained after 4 iterations using *VisOpt*. The k-nearest neighbor search found only one image related to a sunset while *VisOpt* found six images containing a sun set.

In our second experiment, we use an image with yellow trees on the foreground and mountains in the background. We used *VisOpt* to focus on the foreground and alternatively on the background. In Figure 6 we present the results after five iterations. The example shows that the optimization can be easily adapted according to largely different tasks. Note that we only used color histograms for our similarity search, and

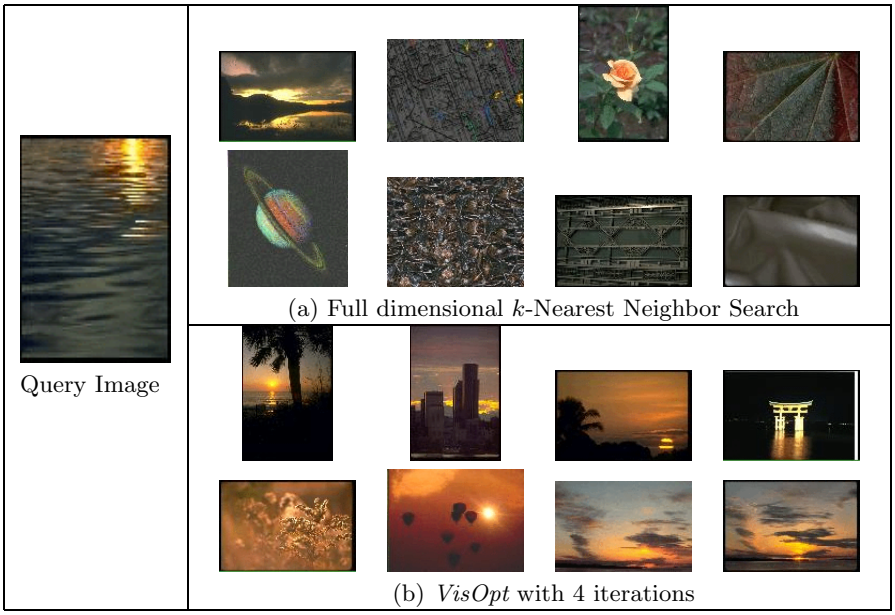


Figure 5. Comparison with full dimensional k nearest neighbor search

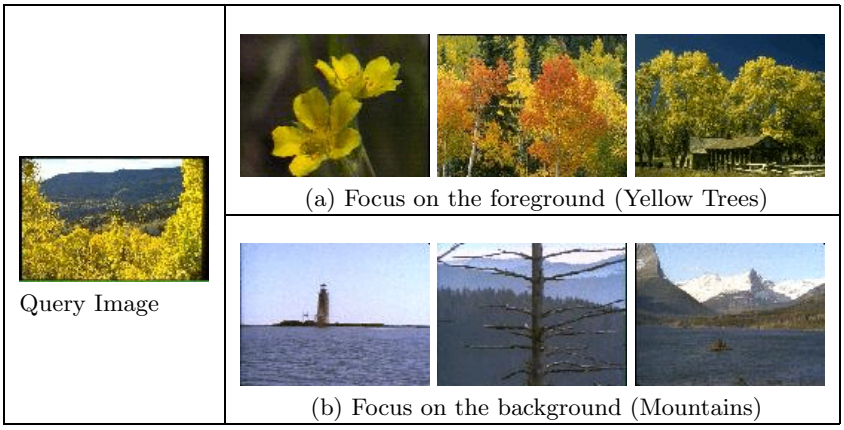


Figure 6. Simulation of users with different focus

that semantic search issues are not necessarily connected to the color distribution in the image.

3. Visual Interaction for Time Table Construction

The problem of time tabling is a hard optimization problem in general. It has been shown that it can be transformed into the graph coloring problem which is NP-complete. The problem can be formalized by taking a number of different constraints into account:

1 Student Constraints
Obligatory lectures and exercises for students in the same semester and the same field of study can not be at the same time. Voluntary lectures and exercises for students in the same semester and the same field of study should not be at the same time. In addition, students from different fields of study may also have to take part in the same lectures and should not have something in their own field of study in parallel (constraints across fields).

2 Professor Constraints
The same professor can not give two lectures at the same time and has times where s/he is not able to give lectures. In addition, each professor has certain preferences for his/her lectures.

3 Room Constraints
The number and sizes of rooms that are available at each point of time is limited. In addition, the location of rooms must be considered so that the student are able to reach the locations in time.

A large number of highly optimized software packages have been developed to solve this problem. Many of the solutions are based on well-known optimization algorithms such as simulated annealing Melício et al., 2000, genetic algorithms Colorni et al., 1990 or constraint-based reasoning Burke et al., 2000. Despite the long history of research in the area of time tabling, the research field is still very active since in practice the systems often do not produce satisfactory solutions (for reasonably sized problems) Burke et al., 1995. The reason is not that the algorithms do not work well enough but that it is impossible to specify all constraints and dependencies which have to be considered. In the above example, it is, for example, impossible to specify all preferences that each professor may have. This is due to the fact that the preferences are usually not hard but soft constraints and also depend on the preferences

of the other professors. In the example, one professor might be willing to give his lectures on Friday afternoon but others exclude this possibility completely. So some professors are getting the better slots while those who did not exclude the unattractive slots will certainly get those slots, which is not a satisfactory solution and will lead to requests for changes.

We developed a first prototype to combine visual interaction with time tabling. Figure 7 shows a screen shot of our system. The two squares on the left side shows conflict summaries of different time tables. The upper square contains the parent population (size 16). The lower square contains the result of a crossover operation of two time tables of the parent population. The bar on the right side contains a list of the best time tables seen so far[1]. The large area in the middle shows a single time table in detail. Each lecture correspond to a row and the green or red bricks correspond the time slots used. The red bricks stand for a setting which causes conflicts with other lectures. The lectures in our example are partitioned into six groups which are divided by small horizontal rows. The bars in the dividing grey areas shows the time and room conflicts for the group below. The grey area at the top shows a summary of the groups.

The visualization can be used to directly interact with the time table under consideration. The user may, for example, move lectures to different time slots and gets immediate feedback on the resulting conflicts. The user may also interactively specify other constraints or run the automated algorithm to further improve the solution. Using the tight integration of visualization and automated optimization algorithms, the user may easily discover and integrate additional context-dependent constraints. A detailed evaluation of the improved effectiveness can be found in Löwe, 2000.

4. Conclusions

There is a significant number of optimization problems where the optimization goal is varying depending on the user and the task. In many cases, the optimization goal can not be formalized and is therefore not amenable to an automated solution using optimization algorithms. In this paper, we therefore propose to make the user an integral part of the optimization process by using visual interaction. We show two examples of optimization problems, where visual interaction can help to communicate constraints which are otherwise hard to express. We show that using visual interaction leads to better solutions of difficult optimization

[1]The best time tables are those with a minimum number of conflicts.

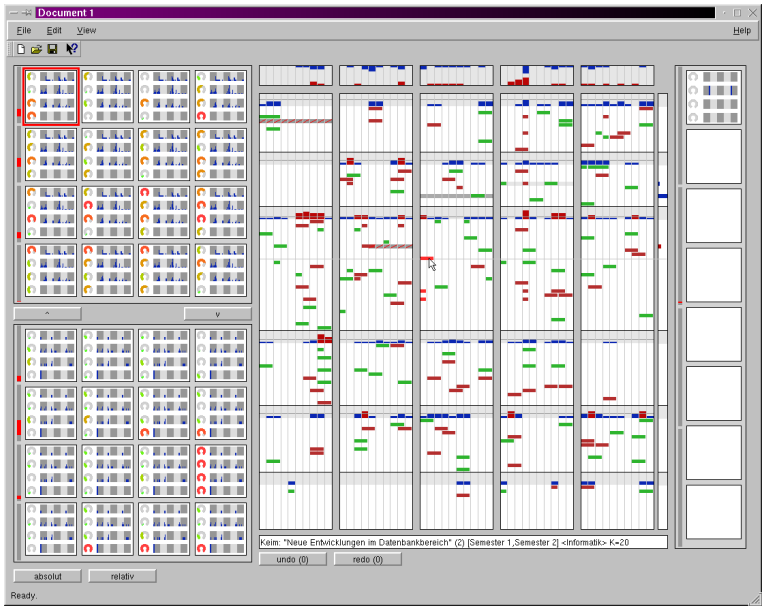


Figure 7. Screen Shot from the Visual Time Tabling system

problems and can be effectively used in a wide range of applications. We hope that the presented ideas may stimulate research into that direction. Future work includes the application of our visual interaction framework to other areas with complex optimization problems – especially graph drawing and data mining.

References

Aarts, E. and Korts, J. (1989). *Simulated Annealing and Boltzmann Machines*. Wiley.

Anderson, D., Anderson, E., Lesh, N., Marks, J., Mirtich, B., Ratajczak, D., and Ryall, K. (2000). Human-guided simple search. *Conference Proc. of AAAI*, pages 209–216. `http://www.merl.com/papers/TR2000-16/`.

Ashley, J., Flickner, M., Hafner, J. L., Lee, D., Niblack, W., and Petkovic, D. (1995). The query by image content (QBIC) system. In *Proc. of the 1995 ACM SIGMOD*, page 475. ACM Press.

Bartak, R. (1999). Constraint programming: In pursuit of the holy grail. *Proc. of WDS99 (invited lecture)*.

Böhm, K., Stojanovic, A., and Weber, R. (2001). Implementing relevance feedback techniques for large image collections efficiently. *Multimedia Information Systems*, pages 113–122.

Burke, E., MacCarthy, B., Petrovic, S., and Qu, R. (2000). Structured cases in CBR: Re-using and adapting cases for timetabling problems. *Knowledge-Based Systems*, 13:159–165.

Burke, E. K., Elliman, D. G., Ford, P. H., and Weare, R. F. (1995). Examination timetabling in british universities - A survey. In *Int. Conference on the Practice and Theory of Automated Timetabling ICPTAT'95, Edinburgh, UK*, pages 423–434.

Cinque, L., Levialdi, S., Olsen, K. A., and Pellicanò, A. (1999). Color-based image retrieval using spatial-chromatic histograms. In *Proc. of the IEEE International Conference on Multimedia Computing and Systems*, volume II, pages 969–973. IEEE CS Press.

Colorni, A., Dorigo, M., and Maniezzo, V. (1990). Genetic algorithms and highly constrained problems: The time-table case. *Proc. Int. Workshop on Parallel Problem Solving from Nature, LNCS 496, Springer*, pages 55–59.

Corbis (2000). `http://www.corbis.com`.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addision-Wesley.

Hinneburg, A., Aggarwal, C. C., and Keim, D. A. (2000). What is the nearest neighbor in high dimensional spaces? In *VLDB 2000, Proc. of 26th International Conference on Very Large Data Bases, 2000, Cairo, Egypt*, pages 506–515. Morgan Kaufmann.

Hinneburg, A., Wawryniuk, M., and Keim, D. A. (1999). HD-Eye: Visual Mining of High-Dimensional Data. *Computer Graphics & Applications Journal*, 19(5):22–31.

Inc., B. S. (2000). Diggit! `www.diggit.com`.

Keim, D. A. (2000). An introduction to information visualization techniques for exploring large databases. In *Tutorial Notes of the IEEE Int. Conference on Visualization*. IEEE CS Press.

Lesh, N., Marks, J., and Patrignani, M. (2000). Interactive partitioning. *Proc. of the conference on Graph Drawing*.

Löwe, J. (2000). VisTime: Visual Interactive Timetabling (in German), Master Thesis, University of Halle, Germany.

Melício, F., Caldeira, P., and Rosa, A. (2000). Solving the timetabling problem by simulated annealing. *in Enterprise Information Systems, Kluwer Academic Publisher*, pages 171–178.

Rojas, R. (1996). *Neural Networks A Systematic Introduction*. Springer, Berlin.

Rui, Y., Huang, T., Ortega, M., and Mehrotra, S. (1998). Relevance feedback: A power tool in interactive content-based image retrieval. *IEEE Transactions On Circ. Sys. Video Tech.*

Seidl, T. and Kriegel, H.-P. (1997). Efficient user-adaptable similarity search in large multimedia databases. In *Proc. of the 23rd VLDB-Conference*, Athen.

Sims, K. (1993). Interactive evolution of equations for procedural models. *The Visual Computer*, 9:466–476.