# Design Visual Thinking tools for Mixed Initiative Systems

Pearl Pu
Ergonomics of Intelligent Systems and Design
Database Laboratory
Swiss Federal Institute of Technology (EPFL)
CH-1015 Ecublens, Switzerland
pearl.pu@epfl.ch

Denis Lalanne
Avenue Major Davel 44,
1800 Vevey, Switzerland
denis_lalanne@hotmail.com

## Abstract

Visual thinking tools are visualization-enabled mixed initiative systems that empower people in solving complex problems by engaging them in the entire resolution process, suggesting appropriate actions with visual cues, and reducing their cognitive load with visual representations of their tasks. At the same time, the visual interaction style provides an alternative to the dialog-based model employed in most mixed-initiative (MI) systems. Visual thinking tools avoid complex analyses of turn taking, and put users in control all the time. We are especially interested in implementing visual "affordances" in such systems and present three examples used in COMIND, a visual MI system that we have developed. We show how humans can more effectively concentrate on synthesizing problems, selecting resolution paths that were unseen by the machine, and reformulating problems if solutions cannot be found or are unsatisfactory. We further discuss our evaluation of the techniques at the end of the paper.

## Keywords

Visual thinking, mixed-initiative systems, interactive information visualization, visualization of complex information, visualizing algorithms

## INTRODUCTION

Mixed-initiative systems (MIS) refer to intelligence systems for which users' input and intervention are solicited during the entire automatic reasoning process. Important issues in earlier dialog based MIS were turn taking analysis as well as natural language processing. More recently, researchers began combining direct manipulation principles with work on intelligent agents [8]. Such systems avoid complex analysis of turn taking, and engage users in more vivid graphical user interfaces. A step further in this direction is to employ visualization techniques [4] to create visual metaphors that will reduce the cognitive load of MIS users and empower them with extra values: 1) external representation of user's task; 2) visual cues to influence human strategy, and solution path selection; 3) possibility to add "human" criteria in the systems' decision making process; 4) and visualization of conflicts so that humans can reformulate problems and explore unseen paths.


*IUI'02*, January 13-16, 2002, San Francisco, California, USA.


In this paper, we draw examples from the specific domain of configuration tasks to illustrate various visual metaphors, although the principles developed in our work have been applied to airline rescheduling systems [16], travel planning [20], product and robot design [11], as well as potential areas such as personal computer, automobile, and insurance policy configuration. Configuration is a difficult computation task for both human and machines, thus an ideal case for MIS. Neither human nor machine alone can solve the problem with satisfactory results.

We first introduce the configuration tasks and describe the artificial intelligence part of our MIS. Then we concentrate on the discussion of the visual metaphors. We show how users rely on **Kaleidoscope** for selecting search strategies, **Tradeoff Maps** for performing multivariant tradeoff analysis in high dimensions, and **Conflict Resolution Lattice** for discovering conflicts in the original problem definition. We also present a user study of our visualization-based UI system, followed by a review of related work.

## Configuration tasks using constraint satisfaction techniques

Mixed-initiative systems rely on inference engines to solve at least some parts of the problem automatically. We found it useful to think in terms of abstraction techniques in identifying the AI part of a MI system. Abstraction, according to the AI literature [5], is an encoding of a problem for which an inference engine exists. In the case of configuration tasks, we use constraint satisfaction problem solving [24] as an abstraction method. The inference engines are various CSP algorithms. JCL [23], the Java Constrain Library, is a repository of some 15 CSP algorithms. Because CSPs are NP-complete, there is no single CSP algorithm that is good for all configuration problems. Human ingenuity is required to choose among the algorithms the most suitable ones depending on problem context.

### Visual reification of tasks

The configuration task has been externalized in a visual interface shown in Figure 1 for the landscape design problem.[1] It is a problem of assigning land pieces to structures in a community. At present, there are six structures: a house complex, an apartment complex, a school, a recreation area, a dumpsite and a cemetery. Eight land pieces are available for construction. Each of them has its characteristics, such as view, noise level, and ground characteristic. Some lots such as lot 11 and 17 can be only used for the cemetery, the dumpsite, or the recreational area. A solution of the landscape design problem (e.g., lot3 -> dumpsite, lot5 -> school, lot7 -> cemetery, lot9 -> apartment, lot10 -> recreational area, lot12 -> houses) is an assignment of land pieces to structures so that constraints are satisfied and criteria are optimized. Do not put a dumpsite next to houses is an example of a constraint. A solution is optimal when the total satisfaction of land use is the highest. Optimization in this case is multi-variant in terms of cost, noise level, quality of the view, closeness to school, etc.

### Kaleidoscope

The first CSP algorithm is a straight-forward backtracking search by assigning land pieces lot 3, lot 5, lot 7, lot 9, lot 10, lot 11, lot 12, or lot 17 to each structure. At the same time it checks for constraint violations. Kaleidoscope (Figure 2) visualizes the internal state of the search process, which starts from the $0^0$ line. It sweeps the entire search space represented by the disk in counter-clockwise direction. The assignment of all structures to existing lots is called a valuation. Unsuccessful valuations are immediately discarded and the algorithm goes back to the last non-violated variable. Black segments correspond to successful assignment of one structure. Thus a successful valuation is a single black line reaching all the way to the exterior circle. On the other hand, some areas do not have any black lines due to violation of constraints occurring right from the beginning. Some areas have short black lines, but they are unable to reach beyond the circle because violation of constraints occurred in later assignments. The color prohibiting a line from penetrating the exterior circle corresponds to the constraint (same color) that has caused the failure of this assignment (Figure 3). These color patterns (black lines and color elements) guide humans to change the current search strategies if results so far are not satisfactory.

For instance, a phenomenon called thrashing [13] during backtracking occurs when the search repeatedly fails on a certain combination of values. This corresponds to a large area in Kaleidoscope blocked by the same color. We can detect easily the constraint responsible for the thrashing by looking up the color coding as shown in Figure 3. The area in light gray indicates that constraint recre !=17 (recreational structure is not to be built on lot 17 because of a severe slope) is blocking many solutions.

The origin of thrashing is that the search algorithm always starts with recre <- lot 17 when lot17 should be eliminated from consideration for the recreational area. To remedy this problem, more intelligent techniques, called pre-processing algorithms [6,17] have been developed that attempt to eliminate as many inconsistencies as possible before the search begins. In addition, search can start with structures that have the most difficulty of finding a land piece. This algorithm, called dynamic variable ordering, can reduce the number of search steps from 343 to 120 in the case of landscape design. It is very effective for large problems. Figure 5 shows the same problem with two different variable orderings.

In addition, a Monte Carlo search algorithm, called the Knuth algorithm [10], can be used to assess whether a search space has numerous solutions or not. The main idea is to explore several places of the search space quickly, although randomly. Users can draw an idea of the shape of the space in order to do a more precise search on a specific part of the space with the backtracking method (Figure 4).

To summarize the first visualization technique, we enumerate the set of inferences that users can perform relying on information given in Kaleidoscope:

- Is the current algorithm fast enough for generating solutions? If not, use Knuth algorithm to guide the selection of search strategies.
- Does thrashing occur and in what type of frequency? If so, preprocessing algorithms should be employed.
- Are solutions diversified or concentrated in clusters?
- Are solutions abundant
- Is search futile? If so more efficient or random algorithms should be used

## PARETO SPACE FOR 2D TRADEOFF ANALYSIS

The second visualization consists of multi-criteria tradeoff analysis and a map of the solution space. This model is especially useful for under-constrained problems where many solutions compete for attention. Users rely on visualization assistance in order to evaluate solutions and make intelligent decisions with human criteria.

Consider the same landscape design example and suppose users have selected two evaluating criteria to maximize: cost factor and view. Figure 7 shows the solution space where nodes represent solutions, and x-axis and y-axis represent solution's performance on the selected criteria. A node, which performs best on both criteria, is called the dominant solution. If such a solution is absent (as in Figure 7), nodes lying on the outer rim of the solution space are called non-dominant nodes (in red). All other nodes (in blue) are called dominated solutions. In most cases a solution map does not contain a dominant node, thus requiring tradeoff analysis to choose a winner (often a non-

[1] Color figures are available at http://lbdsun.epfl.ch/f/staff/pear/.

dominant solution) based on users' dynamic preference measure. For example, if a user prefers to optimize more on the view, but is willing to compromise on cost, then he can choose the red node lying farthest on the x-axis, even though the y-value is not the best. When several competing solutions exist, users will be prompted to choose additional criteria so that some dominant solutions might emerge. However, this visualization (it's called pareto space optimization [19]) is feasible for up to three criteria. Many real-world problems call for tradeoff analysis in much higher dimensions.

### Tradeoff Map

Our new visualization (Figure 11) overcomes this limit by combining color patterns, visual structures and interactivity. It is called tradeoff balance because the solutions are mapped to the x-axis much like the behavior of weights in a balance. Given a solution having 4 criteria as those shown on the right of Figure 11, we calculate its position in the tradeoff balance as follows. The y coordinates represent the total sum of all criteria values (criterion is always optimized instead of max- or minimized). Thus the solution performs the best overall-speaking is the node with the largest y value. To calculate x coordinates, we imagine four weights, corresponding to the criteria values of this solution, lying on the bottom of this tradeoff map. The x coordinate is the center of mass of these weights. While y represents an absolute performance value, x shows the distribution of the underlying criteria much like a balance. When tradeoff analyses are required, users can slide to the left or right from the middle depending on his current preferences. For instance the current solution (shown on the left of Figure 11) performs well on the view, but it is not the most cost effective one. There are cases where solutions are pulled by multiple criteria values, but individually these criteria are not distinguishable. In this case interactivity solves this problem by allowing users to click on the solution nodes and clarify the details. Notice the colors are coded in the small display in synchrony with criteria bars on the bottom.

For tradeoff analysis in multi-variant spaces, we associate the following inferences:

- Is there a dominant solution?
- Are there numerous or few non-dominant solutions?
- Should additional criteria be defined in order to push out dominant solutions?
- Are solutions cluttered around a certain area, or more spread out in the map?

## CONFLICT RESOLUTION LATTICE

The third design consists of a set of reasoning algorithms to discover conflicts in over-constrained problems and the use of a lattice as a visualization tool. Conflicts occur when either the given constraints contain inconsistencies (such as x = y and x > y defined simultaneously) or too many constraints have been defined. Conflict resolution involves pinpointing the set or sets of constraints responsible for over-constrained CSPs so that humans can either eliminate or repair them.

A conflict set is a set of constraints for which no solution exists. Visually they are represented as black (minimal conflict) or dark blue (regular conflict set) squares in an interactive lattice (e.g., Figure 10). The top three sets in Figure 10a are the smallest conflict sets and correspond to constraints pointed by the arrows. The meaning of these constraints will be explained in detail in a moment. Sets are further ordered from the top to the bottom by their sizes with the set on the bottom being the largest. When a square is black, it is a minimal conflict set, and any set containing that set is also a conflict set. When it is dark blue, the set blocks a certain number of solutions. The darker it is, the more potential solutions it blocks. The human conflict solver is to eliminate the black conflict sets, or to find the smallest (top most) and darkest blue set of constraints to relax. While normal lattices contain lines to relate sets to their sub or supersets, this lattice is interactive and only displays the set and superset relationship when a set is clicked on. In Figure 10b, the bottom-most white square is the supper set of all white squares lying above it. This allows displaying a large lattice without the risk of having lines crisscrossing and thus causing visual overloading. We now discuss the different cases of conflict sets.

### Inconsistent CSP

As mentioned, an example of inconsistent CSP is when inconsistent constraints are defined at the same time. Shown in Figure 8 is a simple inconsistent CSP with its definition and lattice visualization. The squares on the top row represent the sets of constraints to be disjunctively eliminated in order to restore consistency to the problem. Furthermore, if one clicks on the square, a number is shown in the side window to indicate the number of potential solutions obtainable. Thus, eliminating x=y will produce 28 solutions, while taking out x>y will generate 8 solutions.

### Over-constrained problems

An over-constrained problem is one where one or several sets of constraints are defined in such a way that no solution could exist. Further if any bigger CSP contains this over-constrained CSP, it will not have any solutions either. Identifying such minimal conflict sets thus becomes a major task. In the visual lattice, this type of conflict sets is displayed in black (Figure 9). Consider the example of the map-coloring problem of three neighboring countries. If each country can be colored only in red or blue and no two neighboring countries should have the same color, then the problem is over-constrained. To find a smaller conflict set to repair, a user clicks on the black square in the lattice. Three blue subsets are shown. Further, it is

shown that repairing any of them will allow exactly 2 solutions.

### Multiple conflict sets

Some problems have several minimal conflict sets as indicated by the black squares in the red/black lattice of Figure 6. Here is an example of selecting watch design criteria and their relationships:

- (`beauty` > 3) -> (`implementation` >3)
- (`complexity` > 3) -> (`implementation` > 3)
- (`usability` > 3) <-> (`complexity` < 5)
- `usability` == 4
- `complexity` == 5
- `beauty` == 4
- `implementation` ==1

The black squares in Figure 6c correspond to all minimal conflict sets. At least one of the minimal conflict sets has to be relaxed to generate any solutions. As before, if a black square is clicked on, Lattice shows subsets (see lattices in Figure 6d). The subsets define parts of the original problems for which solutions exist. This information guides the users to choose the subsets to keep. For example, clicking on the black square on the first lattice in Figure 6d, two subsets are shown. Thus either we keep the first subset or the second. Since the second subset is a larger set, keeping it automatically allows us to keep the largest original problem. By visualizing all conflicts and largest consistent subsets, we offer designers the choice of what to throw away and what to keep, which is the most difficult decision in design and requires experience, gut feeling and dynamic criteria from the designers.

### Futile search

The worst scenario of an over-constrained CSP is one where there are actual solutions, but the search algorithm will take exponential amount of time to find them. For a CSP of size larger than 50 parameters and an equal amount of constraints, it is almost impossible to wait for the results. Thanks to the Knuth algorithm, we can estimate very quickly if the search space of a given CSP is poor. If this occurs, then conflict resolution involves finding the degree of constrainedness of sets of constraints. As in the case of over-constrained problems, we have the notion of sets of constraints. Instead of being a conflict set, these sets allow only few partial solutions, thus few full solutions as well. We define the notion called the blocking rate, which is a percentage of no-good partial valuations over all possible combinations.

In the lattice visualization, recall that the sets are colored in blue and the darker the blue, the more blocking it is. If there are multiple blocking sets, then which one should the user focus on first? The sets lying on the top row are smaller than those underneath. The smaller the set is, the easier it is for users to modify the constraints. The general heuristic is thus to find the smallest (top most) and darkest blue set of constraints to relax. Furthermore, this visualization is linked to the map of solutions as described in the section on Tradeoff Map. Each square on the lattice will invoke secondary windows to display the potential solutions that could be obtained if the constraints had been relaxed. The advantage of this coupling is to avoid searching unless the users are certain about the quality of solutions that they will obtain.

Shown in Figure 12 is a design scenario of a pen which consists of a cap, a body, a button, and a tube in addition to several other variables. As can be seen in the Kaleidoscope, there are only two valuations satisfying all the constraints in the possible space of solutions. With the help of the lattice, we realize that the squares on the top most row look promising since they are small and can potentially yield 26 and 10 solutions respectively. Further, the tradeoff map shows that the solutions permitted by the first square of constraints are ranking high in the tradeoff analysis. Thus one can easily try to either eliminate that constraint or relax it. To summarize this section on over-constrained problems, we list the inferences representing different reasoning tasks and the corresponding results we can get from the lattice visualization:

- Is the given CSP problem over-constrained: a single or several black squares in lattice
- Which one of the conflict sets to relax: either use the side window to select the most optimal one, or look up in the constraint definition to find the most appropriate one
- If certain conflicts are removed, potential solution characteristics are reflected in the map of solutions in Tradeoff.
- If the search is futile, then the degree of constrainedness will lead users to relax certain constraints and obtain more solutions

## EVALUATION

Since we have selected problems that were generally hard to solve neither by human or machines alone, we did not evaluate subjects on their problem solving skills without COMIND. We are interested in knowing if the visual metaphors provide intuitive cues and whether they speed up problem resolution.

We observed 6 industrial design students on how they solve three real design problems in COMIND (one under-constrained, one inconsistent, and one over-constrained).

- Almost all of them used the Knuth algorithm to first find out if the problem yields solutions or is over-constrained (usage of visualization to gauge the search space).
- Almost all of them within 20 minutes solved the two design problems, one being under-constrained, and one over-constrained (normally when they encountered over-constrained problems, they take much longer to find compromises).

- Most of them with a brief explanation of the lattice can use it as a tool to navigate in the constraint editor to modify the problem and later obtain interesting solutions (lattice is intuitive in the problem context).
- Most of them used the interactive search feature (it's fun - was often the remark) in Kaleidoscope while search was underway and construct different search strategies depending on constraint characteristics.

In general, we are satisfied with the learning speed of our subjects. Because the task has been externally represented, subjects have a good mental model of the functionalities of COMIND with minimal explanation. The average duration of design problem solving is much shorter than what they would take if they were to solve these problems in COMIND but with no visualization as feedback. Furthermore, the user's knowledge retention of COMIND is quite good. When subjects are asked to come back to reuse COMIND, almost no explanation needs to be given again.

## BACKGROUND AND RELATED WORK

### Hybrid reasoning and mixed initiative systems

Mixed-initiative systems refer to artificial intelligent systems where users input and intervention are solicited during the entire automatic reasoning process. They are in the middle of the spectrum between two extreme interaction styles [15]: those that employ autonomous interface agents and those that use computer strictly as tools. A set of general principles for developing such mixed-initiative systems was presented in [8], and a set of principles for the specific domain of interactive search [1]. In addition to those principles, we found that humans' goals and criteria are highly dynamic and unpredictable. So instead of using computational resources to judge their goals, we emphasize the formulation and visual reification of users' tasks. We start the design of UIs from the first principle by analyzing the users and their tasks. We have designed in our system the possibility to allow users to reformulate the original problems before search begins, and select criteria in the tradeoff process. [2] presented an example on the extreme end of the interaction spectrum where search engines are viewed strictly as tools to find local minima and humans guide the search by designating promising spaces.

TRAINS [7] is an interactive planning systems where multi-modal interaction using speech, natural language were combined with plan recognition, planning, and simulation components in a human-computer collaborative environment. Visage [21] uses visualization to explain scheduling results, but has not been used to prompt users for appropriate problem-solving actions.

### Knowledge crystallization

In the introduction chapter of [4], Card *et al* gave an example of the knowledge crystallization task using visual tools. It consists of gathering information from diverse sources (foraging), searching for a representational form (a schema) to capture the data, modifying the representation to include all attributes, and performing tradeoff analysis to reach a decision. Throughout this process, humans rely on visual representational techniques to help them organize thoughts, and discover areas of insufficient knowledge. Thus knowledge crystallization is about making ideas clear, bringing them into focus, and discovering new ones. The visual representation of tasks in our method corresponds to the representational form in knowledge crystallization. Problem synthesis in our terms corresponds to synthesis and modification of the schema in knowledge crystallization. A major difference is that all of our visualization methods come with inference engines so that solutions can be automatically generated. In addition, visual cues help humans think and solve problems, not just clarify ideas.

### Intelligent agents

We can find many similar examples in the area of intelligent agents [12,14,22] which treat problem solving. There are mainly two classes of agents: those self-learning agents that watch over the shoulder of a user and become trained to perform the tasks for the user, and those autonomous ones that from the beginning solve problems for users. Our work differs from intelligent agents for two main reasons: the users in our system are also part of the problem solving architecture and the interaction between human and machine is via visualization, not through learning techniques.

### Visualization in multivariate spaces

Visualizing data sets in 2D often employs scatter plots. However, as the data dimension increases, it is more difficult to mediate between the goals of achieving visual clarity and data representation accuracy. For example, when 3D methods are used, data are accurately mapped to points in space, but users have hard times seeing them and navigate in 3D spaces. Several previous works [3,9,18] have explored different methods for rendering multivariate data sets. Our tool, Tradeoff Map, is novel in terms of its unique metaphor to a balance during a tradeoff process.

## ACKNOWLEDGEMENT

This work was sponsored by the Swiss National Science Foundation. We like to thank the reviewers for their valuable comments and feedbacks.

## CONCLUSION

Visualization techniques are beginning to make its way to mixed-initiative user interfaces. The main challenge is the design of visual metaphors to model user's tasks, and prompt them for the right intervention actions. We have shown three visual thinking tools in the configuration domain: **Kaleidoscope**, **Tradeoff Maps**, **and Conflict Resolution Lattice**. Our fundamental contribution is the design of the affordances these tools provide to augment human's reasoning skills in solving problems.

## REFERENCES


[1] Amant, R. S., and Healey, C. G. (2001). Usability Guidelines for Interactive Search in Direct Manipulation Systems, *Proceedings of International Joint Conference on Artificial Intelligence* (pp. 1179-1184).

[2] D. Anderson, E. A., N. Lesh, J. Marks, B. Mirtich, D. Ratajczak, and K. Ryall. (2000). Human-guided simple search, *Proceedings of the National Conference on Artificial Intelligence* (pp. 209-216): AAAI Press.

[3] Beshers, C. and Feiner, S. (1992). *Automated design of virtual worlds for visualizing multivariate relations*. In *Proceedings of IEEE Visualization '92*, pages 283--290.

[4] Card, S.K., Mackinlay, J.D., and Shneiderman, B. (eds.) (1999). Readings in Information Visualization - Using Vision to Think. Morgan Kauffman Publishers, San Francisco CA.

[5] B. Chouiery *et al* (1998). Thoughts on a practical theory of reformulation for reasoning about physical systems. In *Symposium on Abstraction, Reformulation and Approximation*.

[6] Freuder, E.C. (1978). Synthesizing constraint expression, *Communications of the ACM*, 21(11): 958--966.

[7] Ferguson, G., Allen, J. and Miller, B. (1996). TRAINS-95: Towards a mixed-initiative planning assistant, in *Proceedings of the 3rd Conference on AI Planning Systems*.

[8] Horvitz, E. (1999). Principles of mixed-initiative user interfaces. *In Proceedings of Human Factors in Computing Systems*. Pittsburgh, PA, U.S.A., ACM Press.

[9] Kandogan, E. (2000). Star Coordinates: A Multidimensional Visualization Technique with Uniform Treatment of Dimensions. In *Proceedings of the IEEE Information Visualization Symposium*, Late Breaking Hot Topics.

[10] Knuth, E.(1975). Estimating the efficiency of backtrack programs, *Mathematics of Computation*, 29:121--136.

[11] Lalanne, D. (1998). Computer Aided Creativity and Multi-criteria optimization in Design. Ph.D. Thesis No. 1879. Swiss Institute of Technology (EPFL), Lausanne.

[12] Max Metral Yezdi Lashkari and Pattie Maes. (1994). Collaborative interface agents, In *Proceedings of National Conference on Artificial Intelligence*.

[13] Alan K. Mackworth. (1977). Consistency in networks of relations, *Artificial Intelligence*, 8.

[14] Maes, P. (1994). Agents that reduce work and information overload, *CACM*, 37.

[15] Maes, P., and Schneidermand, B. (1997). Direct Manipulation vs. Interface Agents: A Debate. *Interactions,* Vol. IV Number 6, ACM Press.

[16] Melissargos, G. (2000). Interactive Visualization for Resource Allocation Tasks. Ph.D. Thesis No. 2166. Swiss Federal Institute of Technology (EPFL), Lausanne.

[17] Nadel, B. (1988). Tree search and arc consistency in constraint satisfaction algorithms. In L. Kanal and V. Kumar, editors, *Search in Artificial Intelligence*, pages 287--342. Springer-Verlag.

[18] Nielson, G.M., Foley, T.M., Hamann, B. and Lane, D.A. (1991). *Visualizing and modeling scattered multivariate data*. IEEE Computer Graphics and Applications, 11(3):47--55, May 1991.

[19] Pareto, V. (1896). Cours d'économie politique, Technical report, Rouge, Lausanne, Switzerland.

[20] Pu, P. and Faltings, B. (2000). Enriching buyers' experiences: the SmartClient Approach, in *Proceedings of ACM CHI'2000*, ACM Press.

[21] Roth, S.F. et al (1996). Visage: A user interface environment for exploring information, In *Proceedings of Information Visualization*.

[22] Shehory, O. and Kraus, S. (1995). Task allocation via coalition formation among autonomous agents, In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*.

[23] Torrens M., Weigel R. and Faltings B. (1997). Java Constraint Library: bringing constraint technology on the Internet using Java Language. In Working Notes of the Workshop on Constraints and Agents, Technical report WS-97-05, AAAI-97.

[24] Edward Tsang. (1993). Foundations of Constraint Satisfaction, In *Academic Press*.



**Figure 1. A landscape design problem.**

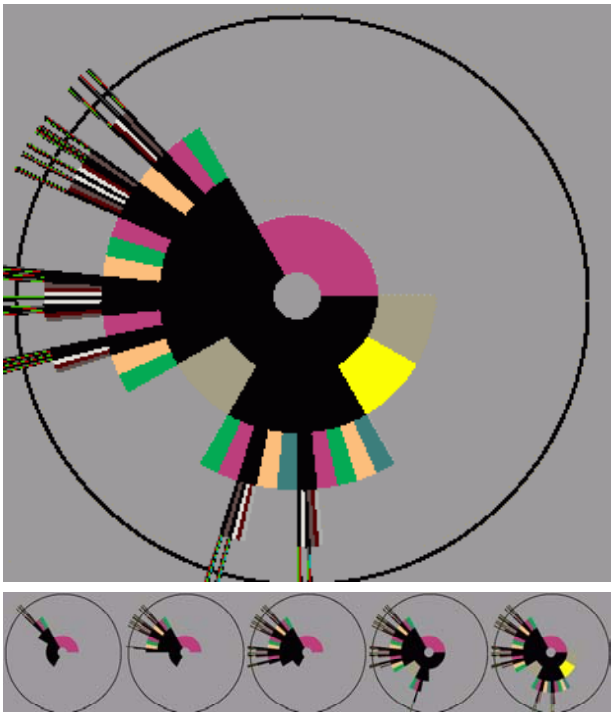

**Figure 2. Kaleidoscope of backtracking search.**


**Figure 3: Color code of constraints.**

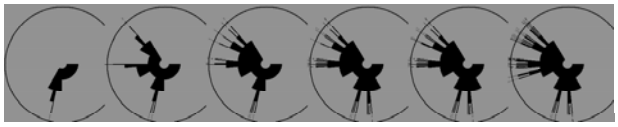

**Figure 4. Knuth algorithms visualized in Kaleidoscope.**

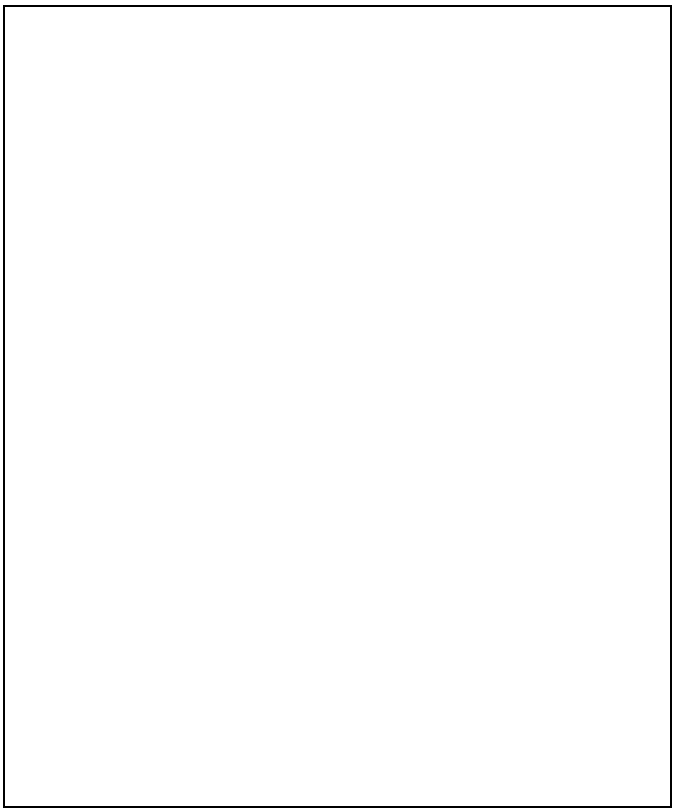

**Figure 5. From top to bottom, parameters' order is more and more optimal in order to accelerate the search.**

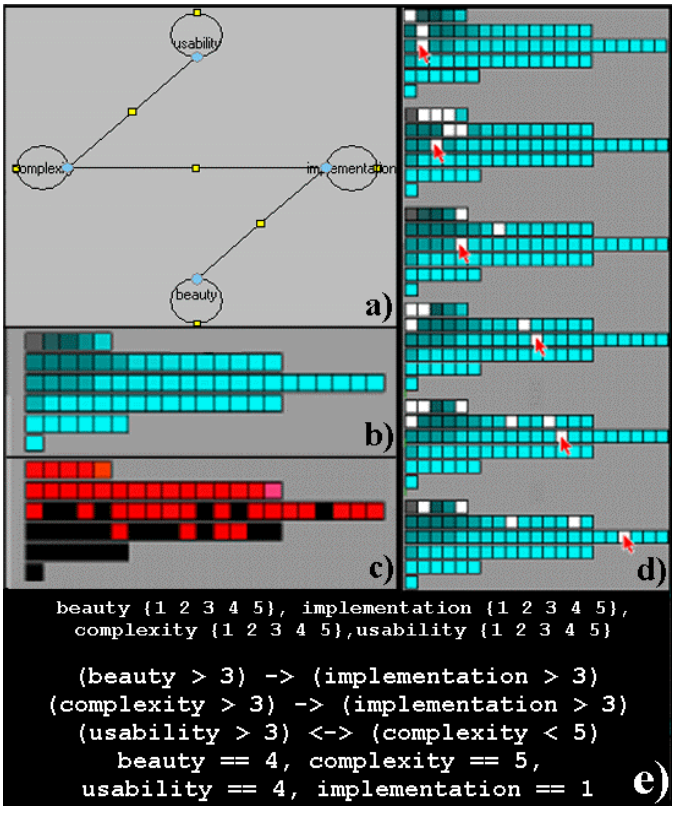


**Figure 6. An over-constrained problem with several conflict sets.**

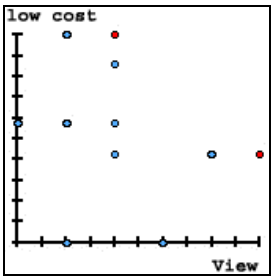


Figure 7. Pareto visualization.

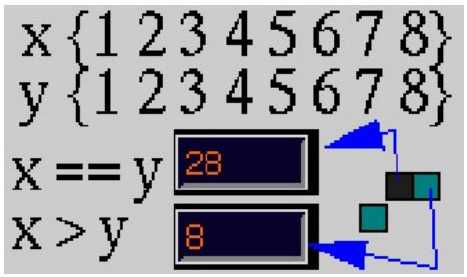


Figure 8. An inconsistent problem.

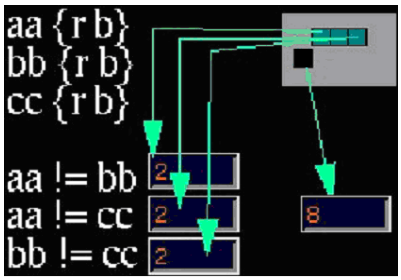


Figure 9. The map coloring problem.

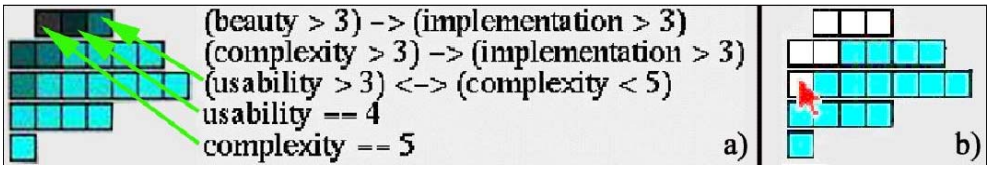


Figure 10. a) Lattice b) its interactive form

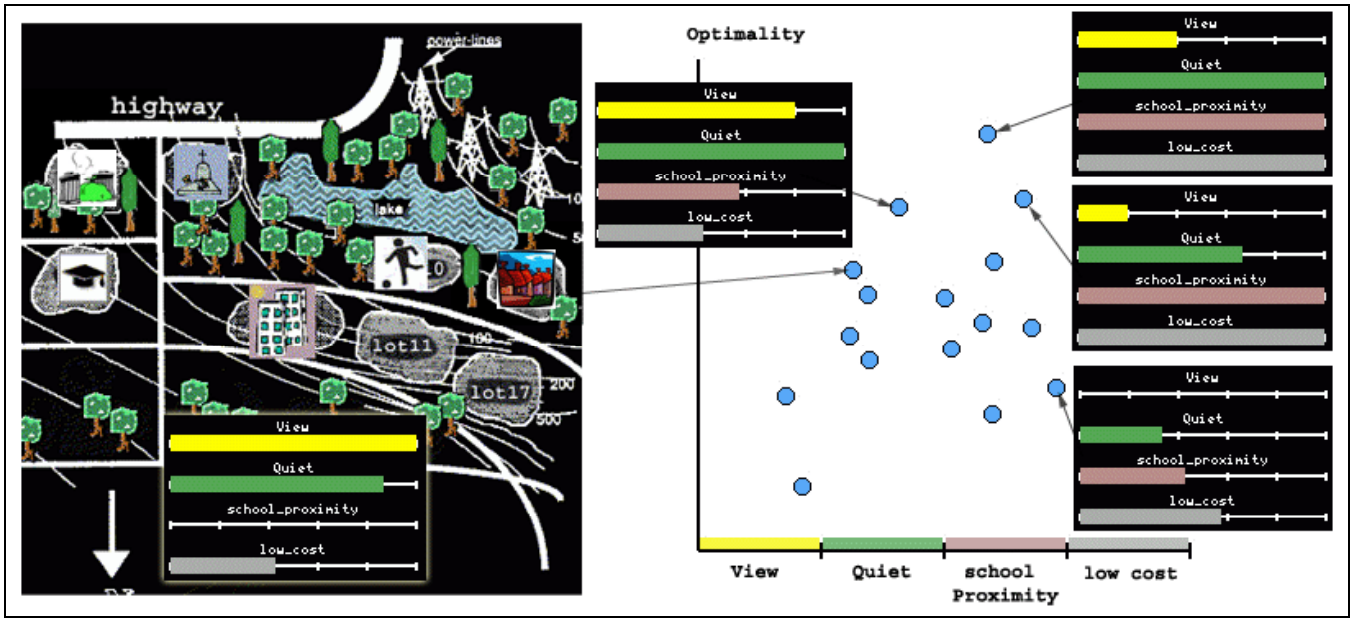


Figure 11. The balance visualization and a selection solution in relation to solution space.

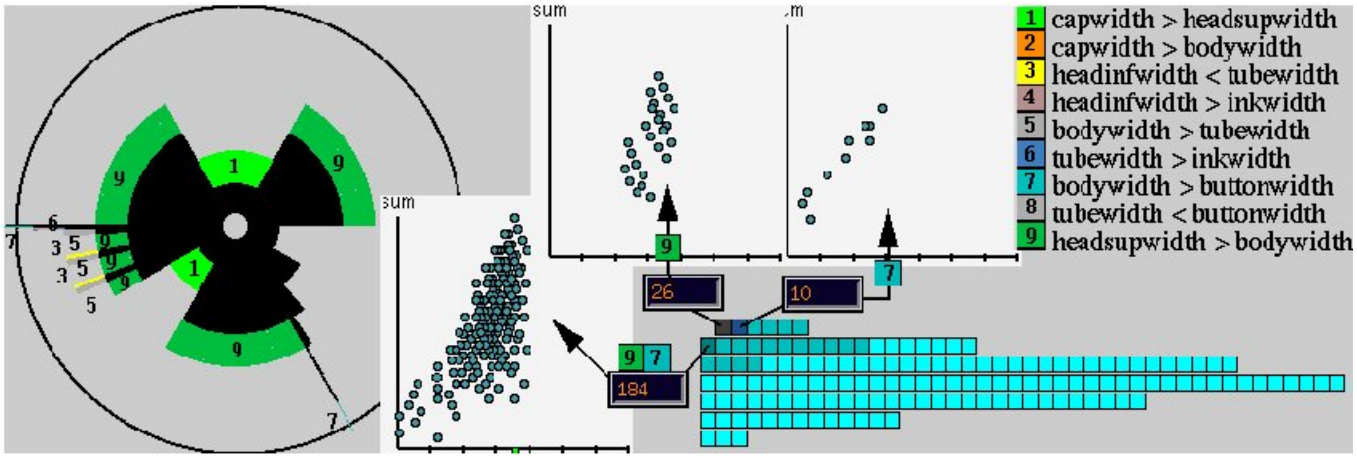


Figure 12. A futile search scenario. The lattice's visualization can be used in collaboration with the tradeoff view in order to browse the potential space of solutions.