Human and Machine Collaboration in Creative Design [1]

Pearl Pu and **Denis Lalanne**[2]

Abstract. Establishing that machines cannot be creative in the same way as humans, we propose a computational model which allows human and machine to collaborate on creative design in a social structure similar to human-human collaboration. We then discuss specific architectural problems associated with the design of such an interactive, collaborative, and intelligent system.

1 introduction

Design is a complex cognitive process which involves creative skills, artistic intuitions, as well as a rich repertory of knowledge. Furthermore design is not description of what is, it is exploration of what might be (Mitchell [8]). According to our experiences with designers, exploration is further characterized by being a process, driven by the desire to change an imperfection, that pushes many limits in the open world. A creative design is one that meets a particular goal and at the same time compromises a set of constraints and current technological limitations.

To free machines from the predefined world or to enrich them with interactions from nature (for a more detailed article, see [14]), we must allow humans to help them in a similar way that humans benefit from the calculation and data processing power of machines. In this paper we propose an approach which integrates several previously built intelligent design systems in an open and distributed architecture. Most importantly, this architecture will facilitate human-machine collaboration and human and machine creativity to co-exist and co-contribute to the creative design process. An open architecture contradicts the traditional algorithmic nature of software and organizes its functionalities at the cognitive level of task distribution instead of the control level of task performance. We propose to use the method of software agents to design and develop such an architecture as depicted in Figure 1. Automatic and semi-automatic agents collaborate with human designers, and they cooperate with each other *seamlessly* to achieve a given goal. Some agents are reflections (mirrors) of the cognitive inner workings of a person while others are computational because they take care of detailed design works such as constraint processing, search, and data visualization. Many of the agents of this architecture have been developed such as the case-based reasoning, constraint processing, and multimedia browsing units. Here we discuss the challenging issues we encountered in the development of this system: 1) what contributions human and machine can each provide to the creative design process, 2) what are the mirroring and processing roles of agents, and 3) what will be the collaborative technique which allows contributions from humans and machines to take place in the creative design process.

[1] Supported by the Fonds National Suisse de la Recherche Scientifique

[2] Laboratoire d'Intelligence Artificielle & Robotique, Institut de Microtechnique / DMT, Swiss Federal Institute of Technology (EPFL), MT-Ecublens, 1015 Lausanne, Switzerland, pu@imtsun9.epfl.ch and lalanne@imtsg10.epfl.ch

© 1996 P. Pu and D. Lalanne
ECAI 96. *12th European Conference on Artificial Intelligence*
Edited by W. Wahlster
Published in 1996 by John Wiley & Sons, Ltd.

Although the system contains components that help humans perform creative idea generation using techniques such as brainstorming and synetics, in this paper we focus on the problem of performing multicriteria optimization and decision making under multiple objectives. This task is known to be difficult and finding solutions to such a problem is considered to be a typical creative process. We will illustrate examples that can be treated in our system, COMIND, [3] along with some descriptions of implementation issues.

2 Human cognitive prosthesis and the role of agents

Humans live in a stimulating and dynamic environment which allows them to be creative and productive. But there are limitations of humans especially in terms of cognitive and mental power. For instance, it is currently known that in short term memory, there are only 4-7 conceptual chunks we can hold [18]. We often have trouble visualizing a set of vectors in three dimensions, let alone doing so in higher dimensions. With the fast growing technology in computer industry, we have become accustomed to using machines as our cognitive and computational prostheses. That is, we rely on their power to help us accomplish tasks. For many of us, it is hard to write well a letter without typing it in a word processor because traditional paper and ink do not allow us to copy and paste easily. There is no spelling checker provided by a piece of paper.

Helping humans to perform word processing may be a trivial task, but helping them solve creative design problems is not. We have to consider a number of issues such as the ergonomics of the system, its architecture, and the social issue between human and machine as collaborators and co-contributors. Our main approach is to use the idea of software agents [16] as the interface between human and machine. Sociologists maintain that the interaction between human and machine is mainly a social one [9]. Furthermore, the anthropomorphism brought by human-like agents is superior than other metaphors such as object-oriented programming. Finally commonsense tells us that it is much easier to tell my agent what I saw today on the subway coming to work than to my program named inspiration.lisp.

In further designing such a system, we use theories and models from three main domains: human-computer interaction, ergonomics of software design, and artificial intelligence. A basic sketch of the architecture of the system is the following: a working area is provided for designers to treat various aspects of the design problem. Agents will appear as iconized caricatures in the working area and they consist mainly of two types: reflective and computational agents.

Reflective agents (appear on the last row in Figure 1) are fully autonomous and their main goal is to serve as a cognitive map of the user. They will show for instance that the designer has just done a

[3] COMIND abbreviates for Computer-aided Creativity for Multi-criteria Optimization IN Design

session of brainstorming and now is instantiating design parameters. Computational agents (appear on the two first rows in Figure 1) process inconsistencies in design space, generate partial and complete solutions, retrieve design cases, help designers visualize a set of competing solutions, and propose tradeoffs. These agents are semi-automatic in the sense that they are only called for at the request of users. They could become more autonomous once a trust period has been established between the user and the agent. However, we minimize the communications among the agents in order to simplify the protocols used in this architecture.

3 Interfacing agents and their roles

In the next several sections, we will describe a framework based on interfacing agents which operate in a design environment to support innovative and creative design.

3.1 Reflective agents

Restricting our focus to the conceptual design domain, we have defined our agents and their goals in the following manner. The reflective agents serve two purposes in the creative design process. For novices like students, these agents act as a guide which reminds the students of the important processes to take during conceptual design. For example when a student is performing brainstorming, he must be sometimes reminded not to criticize his design ideas at this moment. The goal is to produce as many ideas as possible. An agent can observe the behavior of the person and notice the speed in which ideas are produced. If the speed is below average, the agent might guess that the person is evaluating his design ideas and thus doing criticizing. The agent can signal this to the user not to stall on idea production. Another useful reflective agent is one which suggests some related or contrasting ideas when users put down theirs. This method, called synetics, is often used to stimulate ideas by combining two unrelated ones. But we believe that similar ideas can also be useful in synetics. In most cases, humans perform synetics among themselves by taking terms to propose ideas and proposing related or contrasting ideas. But agents can very well accomplish this task so that designers do not have to always find partners in brainstorming sessions.

The other goal of the reflective agents is to serve expert designers. They are useful in showing designers the reasoning path that he/she took to arrive at this point of the design process. In a small window in the corner of the working area, a map can show for instance that a person has done brainstorming, synetics, evaluation using the house of quality method, and now is working on multicriteria tradeoff. Since each person has a different pattern to perform the conceptual design, over a longer period of time, users may observe their own behavioral pattern. They may choose to either follow similar patterns or purposely change the patterns using this visualization tool. This kind of cognitive map [2] has been found useful in prompting humans in exploring unknown territories. Since it is critical that designers do not slip into his usual way of thinking, the cognitive maps can serve as indications of the current attention memory [1] of the human so that if he notices it, he may choose to change it.

These agents are called reflective because they do not themselves perform the design per se, but help reflect, organize and give feedback of the ongoing design process. But in doing so, they help designers achieve better design.

3.2 Computational assistants

Contrast to reflective agents, computational agents actually accomplish tasks for human designers. Their main role is to take away some of the computational burdens from the human and provide visualization to facilitate tradeoff in higher dimensional spaces.

One important element of the design problems we address is making decisions and tradeoffs both before and after the generation of design solutions. We consider mainly two cases. In one, the solution space is sparse and often null because the problems are over constrained. Thus humans will help the system find solutions by adding parameters or deleting constraints in order to relax the solution space. This process, called mutation, is central to many design problems where creativity from humans is required in order to change the current representation of the design problem, either by changing the parameters or relaxing constraints. In the second case, there are many solutions which can potentially cause problems in terms of generating all of them or generating the good ones. Furthermore, after designers have narrowed in on a dozen of them, it is hard to evaluate them because they are competing, or Pareto optimal [12]. This problem, called multicriteria optimization, can be better handled if machines provide good visualization tools in order for humans to analyze and perform tradeoffs.

More precisely, the computational agents currently implemented in our architecture are named PARAM-DEF, SOLVE, PARETO-VISU, SHOW-CASE and TRADEOFF and their main functionalities are as follows. PARAM-DEF allows users to define and represent the current design problem in terms of parameters and constraints. SOLVE is an agent, which when brought over the PARAM-DEF area, will search the solution space. In case it produces a null set, meaning that the problem is over constrained, users will be asked to either change the participating parameters or modify the constraints. SOLVE can be brought in to re-generate solutions. When a problem admits many solutions, users can ask for a neighborhood of solutions and ask PARETO-VISU to plot the Pareto space along a set of evaluation criteria (see Section 3.3) in order to visualize it. Users can propose additional criteria in case there are many Pareto optimal solutions. TRADEOFF can then offer to finalize on the design solution by finding a good balance of objectives. SHOW-CASE is a case-based system which targets mainly at novices in teaching and stimulating them to be creative by showing how expert designers achieve creativities.

We present two scenarios to illustrate how our agents handle the problem of mutation and multicriteria evaluation.

3.2.1 Mutation

In the first scenario, the goal is to design a wrist watch which is also an electronic agenda. The first step is to have users define the main parameters of such a product such as the dimensions, the weight, the material used for the watch base, and the resolution of screen display of the watch in PARAM-DEF as shown in Figure 2. Secondly, designers provide a rough range of these parameters and define constraints that govern these parameters. For instance, the largest surface dimension, which can be the length or diameter depending on whether the watch is square or round, should be within 4 centimeters. The resolution is a function of 10 times the total square centimeters of the dimension. The resolution is currently set to be greater than 300 pixels. PARAM-DEF then formulates the problem as a constraint satisfaction problem [3] where both continuous and discrete design parameters are represented as variables. Each variable has a range of possible values called the domain. Finding a design solution is to generate an instantiation of all variables whose chosen values do not violate any of the constraints.

Difficulties arise when the problem is overconstrained as it is in our case. The resolution constraint is violated because the dimension is too small. That is, for a wrist watch, the viewing area is not satisfactory. At this moment, the system will suggest the designers the following two things. They can relax the constraints or perform mutation. If relaxation of constraints is chosen, they are further given a list of constraints that have caused the null space. In this case, the dimension and resolution constraints are highlighted in the PARAM-DEF's constraint area, signifying conflicts. Or, the designers are signaled to add, modify or delete parameters in order to change the current formulation of the problem. Deciding that he will change parameters, this designer thought about how to achieve a compromise so that one can have a larger display area without increasing the dimension of the watch. He also came upon the thought of an umbrella and its foldable feature. Thus he suggests that the wrist watch display area be foldable and furthermore should have three faces when it is folded out as shown in Figure 3. This gives a resolution of 160 times 2 which is more than 300 pixels.

Novice designers are not likely to perform mutations very well. We don't believe that such magic intuitions or inspirations can be automated either. However, a case-based reasoning system [17, 20] will help stimulate the creativity of novices by providing creative design scenarios via the use of video, animations and text. Our existing system, CORINTH [13], can store design cases in either linear, hierarchical, or linked list structures. Currently we plan to store 10 to 20 creative design scenarios in the SHOW-CASE agent and when design students ask to see one or several of them, we illustrate them in rotational order. There is no adaptation involved. The main purpose is to show users the types of creative possibilities they can integrate into the system.

3.3 Multicriteria evaluation

When a design problem has many solutions, it is important that these solutions can be evaluated along a set of criteria and a best one be chosen. Quite often solutions are found to be competing, called Pareto optimal [12, 15, 10]. That is, some solutions are good in certain aspects, but not in all and thus cannot dominate others. Creativity is required to either discover new criteria to narrow down the Pareto set or reevaluate the solutions in a tradeoff. This corresponds to the "seeing things in a new light" that we do in life.

The following scenario from Navinchandra [10] can best illustrate this type of problems. For the purpose of simplicity, we present a non-design scenario. But our techniques are mainly applied to the design domain.

In finding apartments on arrival to graduate school, a person encounters four choices. Given two criteria, noise level and the size of the apartment, he maps these solutions according to the criteria in the graph shown in Figure 5. Three solutions are found to be Pareto optimal and furthermore they are about equal-distanced from the given criteria. In such situations, the decision maker often lacks the motivation to take any solution. Further exploration with the real-estate agent brought him a fifth solution, an apartment on Brookline street which is currently dominated by other solutions as shown in Figure 4. However, at the same time, he discovered a third criteria to evaluate the solutions, the distance to public transportation systems. Being very close to the subway lines, this Brookline apartment got pushed to the Pareto surface. Now the user is ready to do tradeoff. That is, given these four possibilities, he is to reason which criterion is the most important for him. Since the last solution has a strong optimality with respect to the transportation criterion, it was chosen as a final choice.

Again we believe that human-computer interaction plays an important role in multicriteria evaluation. The additional criteria (emergent criteria phenomenon by Tomlin [19]) are suggested by users during the design exploration. Machines, however, will stimulate the human in criteria emergence by showing design cases or simply providing a multimedia browser to show a photo or a 2D sketch of an artifact. Machines will also provide visualization capabilities in the criteria space. Solutions are mapped in a three dimensional space as shown in Figure 6a. A Pareto surface (Figure 6b) can be constructed to illustrate dominating solutions. Either dominated solutions are thrown out or users can try to add additional criteria so that some of them can climb onto the Pareto surface. Furthermore, any node can be selected and then the mouse can control the viewing area so that the selected solution is compared with other solutions during tradeoff.

3.4 Algorithms

The SOLVE agent uses an efficient consistency-based constraint satisfaction algorithms [5] to prone the search space. For over-constrained problems, this method can quickly indicate which design parameters and constraints are the most restricted ones. If the search space is not null, then SOLVE uses a simple backtracking algorithm to find one or several solutions. It further looks for a neighborhood of similar solutions. These solutions are then compared in the Pareto space according to a set of criteria given by the users. The users can add new criteria, perform tradeoffs, or select a solution. In selecting a solution, the user can also ask SOLVE to change one of the value assignment explicitly, but keep the rest of the assignments. In this case, SOLVE uses the minimum conflict repair algorithm [7] to satisfy the user's request while attempting to resatisfy other variable assignments.

Our main method is in contrast to the implementation of CYCLOPS (Navinchandra [10]) where A* algorithm is used to guide the search towards one solution and along the way Pareto optimality is used to heuristically guide the successful search path. The fact that such a heuristic rule is applied uniformly to all partial solutions prohibits the discovery of unusual partial solutions which seem unpromising for a while. The interactiveness of our approach, on the other hand, allows designers to see all kinds of solutions and then compare them on a set of dynamically determined criteria. If users are dissatisfied with the current solutions, they can abandon these neighborhoods and opt for others. Though it seems that we allow designers to “shoot in the dark” in the process of discovering creative designs, this kind of serendipity is part of discovery.

4 Design of Engaging Interface Agents

One of our hypotheses established in this paper is that humans and computers must meet each other on the screen to benefit from each other's strengths in creative design process. We claim that engaging interface agents can stimulate designers' creativity and increase the output. After addressing the architectural and technical issues of such a system, we discuss the artistical efforts that are required in the design of such a system and its interface. Unlike domains where evaluation can be measured more analytically, such as the O(n) notion for expressing the complexity of algorithms, the interface design is hard to evaluate with scientific scales. Fortunately there is a rich body of theories and guidelines mainly found in HCI (human computer interaction) on how to design and implement interfaces. Without reviewing the whole field, we will discuss some of the principles we have adopted in our design.

The use of agents achieves three goals for the design of the current architecture of COMIND. As stated earlier, the interaction between machine and human is basically social. Thus agents appeal to designers because they have characters which designers familiarize quickly through usage. Second, they help designers classify all the knowledge and rules they need to remember into a manageable set of chunks. Finally, an agent-based architecture is distributed which facilitates the computation of various design tasks in a non-algorithmic structure. That is, a designer is not forced to do brainstorming followed by parameter definition or vise versa. He is free to choose the tool that fits the most with his current state of mind. Such an architecture also evolves as more agents are added to make the functionality more extensive.

Many works on intelligent agents employ the use of learning methods to implement agents which observe users' behavior. After a period of time, these so-called "watching-over-the-shoulder" agents will automatically perform tasks for users. This approach is well suited for domains such as electronic mail sorting [6], especially for users who subscribe to interest groups and receive an enormous amount of mail from the same originator. However, Norman [11] points out that fully autonomous agents can be overwhelming and annoying to users. Many studies point out that it is still too early for humans to completely trust machines. Most of us would like to know a little bit about how machines solve a certain task before we entrust them and launch them freely. This is true especially in the beginning of the user's experience with a program.

Our agents, except for the reflective ones, are never self-imposing. In the beginning, they are called for action by the user through a rather tedious invocation procedure, like pointing to the iconized agent, pulling down a menu and selecting a function. This prohibits the users from invoking them on a hasty basis. In other words, they have to really want to call the agent before it becomes available.

Instead of using a measure to establish a trust between users and machines as described in Maes [6], we use a simple mechanism of associating key bindings with the invocation of agents. Humans are known for routinizing tasks such as typing. Once routinized, executing the task becomes subconscious and thus does not require any cognitive load. Thus instead of building agents which automatically establish a trust with the user, we let the users capture such trust by routinizing the task.

5 Summary

Our main hypothesis in this paper is that the interaction between human and machine has an important role in computer-supported design, especially when creativity is required. We thus have proposed an open architecture where design is not viewed as an algorithmic activity, but a collaboration between a user and a set of software agents. The role of creativity and drawing inspirations from nature falls mainly upon the human while a number of computational and visualization tasks are provided by the machine. This architecture is not limited by the closed world assumption of most artificial intelligence systems. It evolves because human experiences evolve. Furthermore, as more computational modules become available, they can be further added to the system to support more design activities.

In his book [10], Navinchandra did a great job outlining and explaining innovative design and the role of computers. Much inspiration of our work came from there. However, instead of using users as a judgemental factor, we go much deeper into the computer-human interaction issues. Every key step of the design process is visualized. Users and agents collaborate all the time. Thus COMIND is more humanly whereas CYCLOPS is more automatic. COMIND evolves and creates more new design cases whereas CYCLOPS tends to debug designs always using the existing cases.

Finally the handle of agents and the protocol between human and agents takes along a similar line of philosophy. That is, we maximize the natural interaction between human and machine and minimize guess work.

We would like to thank George Melissargos for technical support, and Professors Clavel and Popovic for their encouragement of our work and insights on design methodology. Finally, we thank Boi Faltings for giving us useful comments on the multicriteria optimization and tradeoff problem.

REFERENCES

[1] Allan Baddeley. *The Human Memory, Theory and Practice*. Lawrence Erlbaum Associates, 1990.

[2] Margaret A. Boden, editor. *The Creative Mind*. Basic Books, 1991.

[3] Eugene C. Freuder and Alan K. Mackworth (Guest Editors). Special volume on constraint-based reasoning. *Artificial Intelligence*, (1-3), 1992.

[4] Patrick Hayes and Kenneth Ford. Turing test considered harmful. In *Proceedings of International Joint Conference on Artificial Intelligence*, August 1995.

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

[6] Pattie Maes. Agents that reduce work and information overload. *Communicatons of the ACM*, 37, July 1994.

[7] S. Minton, M. Johnston, A. Philips, and P. Laird. Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, 58:161–205, 1992.

[8] William J. Mitchell. Introduction: A new agenda for computer-aided design in the electronic design studio. In Mitchell McCollough and Eds. Purcell, editors, *The Electronic Design Studio*. MIT Press,Cambridge,Mass., 1990.

[9] Clifford Nass, Jonathan Steuer, and Ellen R. Tauber. Computers are social actors. In *Proceedings of CHI'94, ACM*, April 1994.

[10] D. Navinchandra. *Exploration and Innovation in Design*. Springer-Verlag, New York Inc., 1991.

[11] Donald A. Norman. How might people interact with agents. *Communicatons of the ACM*, 37, July 1994.

[12] V. Pareto. Cours d'economie politique. Technical report, Rouge, Lausanne, Switzerland, 1896.

[13] Chris Pepin. Efficient design systems using case-based techniques. Technical report, University of Connecticut, 1993. Master Thesis.

[14] Pu Pearl and Lalanne Denis. Human and Machine Collaboration in Creative Design. Technical report, Ecole Polytechnique Federale de Lausanne, Switzerland, 1996.

[15] Antony D. Radford and John S. Gero. *Design by optimization in architecture, building, and construction*. Van Nostrand Reinhold Company, New York, 1988.

[16] Doug Riecken. Introduction: Intelligent agent. *Communicatons of the ACM*, 37, July 1994.

[17] Christopher Riesbeck and Roger C. Schank. *Inside Case-based Reasoning*. Lawrence Erlbaum, 1989.

[18] Thomas P. Moran Stuart K. Card and Allen Newell. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, 1983.

[19] D. Tomlin. Personnal communication. Technical report, Harvard University, 1986.

[20] Linda M. Wills and Janet L. Kolodner. Towards more creative case-based design systems. In *Proceedings of National Conference on Artificial Intelligence*, August 1994.

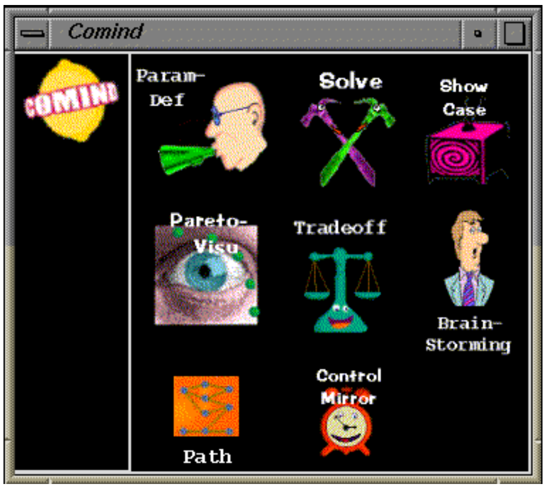
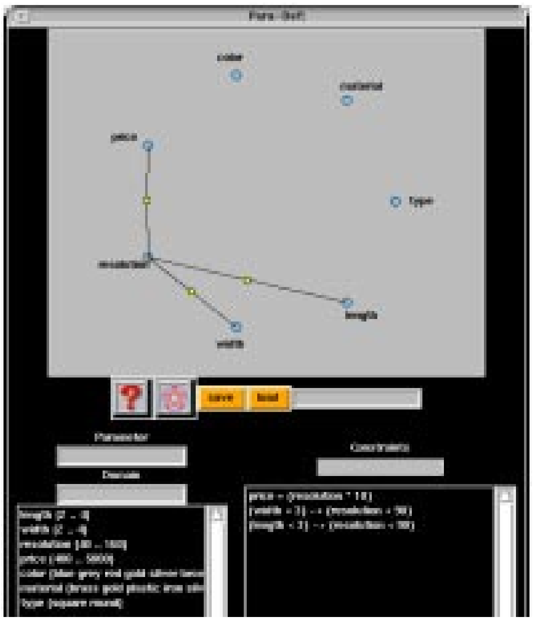


Figure 1. Software agents as appeared in COMIND



```
length [2 .. 4]
width [2 .. 4]
resolution [10 .. 100]
price [400 .. 5000]
color [blue grey red gold silver brow
material [brass gold plastic iron silv
type [square round]
```

```
price = (resolution * 10)
(width = 1) --> (resolution < 50)
(length < 1) --> (resolution < 50)
```

Figure 2. Definition of a design problem

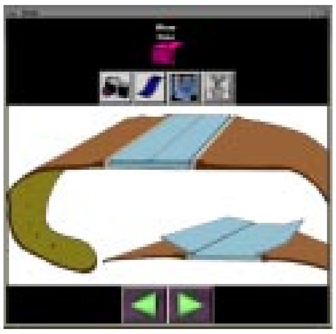


Figure 3. An agenda watch

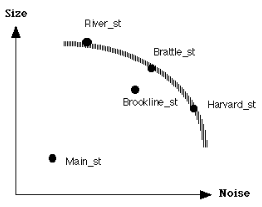


Figure 4. The nodes correspond to the street where the apartment is located.

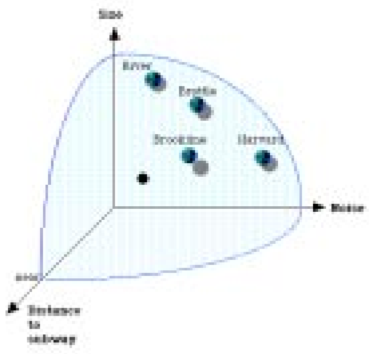


Figure 5. The larger nodes are located on the Pareto surface.

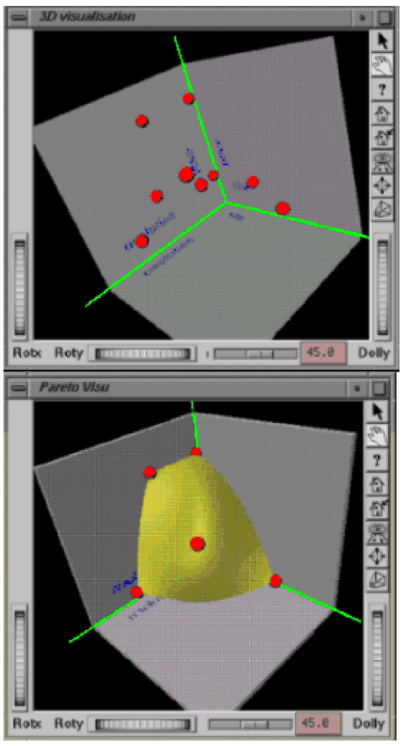


Figure 6. a) visualization of solutions b) visualisation of Pareto surface