Solving complex problems with computational and interfacing tools

D. Lalanne, G. Melissargos, P.Pu, {lalanne,melissargos,pu}@imt.dmt.epfl.ch
Ergonomics of Intelligent Systems and Design, Microengineering Dept, Swiss Federal Institute of Technology

1. Introduction

Solving complex problems requires a considerable amount of computational power, intuition and human skills. We are going to present two specific cases in the area of design and scheduling. They are both complex problems that can be solved with the help of the same problem solving architecture that allows human and machine to cooperate in an effective manner.

1.1 Computer aided creative design

As design has become an important preoccupation according mainly to economic reasons, Computer-Aided Design (CAD) is now a central research in computer science. Many tools have been developed in order to support construction like the well-known AutoCAD.
However, creative design is a more complex cognitive process because creative thinking is involved in this step. Creative design mainly happens in the conceptual step of design. Designers do not like conceptual design step because it is intellectually difficult; it requires a very high attention because the number of criterias is often high, because the goals are often blur, because some internal conflicts often exist and because it requires tradeoffs. However, it is estimated that 85 % of decisions, that allows to elaborate a final product, are taken during conceptual design. Furthermore, reducing the time between design and elaboration would be the guarantee that a new concept reaches to an innovative product.

Creative design requires both the studies of creativity and design. Nevertheless, those two processes of thinking are both ill defined. The former is tackled in numerous domains and its sources are not yet surely defined. It is needed in several areas like art, problem solving and design and also in everyday task for personal problem solving. On the other hand, design is slightly better characterized; some forms of design are formalized and a lot of taxonomy has been produced. However, according to the growing complexity of our world, design is becoming a more difficult task. Every day we are confronted with design results and more and more people are involved in design tasks. Thus, tools that support designers are necessary in order to increase their skill and to make them more creative and more concerned about the quality of their design.

We have found two main types of approaches in creative design. In the first one, researchers try to make machine creative by either using neural networks, case-based reasoning, genetic algorithms, or prototype-based systems ([5], [7]). In the other approach, they try to find the important features of human creativity in order to support it [3]. A CAD system supports creative design if it allows the user to define novel designs and is creative if it discovers new designs by itself.

We do not believe in a clear taxonomy of types of design because their boundaries are blur and because it is generally impossible to classify a specific design. In most of the cases, a designer has to deal with either routine, innovative or creative cognitive processes; it is the level of complexity of each step that distinguishes a type of design from an other. Our taxonomy comes from the observation of the process of creative design. In our opinion, creative design comes from two types of cognitive process: tradeoff and break-through process. Those two kinds of process come from another distinction between two extreme types of problem: over and under-constrained problem. In under-constrained problem, too many solutions are possible. People need to do tradeoffs to find the optimal solution. However, according to the large number of criteria, choices are difficult. In over-constrained problem, there is no solution. People have to find conflicts and to solve them. It requires two things: consistency checking and creativity to enlarge the space of solutions and thus to avoid the conflicts. The limitation of human memory is a problem. Dealing with about a hundred of constraints is difficult and finding the escape requires ingeniosity; the designer has to break-through the current space of solutions.

1.2. Scheduling systems

One of the most frequent application areas of informatics is scheduling. Scheduling problems come in all shapes and sizes. Our work concentrates on reactive schedulers as opposed to predictive ones. Given a global schedule and any related information, the reactive scheduler is responsible of accommodating any requested changes that may affect either the timetable of tasks or the allocation of resources to tasks [6]. The flight rescheduler presented in this article is concerned with the reallocation of resources to the scheduled tasks. The resource allocation problem in reactive scheduling is a hard one to solve. It is often bound to external, to the schedule, events that are hard to be anticipated in advance by the system. Even if the rescheduler succeeds in taking into account all the needed parameters it may end up with an under-constrained problem that produces a large number of solutions, thus converting the initial combinational problem to a discrete optimization one. In both cases the active involvement of an expert human operator is essential. The human expertise, knowledge and reasoning have to be incorporated into the problem solving process. There are already attempts to blend interactivity with artificial intelligence techniques in order to solve scheduling problems [2]. The user is actively involved into the process of problem solving and aids into the performance of the underlying computational engine. He either introduces additional parameters, constraints and rules or he controls the performance of the supplied computational tools. But this is only the one direction from the bi-directional communication link between the human intelligence and the machine's computational engine. In order to succeed in an effective interaction model the machine has to respond in a meaningful way that will help the user's understanding and reasoning process.

For example, in the case of an under-constrainted problem in rescheduling, the system may return a set of possible solutions. Each of the solutions, if applied to the schedule, returns the schedule into a new equilibrium. But which of the solutions is the optimal one? How can the user decide? He has to go through all of them, examine them, reason about their applicability, compare them with each other and finally select the most appropriate one. Each one of the solutions is a set of new resource-to-task assignments. Now we can easily imagine the poor scheduler's operator faced with a long textual list of 200 solutions. It is obviously impossible for him to proceed. Instead, the system should respond with a comprehensive pictorial representation of all these solutions. By encoding in one display all of the quantitative and qualitative characteristics of the solutions we facilitate the processing of information thus leading the user to easier and more efficient decision making.

In our work we try to propose a computer supported framework for scheduling problem solving where interactive visualization enables a better cooperation between the user and the machine. We consider interactive visualization as the main vehicle for seamless human-machine coordination in problem solving for resource allocation. The reasoning process, within the proposed framework, emerges from the continuous collaboration of the machine intelligence and the user expertise.

2. Presenting the two systems at a glance

2.1. COMIND: Computer aided creativity and multicriteria optimization in design

To free machines from the pre-defined world or to enrich them with interactions from nature, we must allow humans to help them in a similar way that humans benefit from the calculation and data processing power of machines. COMIND's design environment allows humans and computers to collaborate and cooperate, sharing their cognitive and computational resources. This human-centered design environment offers many software modules as servicing agents shown as small drawings in Fig.1. These agents help humans define design problems, visualize solutions, look for compromises in design tradeoffs, and discover new design variables. They also reflect the design process, support brainstorming and show design cases.

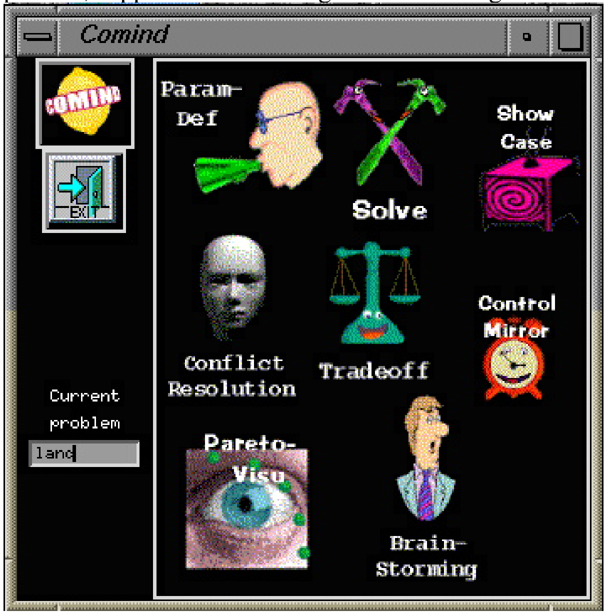


Figure 1: The Comind main entrance.

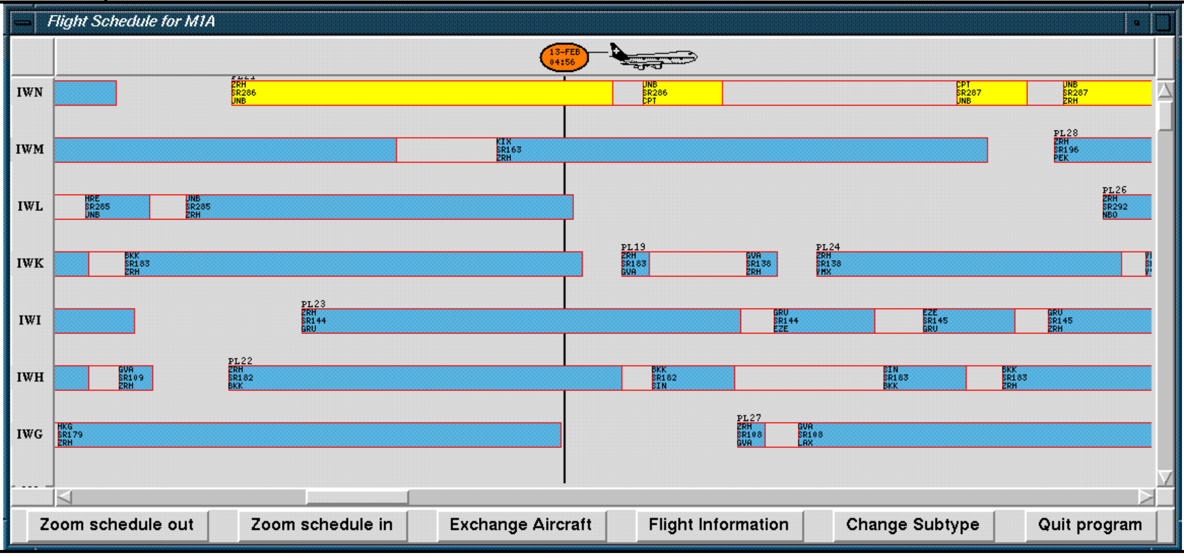


Figure 2: The flight rescheduling system.

2.2. The flight rescheduling system

Flight rescheduling can be viewed as a resource reallocation problem. The aircraft is the resource and the flight is the consumer. There are different kinds of airplanes with such attributes as the number of seats, the occupancy per flight assigned to and a series of constraints and rules, like a set of permissible airport destinations and origins, maintenance ground times, etc. The same holds for the flights which are characterized by constraints and attributes like the departure and arrival times and places, the flight and ground interval times and the types of aircrafts permitted to be flown by. There are more data objects like the airports, which encapsulate a number of interdependencies and characteristics that have to be taken into account when rescheduling. Various artificial intelligence techniques are employed so that a complex information system is built, namely the flight schedule. This complexity is hidden through a representational abstraction of the flight schedule (fig. 2). The flight schedule has been preplanned in a wise fashion and it is designed to meet most of the existing requirements whether they come from the technical, economical or political world of an airline.

There are though, quite often, cases that dictate changes into the current allocation scheme. A good example is the need for aircraft upgrading due to overbooking of a flight. Since all of the aircraft are all of the time assigned to specific tasks the rescheduling system has to shuffle some of the current assignments so that both the request for upgrading can be satisfied and the whole schedule meets the demands as planned

3. A common architecture

To facilitate the study of complex problem solving systems we decompose the process of solving in three phases. In the beginning, the system constructs the necessary information workspace with which the user can interact. The goal of this stage is to better define the problem in itself and construct the appropriate query. The user can interact with the information workspace, explore it looking for aspects of the problem he is not aware of and finally introduce any parameters, rules and constraints that are not already encoded into the system. After the submission of the user's query to the computational engine we pass to the second phase, that of searching for solution. The search engine looks for solutions that satisfy both the user's demands and the predefined constraints. The user can monitor the progress of the search or he can even interact with the search engine in itself. With the final outcome of the search phase we enter into the third stage of the problem solving process. Depending on the result the user may be presented with no solution or multiple ones. If no solution has been found then the problem was over-constrained so the system and the user have to reevaluate the problem, find the violating constraints or the conflicts and relax them. When too many solutions are found, the problem is under-constrained and the user, with the help of the system, has to evaluate them and select the most appropriate one according to a set of tradeoffs. We must emphasize that often, in several application areas, there is no clear cut between the three phases. On the other hand, they intermingle according to the needs of the problem solving process. For example, in a case of network traffic rescheduling that we have studied, it is possible in the preprocessing phase to reach the optimal solution. This is done via an interactive step by step processing of the problem space.

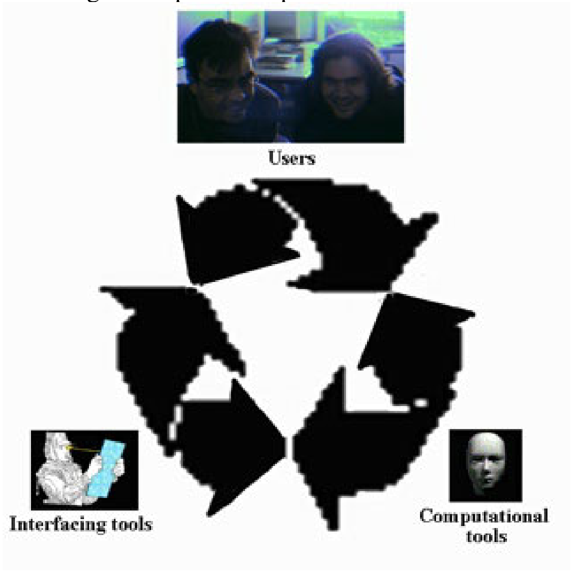


Figure 3: A triangular open architecture.

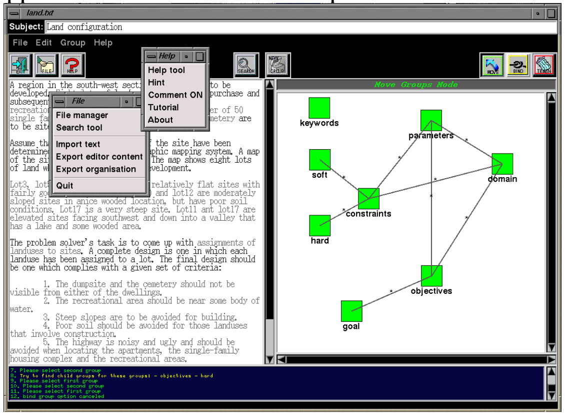
It becomes apparent from studying the three problem solving phases that the computational tools and the human user have to work in a synergy throughout the process of problem solving. In order to achieve an effective cooperation there must be an efficient communication mechanism in place. Our experience has showed us that interactive visualization may be the perfect candidate to reach this target. Machines are powerful for their computational skills and humans are particularly strong for their intuitive and creative qualities. In order to make them able to cooperate, we use a triangular architecture containing the user, computational tools and interfacing tools (fig. 3). Each item can communicate to an other one. The user can profit of the machine calculation power and the machine can profit of the expertise and of the intuition of the user. The interfacing tools serve as an interactive workspace allowing human and machine cooperation. The user can use a same interfacing tool in order to communicate with several computational tools or use a same computational tool with different interfacing tools in order to have different points of view of a same calculation. Furthermore, simple interactions between the computational tools and the users can take place in a straight forward manner without any intermediaries.

However, complex interactive visualizations are more demanding and they require the presence of special interfacing tools. This independency between actors provide an open architecture, a kind of modularity allowing all the combination of tools. Essentially, the interactive visualization acts as a bi-directional link between the artificial intelligence and human intelligence. The benefactor of such an efficient communication is the problem solving process.

3.1. First phase: Defining the problem

3.1.1. In Comind

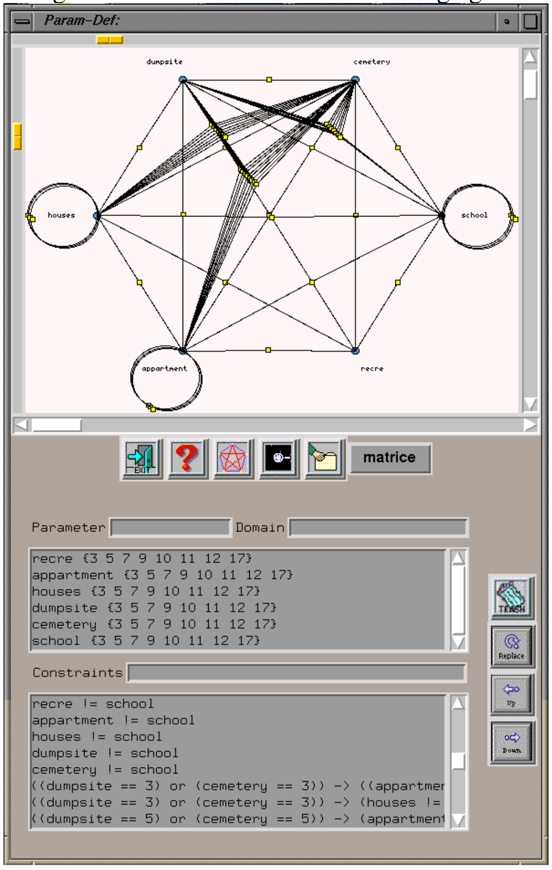
In order to define the problem, the designer is first using the Brainstorming workspace (fig. 4). In this step, the designer can type freely ideas about the problem in an editor. The Brainstorming assistant proposes analogies with older sessions; the goal is to produce as many ideas as possible. Other helps are provided in order to support the structuration of the problem. The aim of the Brainstorming assistant is both to keep a trace of initial ideas, goals, and to support the formalization of the problem.



```
A region in the south-west secti              to be
developed.                                    purchase and
subsequen
recreation                                    er of 50
single fa                                     metery are
to be sit

Assume tha                                    f the site have been
determine                                     phic mapping system. A map
of the si                                     The map shows eight lots
of land wh                                    evelopment.

Lot3, lot                                     relatively flat sites with
fairly go                                     and lot12 are moderately
sloped sites in anice wooded location, but have poor soil
conditions. Lot17 is a very steep site. Lot11 ant lot17 are
elevated sites facing southwest and down into a valley that
has a lake and some wooded area.

The problem solver's task is to come up with assignments of
landuses to sites. A complete design is one in which each
landuse has been assigned to a lot. The final design should
be one which complies with a given set of criteria:

        1. The dumpsite and the cemetery should not be
visible from either of the dwellings.
        2. The recreational area should be near some body of
water.
        3. Steep slopes are to be avoided for building.
        4. Poor soil should be avoided for those landuses
that involve construction.
        5. The highway is noisy and ugly and should be
avoided when locating the apartments, the single-family
housing complex and the recreational areas.
```

```
7. Please select second group
8. Try to find child groups for these groups: - objectives - hard
9. Please select first group
10. Please select second group
11. Please select first group
12. bind group option canceled
```

Figure 4: The Comind's Brainstorming agent.



```
Parameter                 Domain

recre {3 5 7 9 10 11 12 17}
appartment {3 5 7 9 10 11 12 17}
houses {3 5 7 9 10 11 12 17}
dumpsite {3 5 7 9 10 11 12 17}
cemetery {3 5 7 9 10 11 12 17}
school {3 5 7 9 10 11 12 17}

Constraints

recre != school
appartment != school
houses != school
dumpsite != school
cemetery != school
((dumpsite == 3) or (cemetery == 3)) -> ((appartmen
((dumpsite == 3) or (cemetery == 3)) -> (houses !=
((dumpsite == 5) or (cemetery == 5)) -> (appartment
```

Figure 5: The Comind'sParam-Def agent.

After the production of a good number of ideas, the designer decides to use PARAM-DEF in order to define his problem in a more formal way and to be able to solve it (fig.5). As the number of constraints increases, the resolution of the problem can get impossible for a human solver. Artificial intelligence techniques for search and for finding solutions in a constraints' network are well developed. The machine's power has considerably increased and thus computers are good computational assistants for humans. The definition follows the structure of a Constraint Satisfaction Problem. However, the constraints can be defined by either logic rules or matrices. Rules' writing can be a difficult task for a designer. The PARAM-DEF assistant supports this process by providing to the user a menu containing all the possible types of rule and their intuitive translation. It also provides a visualization of the graph created by parameters and constraints as shown in figure 3. It is a good visual feedback for the user because it gives a short view of the constraints' network.

3.1.2. In the Flight rescheduling system

The infosphere [1] is the set of all the available information concerning the area of a scheduling application. For example in the case of flight scheduling the periodical flights plan, the operational data of the airline's fleet, the characteristics of all reachable airports and the actual weather conditions along flying routes constitute a part of the infosphere. All these data do not necessarily reside in the same place but they can be distributed among several, heterogeneous databases. They can also be found as part of the knowledge of the human operator. When a problem arises the operator needs to turn to the rescheduling application for reallocating resources. His first task is to confine the problem area by retrieving from the infosphere only the relative information. The system responds by building a visual workspace. The user can interact with it, adding or collapsing data from the infosphere or information he draws from his personal knowledge. The final goal is to both refine the visual workspace and build a query that best represents the rescheduling problem.

Refining the workspace serves a double objective, it restricts the search space and it helps the user to understand all the parameters of the problem and how possible solutions may affect the current status of allocations. In the case of flight scheduling a possible problem scenario is the demand for upgrading an aircraft scheduled to fly a particular route. Since all of the airplanes are at all times supposed to be allocated at a task - maintenance included - it is obvious that a reshuffle has to take place so that another, larger in seating capacity, airplane is assigned to this flight and at the same time the rest of the schedule can be performed in a similarly satisfactory way. Although the task in hand seems to be trivial, in reality it is a hard combinational problem. First of all, the system is responsible of constructing and maintaining the visual workspace; a representation of a complex information equilibrium of constraints, rules and interdependencies.

Visualizing and interacting within this workspace is the only way to convey to the user all this amount of information. In our example the main visual abstraction being used in the visual workspace is the pictorial representation of the flight schedule, shown in figure 2. The flights, depicted as rectangles, are ordered in the horizontal axis according to time and in the vertical axis according to the aircraft currently assigned to them. Flight names, airport origins and destinations and time dependencies can be instantly seen. With some further inspection more useful information can be deduced like similarity patterns that may point to specific exchange candidates, scheduling periods and aircraft's flying hours. On user's demand, more data can be displayed such as seating capacities, flight occupancies, airport characteristics, aircraft attributes and required ground times. Particular flights can be selected for aircraft reassignment or the user can suggest, by dragging and dropping flight rectangles, specific reallocations to be included in the solutions It is important to note that everything the operator sees in his workspace specifies the subset of the whole schedule on which any changes can take place. It is only the flights, aircraft and time horizon being displayed that determine the boundaries of the search space. In the case of our upgrading example the user may build the visual workspace by selecting the aircraft to be reallocated and all the airplanes of the same type and of another type that has compatible characteristics but larger seating capacity. He can also restrict the time horizon in one week so that any proposed changes will affect the overall schedule within this time period only.

3.2. Solving the problem

3.2.1. In Comind

We have implemented different algorithm for solving a CSP. We provide to the user visualizations of the going-through process of those algorithms. S/he is thus in control of the searches and can decide to stop them when ever s/he wants. For example, an usual user interaction with our system consists in first using node-consistency, and then arc-consistency. The user can then evaluate the number of solutions for the problem by using the knuth algorithm. If it is under-constrained, he can either specify more constraints or use the tradeoff agent in order to compare the solutions. If it is over-constrained, the user should use the elicit conflict agent in order to release the problem. In the case where the problem is neither over nor under-contrained, the user can do a backtracking. In all this step, visualizations are provided. It is a good way for the user to observe the processing and to control it. For example, the visualization in the figure 6 shows the going-through process of an intelligent backtracking, the number of solutions already found, the percentage of the space already browsed and the time remaining. When the algorithm stop, the according responsible constraints are represented by a color. The idea is mainly to make the user able to use several kind of algorithms, making them cooperate through interactive visualization.

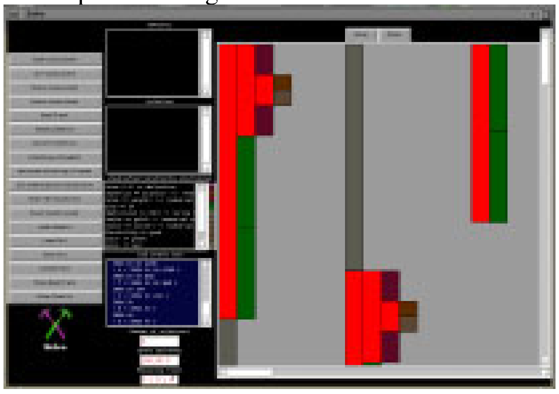


Figure 6: The Comind's Solve agent.

3.2.2. In the Flight rescheduling system

After the user enters any additional constraints or preferences he invokes the system's search engine. At this moment we pass into the second phase of the rescheduling process. Although the flight rescheduling application we have built offers limited interactivity and visualization during this particular phase, it is in some cases desirable to do so. When the search space is large and presents qualitative or quantitative regularities it is often advantageous to visualize both the process of searching and where the solutions are found. This permits to the user to either interrupt the search and continue with the already found solutions or to uncover possible areas of contention. According to the best case scenario the machine based search will respond with one and only solution that best meets the user query. Unfortunately this is rarely the case in resource allocation. If the problem is over-constrained then no answer will be found. We can remedy this situation by returning to the visual workspace and relaxing some of the imposed constraints or simply enlarging its size.

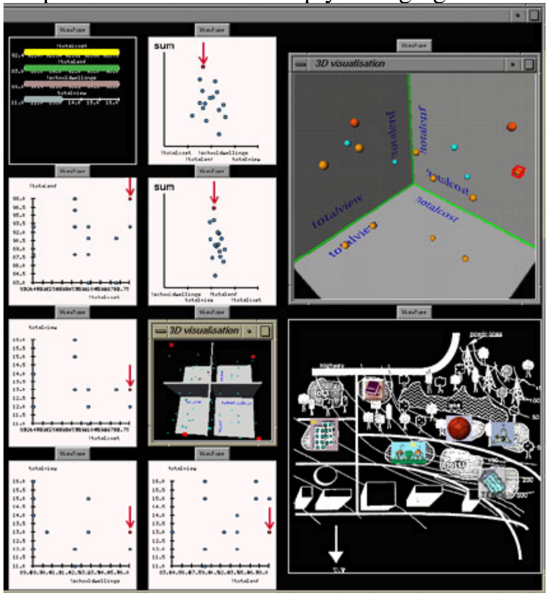


Figure 7: The Comind's Tradeoff assistant.

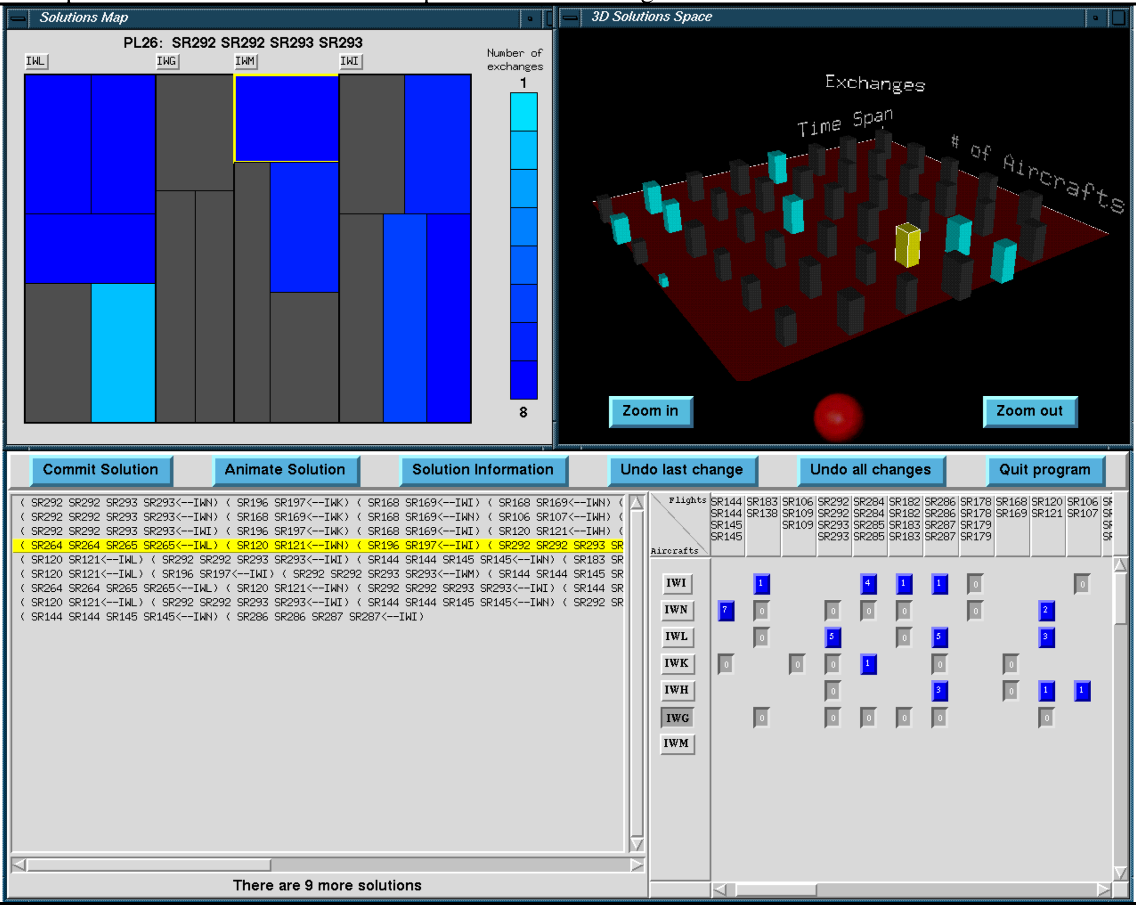
3.3. Helping tradeoff for under-constrained problems

3.3.1. In Comind

The goal of the tradeoff agent is to find the optimal solution in an under-constrained problem. For example, figure 7 represents the multicriteria optimization of a land allocation problem. The user has defined criterias, like the cost, the view's quality or the noise tolerance, in order to compare the solutions. Several visualizations are proposed in order to help the user juggling with tradeoffs. Visual interactivity is particularly important here because it allows, by interacting with different views of solutions, to find the better solutions according to a certain set of criterias. Furthermore, when the user has the intuition that a solution can be good, s/he is free to make it optimal by finding the set of criteria which satisfies it. The designer can thus also evaluate the quality, the sharpness, of the criteria.

3.3.2. In the Flight rescheduling system

At this stage, the human expert must be engaged in a productive loop process of examining, evaluating and filtering out the solutions set. He can reason which solution best fits the problem, based on knowledge patterns drawn from experience, subjective preferences and/or additional criteria, such as the introduction of last minute qualitative constraints. The ultimate goal is to select the optimal solution. Experience during the development of the flight rescheduling system has showed that multiple, partially both complementary and repetitive in context, interactive representations of the solution space can greatly assist this goal. The user interacts, with any of the, displayed in parallel, representations in order to solve any perceptual ambiguities, deepen the level of displayed information and/or impose his own constraints thus limiting the number of available solutions. An intercommunication mechanism between the different representations creates an additional level of visual information flow towards the user. Therefore, the interrelations between solution attributes and contextual groupings of the solutions become apparent. All in all, information abstraction and interactive visualization along with machine based constraint satisfaction, free up the reasoning process by hiding unnecessary information details and speed up the ability to approximate the optimal solution. In the flight rescheduling system there are four morphotypes being utilized in the postprocessing phase, all displayed in the same screen, each occupying a quarter and in operational coordination with each other (fig. 8). The upper left portion of the interface is a map of the children nodes of a tree, build with the ID3 algorithm. The children nodes represent all the available solutions. So, each rectangle is a solution with the color codifying its number of aircraft reassignments. Each subsection of the map represents one of the subtrees of the current level of the tree. Every split of the tree corresponds to a decision to be made on which aircraft is to be assigned to the given flight.



```
( SR292 SR292 SR293 SR293<--IWN) ( SR196 SR197<--IWK) ( SR168 SR169<--IWI) ( SR168 SR169<--IWN) (
( SR292 SR292 SR293 SR293<--IWN) ( SR168 SR169<--IWK) ( SR168 SR169<--IWN) ( SR106 SR107<--IWH) (
( SR292 SR292 SR293 SR293<--IWI) ( SR196 SR197<--IWK) ( SR168 SR169<--IWI) ( SR120 SR121<--IWH) (
( SR264 SR264 SR265 SR265<--IWL) ( SR120 SR121<--IWN) ( SR196 SR197<--IWI) ( SR292 SR292 SR293 SR
( SR120 SR121<--IWL) ( SR292 SR292 SR293 SR293<--IWI) ( SR144 SR144 SR145 SR145<--IWN) ( SR183 SR
( SR120 SR121<--IWL) ( SR196 SR197<--IWI) ( SR292 SR292 SR293 SR293<--IWM) ( SR144 SR144 SR145 SR
( SR264 SR264 SR265 SR265<--IWL) ( SR120 SR121<--IWN) ( SR292 SR292 SR293 SR293<--IWI) ( SR144 SR
( SR120 SR121<--IWL) ( SR292 SR292 SR293 SR293<--IWI) ( SR144 SR144 SR145 SR145<--IWN) ( SR292 SR
( SR144 SR144 SR145 SR145<--IWN) ( SR286 SR286 SR287 SR287<--IWI)
```

There are 9 more solutions

Flights / Aircrafts	SR144 SR144 SR145 SR145	SR183 SR138	SR106 SR109 SR109	SR292 SR292 SR293 SR293	SR284 SR284 SR285 SR285	SR182 SR182 SR183 SR183	SR286 SR286 SR287 SR287	SR178 SR178 SR179 SR179	SR168 SR169	SR120 SR121	SR106 SR107
IWI		1			4	1	1	0			0
IWN	7	0		0	0	0		0		2	
IWL		0		5		0	5			3	
IWK	0		0	0	1		0		0		
IWH				0			3		0	1	1
IWG		0		0	0	0	0			0	
IWM											

Figure 8: The flight rescheduling's tradeoff workspace.

If the user selects one of the possible aircrafts then he is left only with that subtree of solutions to work with and the tree map is redrawn accordingly. The upper right portion of the screen is dedicated to a 3D abstraction of the solutions, with each cube depicting a solution. The 3 dimensions of the cube are analogous to three characteristics of the solution, the number of aircraft reassignments, the time span of the exchanges and the number of different individual aircrafts being involved. The bottom left section of the text is a simple textual representation where each line describes a solution. The right bottom part is a graphical plot of the solutions. The horizontal axis hosts the flights and the vertical one the aircrafts participating in all of the solutions. Each button of the plot indicates how many solutions include the aircraft-flight assignment of that position. The user can take out, if he presses that button, all the corresponding solutions. In a similar way, solutions can be taken out by deselecting individual aircrafts or flights.

It must be mentioned that all of the supported abstractions are interdependent and coordinated with each other. So when for example a 3d cube solution is picked then the right solutions are highlighted in the rest of the representations. Similarly, if solutions are taken out from the solutions plot then the corresponding cubes and rectangles are disabled, dark grayed in color, and the listbox of textual solutions is annotated. Detailed information about any participating object of the abstractions can be retrieved on demand.

3.4. Eliciting the conflict for over-constrained problem

3.4.1. In Comind

An over-constrained problem is a problem that does not produce any solutions. It is over-constrained either because the problem is over-determined, too many constraints have been defined, or because of a conflicting definition. This distingo between an over-determined definition and a conflicting definition is not really correct theoretically, nevertheless, for the user the difference is quite big. When a problem is over-determined, the user has to relax constraints. When the definition is conflicting, he has to rethink his problem.

When a problem is over-constrained, a normal backtracking does not find any solution. Partial CSP is most often used technic for diagnosing over-constrained systems [4]. It consists in the search of a satisfiable sub-problem. The biggest satisfiable sub-problem is considered as the closest of the initial definition and is supposed to generate the optimal solutions. Different optimization criteria, called metrics in PCSP, are used in order to decide which is the smallest set of constraints to relax, or the biggest sub-problem with solutions. The Maximal Satisfiability (Max CSP) is one of those metrics. It selects solutions which maximize the number of constraints satisfied without distinguishing them. More often a weight mechanism is introduced in order to prioritize the relaxation of certain constraints. Hierarchy, probability, fuzziness, temporality, or dynamism are different methods used to extend PCSP. According to the PCSP formalism, less a tuple is violated, better the solution is. Nevertheless, in design, it is not always true. The method we propose is different because it is based on user's intuition and the algorithm we propose help the user diagnosing the conflict. Thus, our method can be seen has an alternative to Partial CSP.

Finding the semantic antagonism that forbids the finding of solution is quite impossible for a machine. Furthermore, a problem can produce no solutions but possesses no real conflict. It seems that it is hard to deal with conflict. In the context of design, eliciting automatically the most important conflict is a difficult task to solve as far as computer programs are not yet able to distinguish a semantic conflict from another. As conflict elicitation is an hard problem, we propose a different approach than the automatic one. The idea is to propose to the user a set of tools that can help him to deal with an over-constrained problem. In general, we mainly want to show that when artificial intelligence techniques fail to solving a problem, interactive systems are able to build a synergy between natural and artificial intelligence.

Furthermore, in my point of view, conflict is just a view of mind, there are no conflict, just constraints violations. What can be considered as a conflict is, at most, a set of constraints which violates its related tuples 100% of the time. But this case is too rare to be used for the implementation. According to this non-belief in conflict for the implementation, we propose several visualizations based on constraint violation instead of conflict.

The elicit conflict entity is distributed in several sub-entities applicable to different type of problem but let to the choice of the user.

Set of constraints responsibility distribution (SCRD)

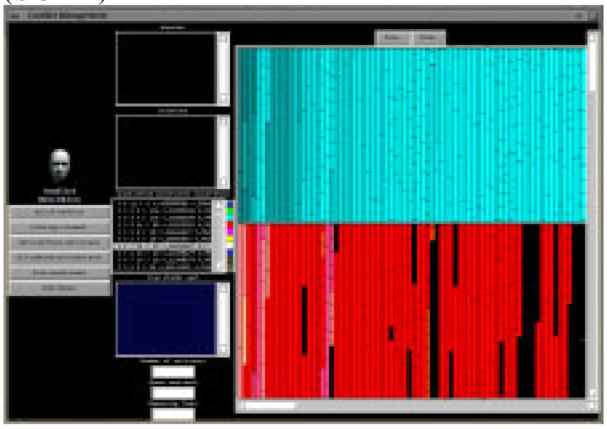


Figure 9: The Comind's Elicit conflict agent.

The idea behind the first algorithm is based on the distribution of the sets of constraints responsible of the failure of each tuples of the space. The algorithm also gives back the the number of tuples each set is violating. The visualization contains two different views of the same space (fig. 9). Both represent the total space of tuples. Each different region represents the tuples forbidden by a specific set of constraints. Thus, the size of the region represents the number of tuples violated by a specific set of constraints. The sum of all the violations of tuples is equal to the number maximum of tuples for an over-constrained problem. The rectangles, the set of constraints, are ordered by their numbers of constraints. The smallest sets are the first ones beginning from the top-left part of the visualization. In fact, it is generally more easy to relax a small number of constraints than a big one for cognitive reasons mainly.

The first space at the top of the visualization uses a second argument for ordering. Inside each family having the same number of constraints, the sets are ordered by their inclusive degree of violation. All the constraints of the set need to be violated in order to be considered as responsible of the violation. For example, a set of constraints containing only one constraint which is never satisfied (100% of violation) is going to be the first one at the top left part of the visualization. Notice that each set of constraints have a different degree of darkness according to their percentage of inclusive violation. The second space at the bottom represents also the total space of tuples. Inside each family having the same number of constraints, the sets are ordered by their exclusive degree of violation. Only one constraint of the set needs to be in order for the set to be considered as responsible of the violation. Notice that here also each set of constraints have a different degree of darkness according to their percentage of exclusive violation. Notice also that when the color of the space is black, the set of constraints is violating at 100%, thus no solution can be found without relaxing it. It is a conflict.

The visualization provide also the number of tuples that are forbidden by a set of constraints. If a set of constraints is removed of the problem, the solutions found will correspond to the size of the space of its corresponding region plus the size of the space of its corresponding sub-set of constraints. This provides to the designer a good indication of which set of constraints he has to work on in order to find optimal solutions. Furthermore, by cliquing on a specific area of the visualization, he obtains the solutions related.

Knuth based pruning of the conflicting space

When the number of parameters and the size of their domain increases, the first family of algorithms is still consuming. For this reason, we propose an interactive approach that combines different simple algorithms in order to reduce the initial space and make the use of the previous algorithm possible.

It first uses the Knuth algorithm for evaluating the efficiency of backtracking programs. The Knuth algorithm is using a Monte Carlo approach, based on a random exploration of the tree. For each partial solution, a random valid continuation is chosen. When the algorithm reach a leave of the tree, the estimated number of solutions is returned, according to the going through. The algorithm goes through only one node of each level in the search tree and is thus very fast.

We use here this approach in order to estimate the number of solutions of each sub-problems, all the problem minus one parameter. The previous algorithm is very fast but is an estimation and thus some mistakes can happens. Nevertheless, its goal is mainly to give to the user an idea of which part of the problem he has to revise. It is also going to be useful for the user in order to reducing the size of the problem and for engaging a future more detailed research. The visualization attached to this algorithm is very simple. It represents the different regions of the initial problem minus one parameter and the estimation of their number of solutions. Each rectangle represents the relative number of solutions like a classic histogram.

4. Conclusion

We have shown two example systems targeting complex problem solving and the common architecture they use. The triangle composed of the user, computational tools, and interfacing tools is an effective and simple architecture that can be applied to many problem solving application areas. Where artificial intelligence technics fail, human intelligence can be used as the catalyst for solving any deadlocks. Furthermore, the human can use the available computational tools to expand his memory, processing and reasoning capabilities and the support of interfacing tools to improve his attentional memory and his perceptual skills.

Bibliography

1. Card "Visualizing Retrieved Information: A Survey", IEEE Computer Graphics and Applications, March 1996, pp.63-67.
2. Choueiry, B. Y. and Faltings, B., "Interactive Resource Allocation by Problem Decomposition and Temporal Abstractions", Second European Workshop on Planning, Vadstena, Sweden, 1993. In Current Trends in AI Planning, In series Frontiers in AI and Applications, IOS Press, Amesterdam, 1994.
3. Fischer, G.: Creativity enhancing design environments, in Modeling Creativity and Knowledge-Based Creative Design, edited by J. S. Gero and M. L. Maher, Lawrence Erlbaum Associates, Inc., Publishers, 1993.
4. Freuder E.C. and Wallace R.J., Partial Constraint Satisfaction, in Lecture Notes in Computer Science, Springer Verlag, 1995.
5. Gomes P., Bento C., Gago P., and Costa E.: Towards a Case-Based Model for Creative Processes, in Proceedings of the Eurpean Conference on Artificial Intelligence, Budapest, 1996.
6. Wallace, M., "Applying Constraints for Scheduling", in B. Mayoh, E. Tyugu, J. Penjaam (Eds.), "Constraint Programming: Proceedings 1993 NATO ASI", NATO Advanced Science Institute Series, Springer, 1994.
7. Woodbury, R.: A genetic approach to creative design, in Modeling Creativity and Knowledge-Based Creative Design, edited by J. S. Gero and M. L. Maher, Lawrence Erlbaum Associates, Inc., Publishers, 1993.