Artificial Intelligence for Engineering Design, Analysis and Manufacturing (1995), *9*, 313-323. Printed in the USA.

Management of conflict for preliminary engineering design tasks

DJAMILA HAROUD,[1] SYLVIE BOULANGER,[2] ESTHER GELLE,[1] AND IAN SMITH[1]

[1] Artificial Intelligence Laboratory (LIA) and [2] ICOM (Steel Structures), Swiss Federal Institute of Technology (EPFL), IN-Ecublens, 1015 Lausanne, Switzerland

(RECEIVED December 15, 1994; ACCEPTED February 7, 1995)

Abstract

Much of preliminary engineering design is a constraint-driven non-monotonic exploration process. Initial decisions are made when information is incomplete and many goals are contradictory. Such conditions are present regardless of whether one or several designers contribute to designs. This paper presents an approach for supporting decisions in situations of incomplete and conflicting knowledge. In particular, we use assumptions and conflict management to achieve efficient search in contexts where little reliable information exists. A knowledge representation, containing a semantic differentiation between two types of assumptions, is used within a computational model based on the dynamic constraint satisfaction paradigm. Conflict management strategies consist of three *generic* mechanisms adapted to the type of constraints involved. These strategies may be refined through consideration of variable importance, context, and design inertia.

Keywords: Conflict Resolution Strategies; Dynamic Constraint Satisfaction; Preference and Default Assumptions

1. INTRODUCTION

During the preliminary stages of many engineering design tasks, the underlying search space is unbounded and inexact. Relevant information is revealed only as the design proceeds and as decisions are taken. In practice, designers are able to deal with this complexity by managing a large amount of unformalized—and potentially retractable—knowledge, such as *assumptions*. Their experience and skills are judged through their ability to make good decisions in situations of incomplete knowledge. To support design tasks efficiently, a knowledge-based system should explicitly model strategies that cope with incompleteness and inexactness. Such strategies should integrate appropriate representation of *assumptions* with techniques for *solving conflicts* which they engender.

Assumptions are widely used in preliminary design tasks. In traditional computational models, such as those using assumption-based truth maintenance architectures [ATMS (De Kleer, 1986)], an assumption is mainly considered to be a *defeasible decision* useful to initiate the search process and guide it through incomplete knowledge (Logan et al., 1992; Sham, 1993). Such use of assumptions does not introduce any semantic differentiation between the types of assumptions made. Consequently, there is no explicit link between the way one resolves a conflict and the kinds of assumptions involved in it. In practice however, the designer incorporates within assumptions the following two different types of knowledge:

- *default information* including what is known to be the case "usually,"
- criteria reflecting *desirable properties* of the final structure, etc. . . .

This categorization is not only used for orienting the search in a given direction but also for guiding the choice of conflict management strategies (e.g., choosing whether to accept a compromise on a given solution or try a different alternative).

Traditionally, conflict resolution techniques in constraint-based models of the design process use *backtracking* and *constraint relaxation*. The latter refers to constraint-driven methods using an explicit hierarchy of the constraints for the choice of relaxable constraints. The implementation method consists generally of removing constraints to identify feasible solutions. Constraint classification includes different methodologies such as mathematical analysis (monotonicity analysis) (Krishnan et al.,

Reprint requests to: Djamila Haroud, Artificial Intelligence Laboratory (LIA), Swiss Federal Institute of Technology (EPFL), IN-Ecublens, 1015 Lausanne, Switzerland. Phone: +41 21 693 52 09.

1990), and domain-dependent taxonomies (Bowen & Bahler, 1992*a*, 1992*b*). We propose a framework for conflict management that explicitly links strategies to the type of constraints involved.

This paper describes a design system that supports conflict management. Although the ultimate goal is to provide an interactive system, the strategies described help automate tedious decisions that are linked to higher-level design requirements. A knowledge representation that introduces a semantic differentiation between types of assumptions, *default assumptions* and *preference assumptions* is proposed. It shows that this categorization can be used by control knowledge for task organization and conflict management. The next section introduces the need for defining a framework for using assumptions in preliminary engineering design. Section 3 describes how this framework can be integrated into a dynamic constraint satisfaction model of the design process. Section 4 focuses on issues of conflict management and finally limitations are discussed in Section 5.

2. ASSUMPTIONS IN PRELIMINARY ENGINEERING DESIGN

In preliminary engineering design, various methods, ranging from traditional statistical tools to more recent techniques such as default reasoning, fuzzy logic and neural networks, are currently proposed for treating imperfect knowledge.

Typical trends of two design characteristics are shown in Figure 1. The most important decisions, related to factors such as costs, safety, and environmental impact, are made at the beginning. The curve *precision of knowledge* follows an opposite trend: knowledge is poor at first and increases as tasks near completion. On this curve, it is also shown when certain methods, which accommodate imprecise knowledge, become relevant for implementation.

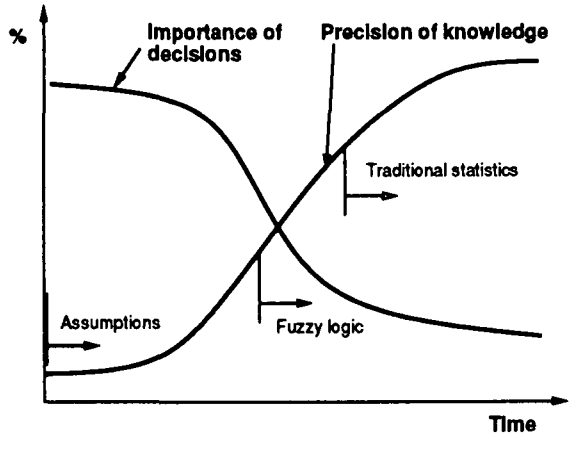

Fig. 1. The importance of design decisions and the precision of knowledge versus time. The most important decisions are made when knowledge is least precise.

Traditional statistical methods require knowledge of all important parameters as well as the nature of their distributions and fuzzy-logic methods require weight-factor distributions for each parameter. Although neural networks are applicable at the beginning of engineering tasks, their usefulness strongly depends upon the quality of examples that were used to train the networks. Furthermore, it is difficult to maintain performance when these methods are integrated within iterative decision-making processes as the task proceeds. The notion of assumptions refers to the implicit knowledge used by designers to create complex structures from imperfect knowledge. Assumptions cover two types of information:

- *default* knowledge: reflecting a rough statistical understanding of previous experience in the field
  "*Buildings usually have four exterior walls*"
- *preferences*: reflecting the wishes to proceed a certain way according to ill-formalized criteria (aesthetic, environmental considerations, politics, etc.)
  "*Given a choice, study the cheapest alternative*"

In this work, a framework is defined for explicit representation of these two types of assumptions: *preference* assumptions and *default* assumptions. It is integrated within a computational model of preliminary design task to initiate and guide the decision process in situations of incomplete and imprecise knowledge.

3. DYNAMIC CONSTRAINT SATISFACTION

Design problems can be described naturally in terms of the dynamic constraint satisfaction paradigm (Mittal & Falkenheiner, 1990). Design relationships can be stated as *constraints* while *variables* stand for particular characteristics or dimensions of the design. To develop a computational model of the design process according to the constraint satisfaction paradigm, two major characteristics have to be taken into account:

- the set of variables and constraints involved is not independent of particular solutions: many changes in the design are made in response to values of variables. Therefore, activation of many variables and constraints depends upon earlier decisions.
- a number of variables have values in continuous domains.

The first consideration stems from the fact that constraint activation is *conditional*. For example, as shown in Figure 2, depending on the values of the variables `beam-depth` and `span` different constraint combinations hold. A *search space* of different combinations of active constraints can therefore be generated, which can form disjoint solution sets. In general, constraint propagation as well as optimization techniques assume a single convex so-

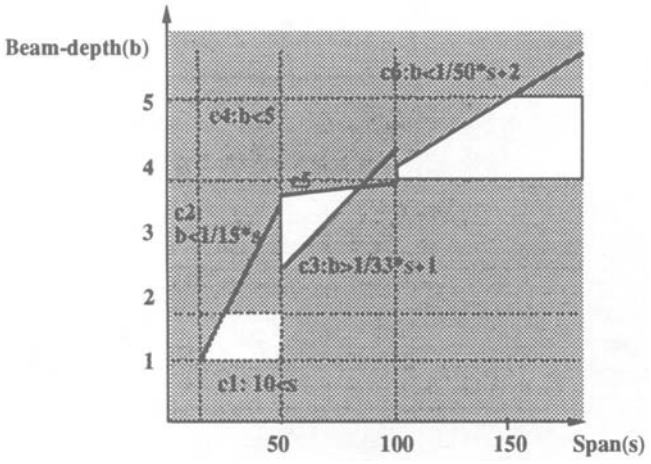

Fig. 2. In a dynamic constraint satisfaction problem, even linear constraints can result in an unbounded number of disjoint regions.

lution set. To satisfy this assumption, solving a Dynamic Constraint Satisfaction Problem (DCSP) requires combination of a *search process* deriving the consistent constraint combinations with *propagation* of the constraints separately within each solution region (depicted by Fig. 3). The nature of this search is determined by the second consideration; in continuous domains the set of possible alternatives is not enumerable. Consequently, constraint networks describing designs are generated and maintained incrementally; relevant contexts are therefore generated through a *sequential* search. The search process mentioned above refers to *structuring* aspects of design rather than problem-*solving* or value assignment as carried out in WRIGHT (Baykan & Fox, 1992).

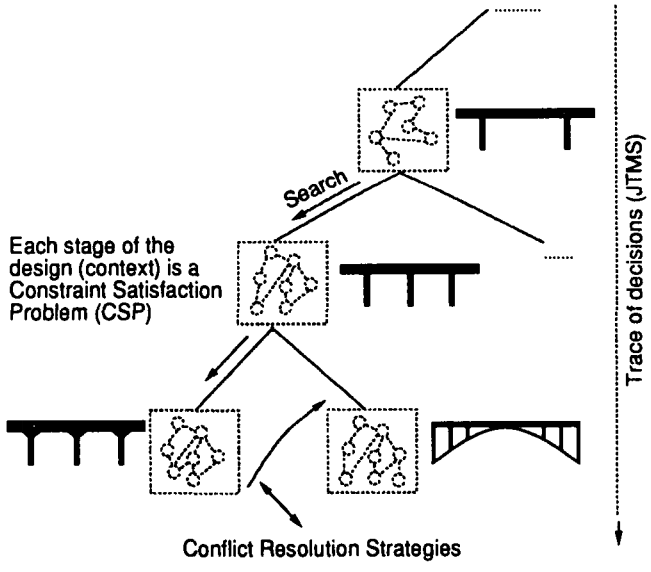

Fig. 3. The DCSP model of the design process: Constraint propagation ensures consistency at each level while the search process derives sequentially relevant variables and constraints. The search process generally starts with default assumptions, and it is guided toward preferred alternatives by preference assumptions.

Forward Reasoning using Assumptions: Conceptual design introduces incrementally *structures* and *relationships* using sets of *design rules*. According to the DCSP formulation, our algorithm translates this notion into a dynamic introduction of *variables* and *constraints* using *rules*. Incremental introduction of rule sets establishes priorities as is discussed in a subsequent section. To integrate assumptions explicitly in the search process, rules are divided into two categories: *fixed* and *assumptions*. Assumption rules integrate the knowledge which is most important for effective support of preliminary stages of engineering tasks. The information inferred take the form of *variables* or *constraints*, which are propagated through one another to fix consistent constraints. In the system we have implemented, a *constraint* is a numerical equality or inequality involving variables and constants as well as a more traditional symbolic constraint. Three different types of constraints are considered:

- *default constraints:* inferred by default rules, initiate the search when no precise information is available;
- *preference constraints:* inferred by preference rules, focus the search on preferred alternatives according to criteria that are difficult to represent more explicitly than through precise models (aesthetic, cost etc . . .);
- *fixed constraints:* inferred by fixed rules, orient the search in mandatory directions according to physical principles (e.g., "*the sum of all bridge spans must equal the length of the bridge*") or technological considerations (e.g., "*during construction, the maximum element length is 30 meters*").

Each rule set may contain rules which activate any type of constraint.

Constraints and variables are asserted with justifications that are managed in a justification-based truth maintenance system (JTMS) (Doyle, 1979), which has been extended to accommodate constraints expressed using continuous variables. The algorithm used iterates through the following steps until no further knowledge can be introduced:

1. load first set of rules;
2. using interval values for parameters activated by rules or default values contained in frames (a decomposition hierarchy), propagate values through constraint network;
3. check for feasible solution;
4. if no feasible solution, activate conflict management strategies and go to step 2;
5. if a feasible solution exists, load new rule set and go to step 2.

In our computational model, each stage of the design process is represented by a *context* consisting of the following elements:

- a set of active *variables* with their associated range of values (intervals for continuous variables);
- a set of active *constraints*;
- *justifications* of constraints and variable values in terms of the decisions that introduce them.

Given that the current context is *feasible*, rules are fired that deduce new knowledge (variables and constraints) on this knowledge. Whenever a new rule is fired, the admissible values are adjusted using the Waltz constraint propagation algorithm (Mackworth, 1977). Because the Waltz algorithm does not necessarily lead to globally consistent solutions, special techniques are under development for ensuring that feasible solutions are possible (Haroud & Faltings, 1994). More details on the computational model developed and the DCSP algorithm implemented are given in (Faltings et al., 1992).

A change in context occurs when

- a rule activation condition splits the variable value (interval) in two subintervals;
- conflict management strategies are invoked.

In the first case, the rule condition satisfies only part of the current value and two contexts containing the feasible subintervals are created. In the second case, underlying justifications change, thereby introducing new constraints and therefore, a new context. Maintaining explicit contexts and justifying them in terms of the decisions leading to their introduction, implements the necessary distinction between *control* and *domain* knowledge. Similar ideas of explicit context maintenance have been proposed by Petrie (1991).

4. CONFLICT MANAGEMENT

As a natural consequence of reasoning in situations of incomplete and inaccurate information, a large amount of conflicts arise that must be managed by the system. Simple constraint resolution is not appropriate because of the non-monotonic nature of design; often situations arise where it is possible to reintroduce design constraints.

The use of assumptions, translated into *default* and *preference* constraints along with the use of *fixed* constraints, is not only useful to guide the search but it also allows the implementation of conflict management strategies close to those applied in practice by designers. Indeed, when involved in situations of conflicting requirements, designers may use the following strategies:

- default assumptions are generally *abandoned* with no further consequences when more accurate information becomes available;
- preferences are not abandoned completely in case further information creates a *situation* where they can be satisfied;
- conflicts among fixed requirements necessitate the revision of activation conditions and design paths.

As illustrated in Figure 4, the conflict management strategies provided by our algorithm, allow us to support these strategies during the decision-making process. This design process, depicted by the sequence of bridges in this example, starts with a default solution consisting of a constant depth beam bridge with two piers and continuous space to be erected by crane. This solution is overridden as soon as conflicts with aesthetic, design, and earthquake requirements are detected. Earthquake considerations imply continuity of the central spans and hybrid end spans. This decision is *weakened* for the fourth span (at the third stage) as it conflicts with settlement considerations. A second conflict occurs at the third stage because the newly introduced construction preference (specifying that all the spans should be less than 50 m) is incompatible with the geometric condition that the sum of all spans must equal the length of the bridge. The construction preference is therefore weakened by allowing the second span to exceed 50 m. Finally at the last stage, backtracking on the type of construction is performed as crane method was not compatible with soil conditions and slope stability considerations.

In our system, a conflict is detected when the active constraint set in a context does not admit a consistent labeling. According to the nature of constraints involved, we distinguish two types of conflicts: *feasibility* conflicts, detected when fixed constraints are conflicting, and *assumption* conflicts, detected when assumption constraints (preferences or defaults) are involved.

The algorithm employed to select appropriate strategies maintains explicit search spaces as illustrated in Figure 5. Several conflict management strategies, pertaining to the nature of the conflict, are implemented. These strategies apply to the constraints involved in a conflict. The procedure for selecting a relaxable constraint plays an important role in our algorithm. The conflicting constraint set is generated by tracing the justifications of the value that is violated (the interval bounds violated in the case of continuous variables).

The order of importance given to different design criteria will affect not only how the initial rules are launched but also the result after conflict management strategies have been used, for example, see Figure 6. Therefore, rule introduction has a far greater impact than it does in traditional expert systems. As a result, we have linked rule introduction to design criteria to create a link to seman-

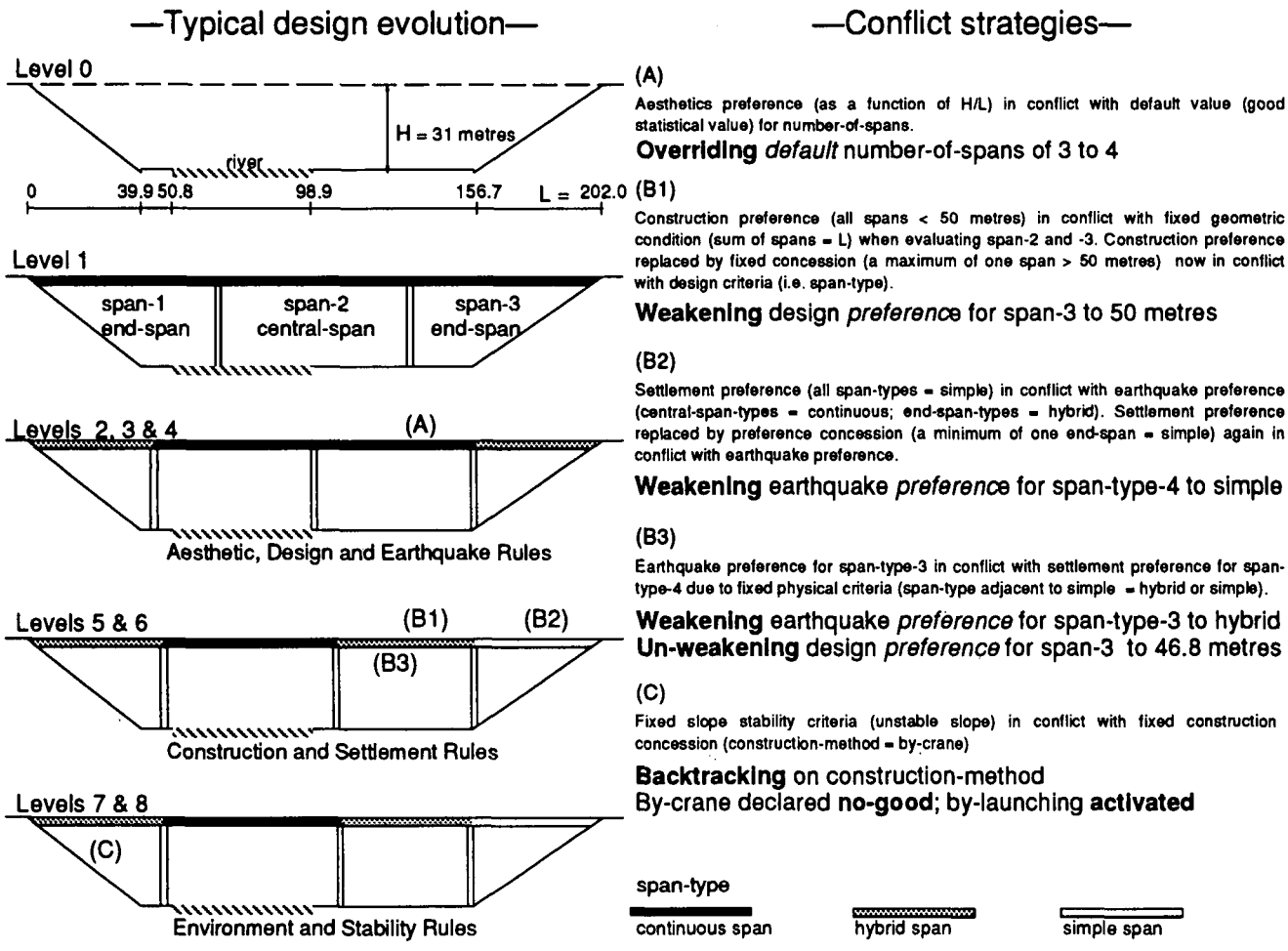

Fig. 4. A typical design requires different conflict management strategies: overriding, weakening, and backtracking. Overriding incorporates more accurate information by dropping defaults, weakening preferences reflects experience and compromises, backtracking revises the activation conditions of conflicting feasibility constraints. This example shows the evolution of a design considering rules derived from several criteria (aesthetic, construction, environment, etc.). Each rule set corresponds to one criteria and influences the design at a given level.

tic design aspects that can be understood by the designer. In this way, designers control the behavior of the system through changing the importance of design criteria, without direct consideration of rule order. Details are provided in the next section.

4.1. Conflict management strategies

As shown in Figure 7, these generic conflict management strategies are implemented to resolve conflicts: assumption conflicts are solved either by *dropping* default constraints or *weakening* preference constraints while fixed conflicts result in *dependency-directed backtracking*. An important step in the conflict management process consists of determining the most *relaxable* constraint (i.e., the constraint to drop, to weaken, or on which to backtrack). In general, the set of conflicting constraints contains more than one candidate, and a ranking of criteria is required to decide which one is more important for the designed structure. The design resulting from conflict management is *order-dependent* with respect to the constraint ranking adopted. In this work, a constraint ranking is established by the user. This ranking is currently based on the design criteria (cost, safety, etc.) represented by the constraints, but it can be refined using more specific considerations such as context-dependent information. The following points describe in more detail our conflict management strategies.

Overriding defaults: Default constraints are used to proceed with reasoning. They are based primarily upon empirical observations of previous designs and may have no relevance to the design context under consideration. Consequently, as soon as a new constraint, conflicting with a default constraint, is activated, the latter is *dropped* with no further consequences. This means that the knowledge inferred from the conflicting default constraint is

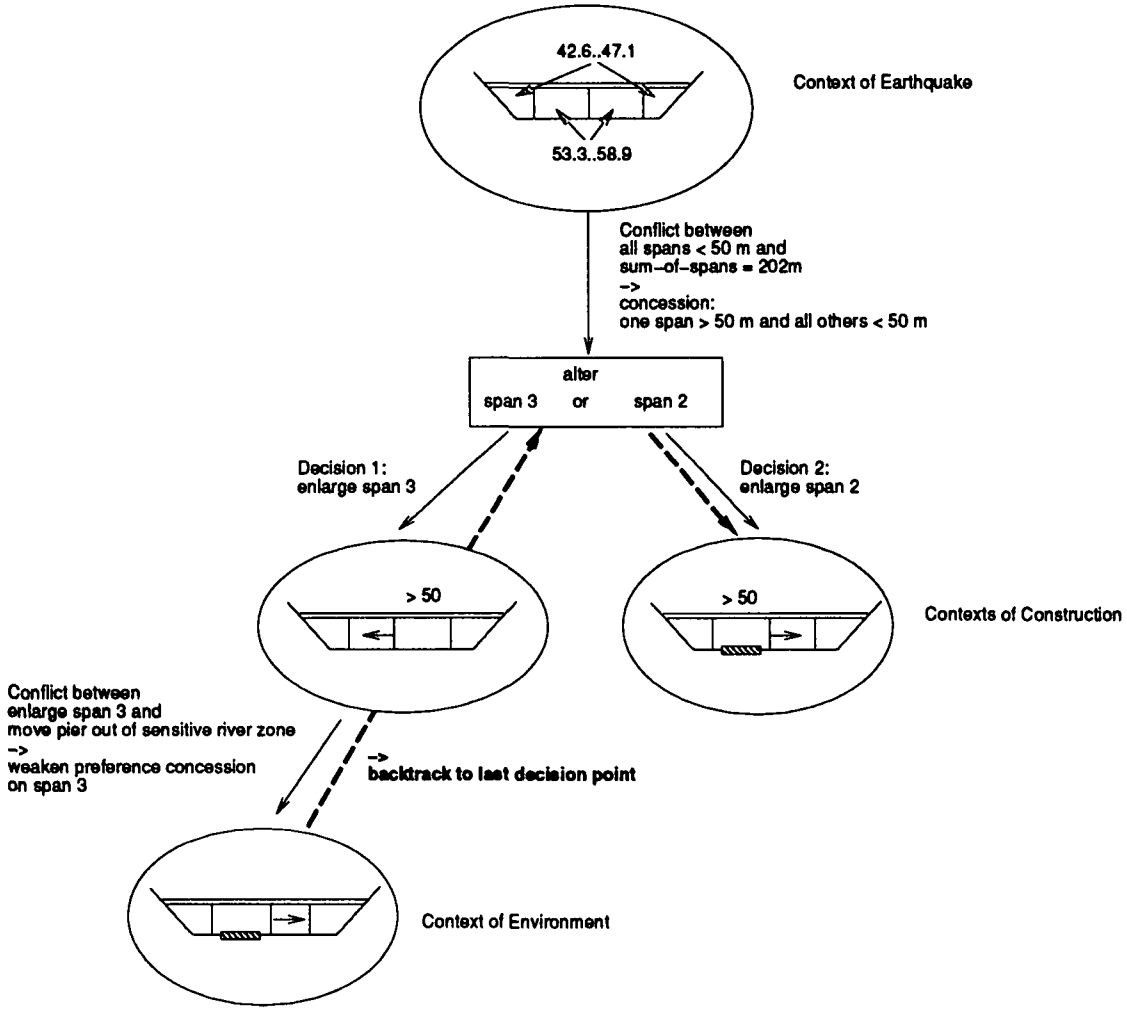

Fig. 5. *Context management* makes explicit different search spaces. If a conflict appears and none of the preference constraints can be weakened then the algorithm would backtrack to the nearest choice point (a compiled form of a set of contexts) and develop the next context from it. In this example, the choice of two alternatives (changing span-2 or span-3) must be made explicit by creating two different contexts. If altering span-3 is not possible (conflict involving fixed constraints), then backtracking automatically leads to the second choice of changing span 2.

simply retracted and the new set of constraints is propagated again and rechecked for consistency [see (A) in Fig. 4].

Weakening preferences: If a conflict remains after all related default information has been dropped, the relevant preference constraints are then examined. Preference constraints define the most satisfactory values for a local context; the closer the value is to the preference the better the criterion is adhered to. Conflicting preference constraints are therefore not *dropped* as default constraints but *weakened*. Two weakening approaches are used. The first weakening approach consists of replacing a conflicting constraint by an explicitly stated *second choice* (B1 and B2 in Fig. 4). For example,

"*If I cannot have all the spans less than 50 m, I will accept that one of them exceeds 50 m by a maximum of 30%.*"

This is implemented by specifying *concession* clauses in a preference constraint. When no concession clause is available for resolving the conflict, and this is most often the case, the principal weakening procedure is activated (B3 in Fig. 4). It consists of dismissing the conflicting constraint keeping its potential influence in memory: activation condition of the preference constraint *are not* reviewed and the decision of weakening the constraint is recorded as a justification for the feasible value issued from the propagation of the new constraint set. Hence, the weakened constraint can be fully reinstated when a situation arises where the decision of weakening is no longer necessary (B3 in Fig. 4). When specified, rules in concession clauses can be of two types, *preference* or *fixed*. In the first case, if the preference constraint activated by the concession clause doesn't remove the conflict, the basic weakening procedure is activated normally as described

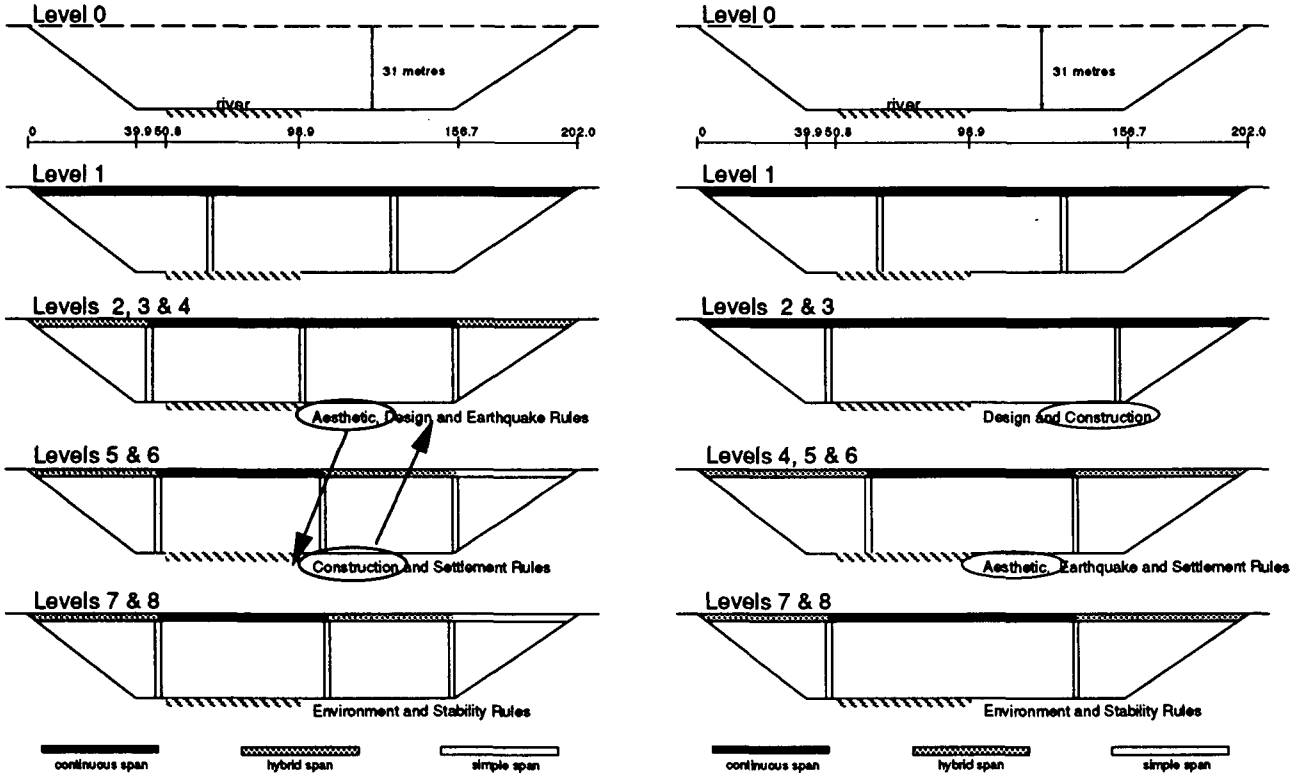

Fig. 6. Changing the order of introduction of knowledge alters the design proposal due to different activation of knowledge as well as different use of conflict management strategies.

previously. Finally, when the concession clause contains a fixed rule (B1 in Fig. 4), subsequent conflicts are treated the same way as *feasibility* conflicts, by dependency-directed backtracking, as described next.

Dependency-directed backtracking: When the conflict remains after all the preferences have been weakened, our algorithm performs dependency-directed backtracking to investigate alternative solution paths. Backtracking is done on the activation conditions of the conflicting constraints. The inconsistency is recorded by the TMS as a *no good* to ensure that the related path will not be tried again [see (C) in Fig. 4].

4.2. Constraint ranking

Two types of constraint ranking exist to manage the design environment: an overall ranking and a ranking that results from special strategies. They can be distinguished according to which main strategies employ the ranking, to what degree the ranking depends on the domain and on which entity the focus of attention is directed (constraints or variables).

Overall ranking
- applicable to three main strategies
- domain-independent
- constraint-driven

Special strategies
- used only when weakening is needed
- domain-dependent
- constraint and variable-driven

In both types of ranking, a close observation of the design process is required. In the overall ranking, two considerations are taken into account. First, a design task is hierarchical by nature. Diverse criteria such as those related to earthquake, construction or aesthetic requirements cannot be considered at once. Criteria ordering is influenced by the relative importance of these requirements attributed by each designer. Second, the design proceeds from less informed decisions to more informed ones and from less refined intervals of values to more refined ones. This implies a dual hierarchy because parallel to this procession is the fact that earlier rules generally introduce the most important criteria and the most important variables first. Final solutions are largely influenced by order (Fig. 6). Hence, in special strategies, three additional considerations introduce a differentiation of variables that would otherwise not appear in the overall ranking. Additional considerations include an examination of the context in which variables interact, an aggregation of constraints to reflect a proportional impact of constraints that are of the same nature, finally, an evaluation of the accumulated design commitment that avoids

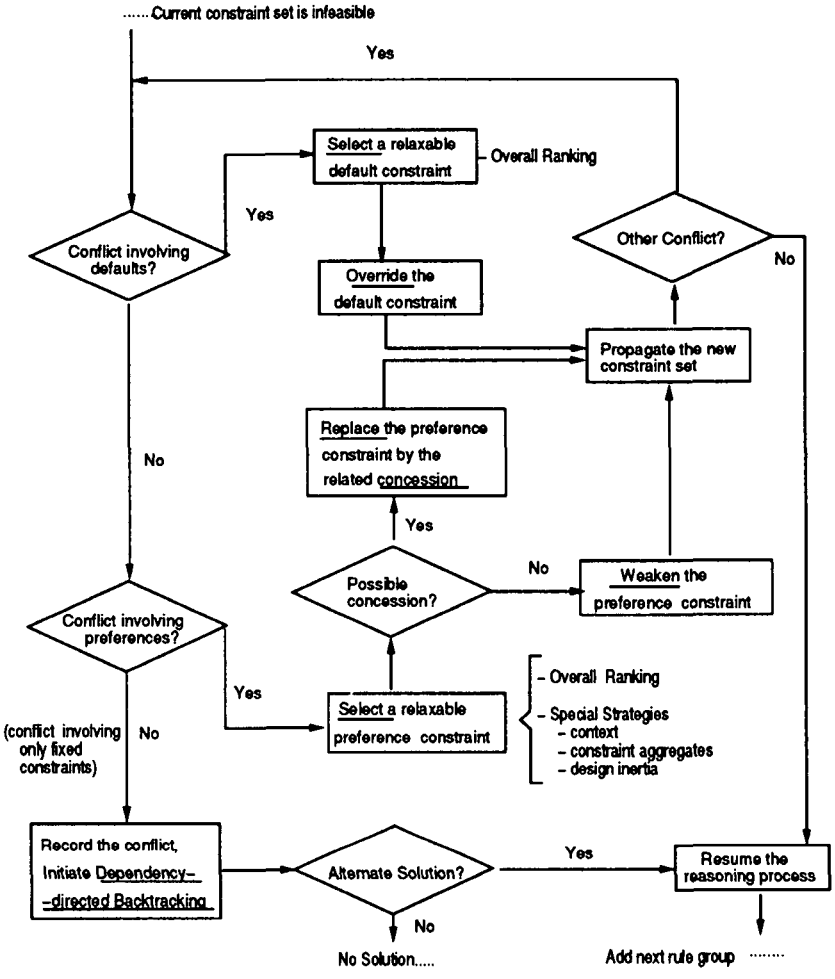

Fig. 7. The conflict management algorithm.

relaxation of values that have been subjected to compromises before relaxation of those variable values that depend upon fewer constraints.

4.2.1. *Overall constraint ranking*

In our system, a hierarchy of rule sets is established to incorporate the notion that the order of introduction of rule sets is important. When rule sets are formed according to design criterion, the importance of each criterion is established by the order of introduction of the rules in the system. In this way, earlier assumptions reflect either less informed or less important choices in the design process. According to this rule hierarchy, a basic constraint ranking can be established as follows:

- earlier preferences or default constraints are more relaxable than the later ones: they correspond to weaker criteria or less informed choices;
- earlier fixed constraints are more resilient than the later ones: the algorithm first backtracks on the activation condition of the later conflicting fixed constraint. In this manner, the search process will explore all possible detailed solutions before changing the global characteristics of the design. This makes the overall process more *efficient* as it tries to resolve conflicts at the stage that requires the least changes. The system always retracts the more recently asserted feasibility constraint provided that its justification is not an unretractable premise.

4.2.2. *Special strategies*

Special strategies for preferences were developed to improve the performance of the system. When testing the system using only the overall ranking, it was found that important decisions were being made arbitrarily and this resulted in unacceptable design proposals. For many types of conflicts, more specific conflict management knowledge may introduce new solutions more directly and more adequately than general strategies. Indeed, the overall ranking is unable to integrate explicitly considerations related to contexts, knowledge levels, and design commitment. Special control knowledge integrates such considerations. Note that these special strategies are only used to assist the overall ranking in decisions involving equitable choices

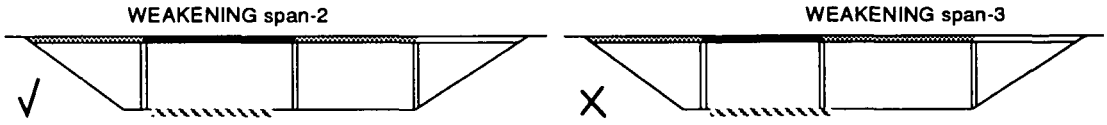

Fig. 8. Context sensitivity—When confronted with weakening two constraints of the same importance, in this case involving variables span-2 and span-3 for construction reasons, context may provide information in addition to the overall ranking: for example, the presence of the river under span-2.

and while they are domain dependent, they are *not* ad hoc; the strategies described below are valid for all bridge structures and some may be generalized for a range of designs.

Context-sensitive considerations. The information available to the system for deciding which variable should be weakened is sometimes ambiguous when multiple variables are involved. For example, suppose a designer is studying construction by crane and would accept to have at the most one span exceed 50 m. If there is a choice between two spans (span-2 or span-3 in Fig. 8)—corresponding to level 1 of Figure 4—the designer would likely choose to relax the span over the river and limit a side-span to have the largest span over the river. The obstacle or river has an important impact on the entire design. It is important that there is a geometrical interpretation of the environment or valley. In Figure 8, the system determines whether the valley is symmetrical (by comparing the center of gravity of the space occupied by the valley and the center line of the valley) and whether the obstacle is offset with respect to the center of gravity of the valley.

Constraint aggregates. In constraint-based design systems, constraints are generally implemented as numerical equalities or inequalities, and traditional constraint-driven conflict resolution strategies consider individually each numerical constraint. The dependencies between constraints are only specified at the numerical level. However, a given design rule which covers *one* design criterion may translate into *several* numerical or symbolic constraints. A semantic link exists therefore between the set of numerical constraints related to a specific criterion. In this case, satisfying partially the set of constraints may no longer meet the original requirement. For example, in Figure 9—corresponding to levels 2, 3, and 4 of Figure 4—a balance between adjacent spans requires a total proportioning of the bridge. Constraints describing proportions must be applied simultaneously. In the absence of such a concept, conflict management at this level would attempt to relax the least number of constraints resulting in awkward dimensions between span-4 and the other spans. Relaxation is performed on a single span instead of being distributed equally on all spans.

Design inertia. This strategy reflects the understandable resistance by the designers to undo work performed up to a certain stage. Early in the design, engineers accept to change values for variables more easily and the basic ranking is suitable. Gradually however, they commit themselves to values for some variables fixed earlier. For example, suppose after the first six levels of work in Figure 4, there is a conflict involving two possible alternatives or subtasks around pier-2 (see Fig. 10): decrease number-of-spans (variable fixed early) or modify position-of-pier (more recently examined variable). Designers first try to move pier-3 to the right of the environmentally sensitive zone before switching abruptly to three piers, which requires a new configuration of foundations, beams, piers, and deck.

5. LIMITATIONS

The special strategies explained above were developed after testing with design examples revealed certain weaknesses. The most important weakness was that too many important decisions were being made arbitrarily. This lead to inferior design proposals and excessive search. Another weakness of earlier implementations was that fairly late on in the design, a weakening of an early preference changed the design completely. Such changes were considered to be too drastic and as a result, the design inertia strategy was developed. We are currently testing all of these new strategies on a range of examples.

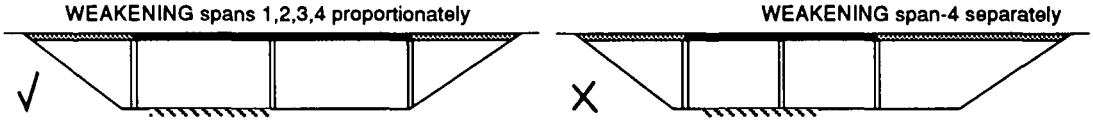

Fig. 9. Constraint aggregate—When one constraint associated with one criteria is weakened independently of closely related constraints, in this case involving aesthetic proportions, constraint linking resulting into a constraint-aggregate may improve satisfaction of the criteria: for instance, weakening the constraint-aggregate linking all spans instead of span-4 only.

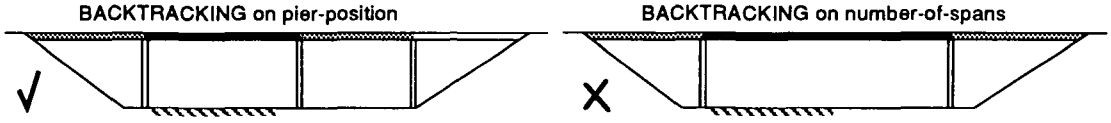

Fig. 10. Design inertia—Once several criteria have been considered and an acceptable alternative is being approximated, that is, after an accumulation of design inertia, backtracking first on recently activated variables avoids major redesign of more established variables; for example, by first backtracking on the interval of a pier before modifying number-of-spans.

Another issue is related to when the user should be consulted. Currently, the user fixes priorities according to design criteria (linked to rule sets) at the beginning of each session. Whereas dropping defaults and backtracking on fixed constraints needs little interaction, weakening of preferences and use of special strategies requires user monitoring and control. Indeed more explicit and understandable control structures would improve functionality. These issues are under study. Finally, we need to increase our efforts to validate the system. Our goal is not to create an automatic design system. Instead, we aim to provide a framework for interactive design. Although designers have been consulted periodically throughout this project, we are not yet able to provide strong evidence that design engineers find that the strategies for conflict management used in this system correspond to the ways they *prefer* to carry out design tasks. We are currently implementing these strategies with an interface that will be suitable for testing with designers.

6. CONCLUSIONS

Default and preference assumptions determine a framework for developing conflict management strategies, which are compatible to those used by designers in practice. Assumptions are an essential part of knowledge used during preliminary stages of engineering tasks. A categorization of assumptions into *default* and *preference* is a useful means for guiding processes in situations of incomplete knowledge. In a search space of an unbounded size, this categorization allows efficient derivations of relevant solutions that lead to a feasible structure. The combination of adequate search strategies based upon assumptions, with constraint satisfaction mechanisms, offer much potential for application to intelligent design because they are capable of covering two major aspects: 1) supporting design processes and 2) consistent numerical representation of artifacts.

ACKNOWLEDGMENTS

Funding for this research was provided by the Swiss National Science Foundation. The authors thank Boi Faltings for useful discussions. Also, the contribution of Dr. Peter Buckland, Buckland and Taylor Consulting Engineers, North Vancouver with respect to the knowledge used for the design example described in this paper is gratefully recognized. Finally, discussions with numerous design engineers in Switzerland have been invaluable. The authors acknowledge the Swiss Federal Highway Bureau, The Swiss Railways, Hartenbach & Wenger Engineers, and Bonnard & Gardel Engineers.

REFERENCES

Baykan, C.A., & Fox, M.S. (1992). WRIGHT: A constraint-based spatial layout system. In *Artificial Intelligence in Design*, (Tong and Siriam, Eds.), Vol. 1, pp. 395–432. Academic Press, Inc., New York.

Bowen, J., & Bahler, D. (1992*a*). Supporting multiple perspectives. A constraint-based approach to concurrent engineering. In *Artificial Intelligence in Design '92*, (Gero, J.S., Ed.), pp. 85–97. Kluwer Academic Publishers.

Bowen, J., & Bahler, D. (1992*b*). Frames, quantification, perspectives and negotiation in constraint network for life-cycle engineering. *Int. J. Artif. Intell. Eng. 7(4)*, 119–226.

De Kleer, J. (1986). An assumption-based truth maintenance system. *Artif. Intell. 28*, 127–161.

Doyle, J. (1979). A truth maintenance system. *Artif. Intell. 12*, 231–275.

Faltings, B., Haroud, D., & Smith, I. (1992). Dynamic constraint propagation with continuous variables. In *Proc. Tenth Europ. Conf. Artif. Intell.*, 754–758. Bernd Neumann, Vienna.

Haroud, D., & Faltings, B. (1994). Global consistency for continuous variables. In *Proc. Eleventh Europ. Conf. Artif. Intell.*, pp. 115–119. Amsterdam.

Krishnan, V., Navinchandra, D., Rane, P., & Rinderle, J.R. (1990). Constraint reasoning and planning in concurrent design. Tech. Rep. CMU-RI-TR-90-03, Carnegie Mellon University.

Logan, B.S., Corne, D.W., & Smithers, T. (1992). Enduring support. In *Artificial Intelligence in Design '92*, (Gero, J.S., Ed.), pp. 433–454. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Mackworth, A. (1977). Consistency in networks of relations. *Artif. Intell. 8*, 99–118.

Mittal, S., & Falkenheiner, B. (1990). Dynamic constraint satisfaction problems. In *Proc. Eighth Nat. Conf. Artif. Intell.*, pp. 25–32. Boston.

Petrie, C.J. (1991). Context maintenance. In *Proc. Ninth Nat. Conf. Artif. Intell.*, pp. 288–295. Anaheim.

Sham, S.H.R. (1993). Nonmonotonic reasoning in design. *J. Comput. Civil Eng. 7(1)*, 36–53.

Djamila Haroud is currently a Ph.D. candidate in the Artificial Intelligence Laboratory at the Swiss Federal Institute of Technology, Lausanne (EPFL). Her research interests include constraint checking techniques for continuous domains, over-constrained systems, and dynamic constraint satisfaction problems in general. She received the B.S. degree from the University of Sciences and Technology, Algiers, and the M.S. degree from EPFL in 1987 and 1990, respectively.

Sylvie Boulanger obtained her B.Sc. in Civil Engineering from the University of Alberta, Canada, and her M.S. from the University of California at Berkeley. She worked over 5 years for the Canadian Institute of Steel Construction and is currently doing a Ph.D. in the area of Knowledge Systems at ICOM (Steel Structures), EPFL through collaboration with the Artificial Intelligence Laboratory. She also had a 3-month appointment at the Key Centre for Design Computing in Sydney in 1995.

Esther Gelle received a M.S. from EPFL in computer science in 1993. She is currently a Ph.D. candidate in the Artificial Intelligence Laboratory at EPFL. Her research interests include constraint satisfaction with continuous variables and control strategies for dynamic constraint satisfaction problems.

Ian Smith obtained his B.A.Sc. in Engineering at the University of Waterloo, Canada in 1978 and his Ph.D. at Cambridge University, United Kingdom in 1982. He has been involved in various aspects of structural design since 1975. He has also consulted with industry on knowledge systems and structural design in Europe, North America and Japan. After nine years at ICOM, EPFL where he formed a knowledge systems group, he joined the Artificial Intelligence Laboratory as the Adjoint Director in 1991. He is the current Chair of the European Group for Structural Engineering Applications of Artificial Intelligence (EG-SEA-AI).