From: AAAI-90 Proceedings. Copyright ©1990, AAAI (www.aaai.org). All rights reserved.

Parametric Engineering Design Using Constraint-Based Reasoning

Niall Murtagh and **Masamichi Shimura**
Department of Computer Science, Tokyo Institute of Technology
2-12-1 O-okayama, Meguro-ku, Tokyo 152, JAPAN
email: niall@cs.titech.ac.jp

Abstract

Conventional methods for the parametric design of engineering structures rely on the iterative re-use of analysis programs in order to converge on a satisfactory solution. Since finite element and other analysis programs require considerable computer resources, this research proposes a general method to minimize their use, by utilizing constraint-based reasoning to carry out redesign. A problem-solver, consisting of constraint networks which express basic relationships between individual design parameters and variables, is attached to the analysis programs. Once an initial design description has been set out using the conventional analysis programs, the networks can then reason about required adjustments in order to find a consistent set of parameter values. We describe how global constraints representing standard design behavioral equations are decomposed to form binary constraint networks. The networks use approximate reasoning to determine dependencies between key parameters, and after an adjustment has been made, use exact relationship information to update only those parts of the design description that are affected by the adjustment. We illustrate the ideas by taking as an example the design of a continuous prestressed concrete beam.

1 Introduction

Parametric design refers to the design of engineering objects, in which the parameters and variables describing the object are known, and the problem is one of finding a consistent set of parameter values which conform to specified requirements. Conventional parametric design algorithms for civil and mechanical engineering structures use finite element and other structural analysis programs to propose an initial solution. If the solution is infeasible, redesign is carried out in iterative fashion in order to converge on a feasible solution. The analysis programs have to be treated as black-boxes with fixed input and output, and hence each redesign iteration requires a complete re-processing, irrespective of how small the change made to the previous design description. Furthermore, these analysis programs are often large and require considerable computer resources [Murthy and Addanki, 1987]. Therefore, any reduction in their usage through the application of intelligent redesign methods is advantageous for design efficiency.

Previous research into structural engineering design has dealt with meta-level control [Orelup et al, 1988], domain independence [Dixon et al, 1987], and innovative design [Murthy and Addanki, 1987], but has not applied intelligent technology to the fundamental analysis part of parametric design. In this research, we replace the analysis programs in redesign with a knowledge-based constraint reasoning process. This provides the double advantage of minimizing the use of expensive analysis programs, and of being able to update only those parameters or variables in the design description which require updating.

Constraint-based techniques have been shown to improve the problem-solving capabilities for applications such as combinatorial problems and vision [Montanari, 1974], [Mackworth, 1977], electrical circuit analysis [Sussman and Steele, 1980], and preliminary structural design [Sriram and Maher, 1986]. A constraint-based approach reflects the view that design is essentially a process of integrating constraints from a variety of sources [Mostow, 1985]. In the present system, however, the constraints considered are restricted to those that can be conveniently handled, i.e., numeric constraints on design parameter values. We explain how constraint networks are formed for the type of application considered here, and we propose an architecture to accommodate a constraint-based problem solver together with structural analysis programs. The actual design process is then detailed, showing how backward reasoning enables dependencies between key parameters to be determined, and how forward propagation through the constraint network obviates the need to re-access analysis modules. We illustrate our techniques by referring to the design of a statically indeterminate prestressed concrete beam.

2 Design Process Model

A conventional parametric design problem is described by parameters and variables, which can be classified according to the following three groups:

Initial Parameters These include the user specification together with certain other unknown parameters whose values must be estimated before processing can be carried out to determine performance parameter values.

Intermediate Variables These are determined as part of the processing of the initial parameters.

Design Performance Parameters These are provided either by the user or by engineering codes, and they determine whether the solution proposed by the initial parameter values is adequate or not.

Since it is not possible to directly determine the unknown parameters from the specification and required performance, conventional design is carried out in an iterative generate-and-test manner, using analysis programs to generate a complete description, before testing the performance parameters. The analysis programs, cannot be fully replaced by more flexible reasoning systems, such as constraint-based problem solvers, because the analysis programs have certain capabilities–for example, the handling of various data types–which constraint-based reasoning systems do not possess, e.g., analysis programs process both single and multi-valued parameters, such as arrays of loads on a multi-span beam, while constraint-based systems can only reason about single valued parameters. Therefore, we use both conventional and constraint-based processing of parameters. Our system initially follows the conventional algorithm using analysis programs to generate an initial complete design description. However, in subsequent redesign, only extreme values need be considered for each parameter, so that constraint-based reasoning alone can complete the design problem.

3 Constraint Network Model

A standard constraint-satisfaction problem is characterized by a set of variables or parameters, each of which has a domain, finite or infinite, of possible values. To this is added a set of domain constraints and a set of inter-parameter constraints.

Domain Constraints are unary constraints, $c(p)$, where c is a function mapping a parameter, p, onto a particular domain of allowable values. In our model, these constraints act on the initial and performance parameters. Intermediate variables do not have unary constraints and will accept any value propagated to them. Two levels of unary constraint exist:

- Unary *limit*-constraints restrict the parameter to a multi-value domain within fixed limits, which cannot be adjusted during design. They may be obtained from a domain expert or textbook, or may be specified by the user.
- Unary *value*-constraints assign a single numerical value to each initial parameter, i.e., they further tighten the limit-constraints so that a single value is associated with the parameter prior to propagation. These value-constraints may be supplied either by the user as part of the design specification, in which case the limit-constraints ensure that the value is in the allowable domain, or they may be supplied by the system, through default or other methods. System-supplied unary value-constraints may be automatically adjusted during redesign, whereas user-supplied constraints are considered as part of the specification and may not be altered.

Inter-Parameter Constraints are "n-ary" constraints, $c(p_1, p_2 ..... p_n)$, where c is a function relating parameters $p_1, p_2 ..... p_n, n \geq 2$. These constraints, representing the laws of statics, geometry, mathematics, etc., link all the parameters/variables in the design description. They are used to propagate the value-constraints on the initial parameters, i.e., they force an "n-ary" constraint between the initial and performance parameters. If the constraints cannot be satisfied, adjustments have to be made. This is where the manner of representation of the inter-parameter constraints can have a big influence on the efficiency of the design. We now describe the treatment of these constraints.

3.1 Constraint Decomposition

In conventional engineering design methods, global inter-parameter constraints are represented by behavioral equations contained in the analysis programs which process the input description and check the result against performance requirements. This can be regarded as applying an "n-ary" constraint to the initial and performance parameters, where n is the number of parameters involved.

We convert these "n-ary" constraints into binary constraint networks for the following reasons:

- In conventional methods which use "n-ary" constraints as in Figure 1(a), specific or generalized dependency relations between initial and performance parameters must be explicitly stated, e.g., factor × imposed loading = tensile stress. By linking all parameters together in a binary network, part of which is shown in Figure 1(c), approximate dependency relations can be worked out by the system.
- With a few "n-ary" constraints linking many parameters, as in Figure 1(a), if one parameter is adjusted then a complete re-computation involving all n parameters, is generally required to determine the adjustment necessary for the other parameters. The "n-ary" constraint gives no information regarding

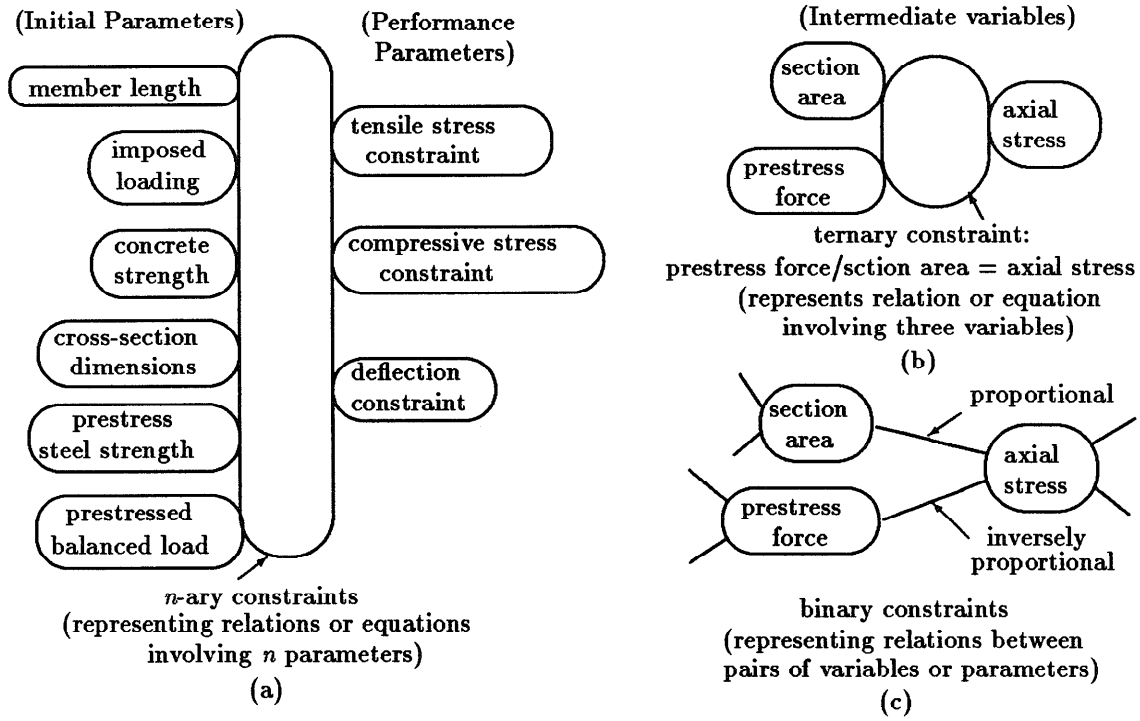


Figure 1: Constraint Decomposition

individual parameter-to-parameter relations, so that partial re-adjustment of the design description is not possible. However, a binary constraint network directly relates individual parameters, enabling local adjustments to be carried out and propagated only to those parameters affected by a change.

The "n-ary" constraints as in Figure 1(a), are decomposed by firstly introducing intermediate variables as "stepping-stones" in the propagation process, i.e., we explicitly represent all steps in the evaluation of the behavioral equations as nodes in the constraint network. This reduces the "n-ary" constraint to simpler forms, e.g., the ternary constraint shown in Figure 1(b), which processes two variable-values to produce one new variable-value. Secondly, these simpler constraints are then further decomposed into binary constraints, or relations between pairs of nodes. After the first step the constraints still represent equations, as in Figure 1(b), but after the second, we only have one-to-one relations between nodes as in Figure 1(c), and we can directly transfer an adjustment from one node to another, without having to consider any more than two nodes at a time. In our system binary relations similar to those shown in Figure 1(c), cater for addition, multiplication, etc., and power operators, as well as the reverse operators. Certain complications that occur in the adjustment transfer process are dealt with in section 4.1.

3.2 System Architecture

An expert system such as the one proposed here will generally be built on top of existing procedural programs, not the other way around. Therefore, it is necessary to adapt constraint-based ideas to suit the demands of conventional programs. As shown in Figure 2, the architecture consists of a 2-level structure, with the constraint network set out in the upper, shallow-knowledge level, and conventional analysis modules located in the lower, procedural knowledge level. The links between nodes in the upper level consist of simplified direct dependency relations, as explained in the previous section, while those which pass through the lower level utilize traditional software tools, such as finite element programs, etc. Unary constraints are shown acting vertically down on the upper level. This separation between top and bottom levels reflects the distinction in conventional engineering terminology between "design" and "analysis". Design concerns initial decisions, outline data, etc., and uses heuristics or shallow knowledge, while analysis concerns precise numerical values and uses procedural knowledge embedded in algorithmic programs. User communication and system output take place through the upper design level, which is also used in redesign to carry out propagation in forward and backward directions. The lower level analysis modules are used in setting out a full initial description, and typically link several nodes together,

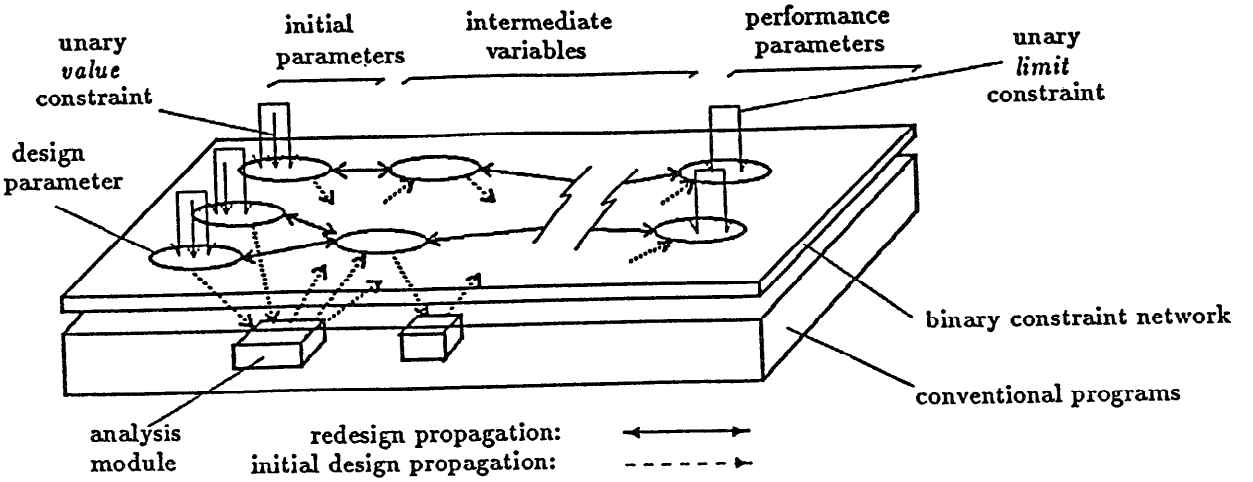


Figure 2: System Architecture

i.e., they form "n-ary" constraints, which, for clarity, we illustrate by dotted-line arrows.

4 Problem-solving Process

4.1 Application of Constraints

A design problem is commenced by the application of unary constraints. Allowable domains and specific values are assigned to the initial parameters, and allowable domains are assigned to the performance parameters. The user specification provides some of these unary constraints and the remainder are supplied by the system. This is done either through system-dependent default values, or through constraint derivation, which generalizes standard textbook heuristics linking certain parameters, e.g., span/depth ratios. (For details of the constraint derivation algorithm, see [Murtagh and Shimura, 1989]).

The values assigned to the initial parameters are then propagated using the "n-ary" constraints. This resembles standard propagation except that in the initial propagation the lower level algorithmic modules are used and multiple parameter values are simultaneously operated on. A complete description is thus obtained, and its validity is tested by the performance parameters. Generally this first complete description will be invalid due to conflict between the propagated constraint values and the allowable domains of the performance parameters, and a redesign phase is automatically commenced.

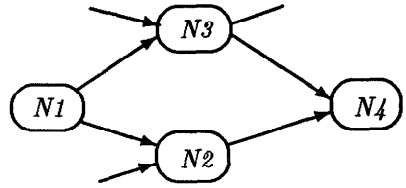
In the redesign phase, the system first determines the dependency information necessary for redesign, i.e., it back-propagates from the critical performance parameter to determine how much the initial parameters should be adjusted in order to achieve the required alteration in the critical performance parameter. Thus, it obviates the need to set out parameter-to-parameter dependency information, although parameter ordering [Dixon et al, 1987], specifying which parameter to adjust first, is still required.

After the adjustments required in the initial parameters have been established, one initial parameter is selected for adjustment in accordance with the current redesign strategy. The adjustment is carried out, and forward propagation through the network updates the nodes related to the adjusted initial parameter.

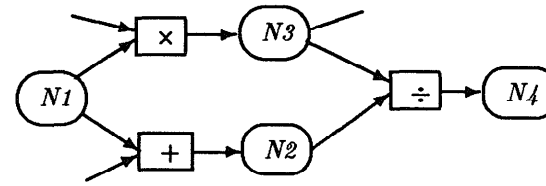
While the initial forward propagation can be regarded as a global propagation, the redesign phase is based on local propagation. Local propagation has the advantage that it is fast because it uses only simple relationships and deals only with single-valued nodes. However, it relies on approximate reasoning in back-propagation, due to the presence of loops or cycles in the network. Referring to Figure 3(a), in back-propagation where more than one link lead out of the node *N4*, the required adjustment in *N4* is propagated to both *N2* and *N3*, an adjustment in either one being sufficient to produce the required adjustment in *N4*. Where the two paths subsequently lead into a single node *N1*, a cycle or loop occurs. In general, the two adjustments back-propagated to *N1* along the two paths, will not be the same. In order to combine the values, we propose an empirically derived approximate equivalence equation, supplemented with a self-updating heuristic correction factor, F.

proposed adjustments		executed adjustment
x, y	$\Longrightarrow$	$F \times [x \times y/(x+y)]$

The heuristic factor, initially set to 1, and updated after adjustment, is used to speed up the adjustment process, i.e., it attempts to move immediately to the top of the current hill. This factor is problem-specific



(a) Node relations represented by binary links



(b) Node relations represented by operators

(Arrows show direction of forward propagation)

Figure 3: Loop in Binary Constraint Network

Design Parameter Values		
parameter name	*user specification*	
no of spans	3	
span lengths	10 *m*, 18 *m*, 10 *m*	
end fixities	cantilever, pinned	
dead load	5 *kN/m*	
live load	6 *kN/m*	
	computed parameters	
	initial value	*final value*
concrete strength	35 N/mm^2	40 N/mm^2
section width	300 *mm*	450 *mm*
section depth	650 *mm*	800 *mm*
prestress steel strength	1950 N/mm^2	1950 N/mm^2
cover to steel †	30 *mm*	103 *mm*
balanced load factor ‡	0.4	0.486

†Steel cover determines max prestress tendon sag.
‡Proportion of load balanced by prestress.

Table 1: Design Description for Prestressed Beam

and path-specific. Hence, although it has no usefulness in the first adjustment, since its value is 1, it performs better than over-generalized statistical factors and will enable convergence to be quickly obtained in a subsequent adjustment of the same parameter.

In forward propagation, where multiple adjustments are passed to a single node, they are super-imposed to give the exact required adjustment. The difference between forward and backward propagation, while not obvious from Figure 3(a), can be seen more clearly in Figure 3(b), which shows the links passing through operators, which act like ternary constraints.

5 Implementation and Results

The system at present comprises four partially overlapping constraint networks, each network joining the initial parameters with one particular performance parameter. These networks are written in Prolog, while the structural analysis modules are in C. The analysis modules in the current system carry out bending moment and shear computations, prestress system computations, and ultimate strength verification. Table 1 shows the numeric data for the design of a 3-span prestressed concrete beam. The design was carried out using the constraint network for redesign and also using conventional techniques, i.e., using the analysis programs only. The same results were obtained for each case, since the constraint networks exactly reflect the processing of the analysis programs. However, the conventional design method necessitated accessing each of the analysis programs a total of 18 times, while the constraint-based method only required the use of the analysis programs once, in the initial design.

6 Comparison with Related Research

Relaxation [Leler, 1988] involves making an initial guess at the value of an object, estimating the resultant error, and making new guesses in order to converge on a rational value. We have adapted this technique to the demands of engineering design, focusing on how to represent relations between the objects, and how to improve the convergence methods.

In [Chan and Paulson, 1987], constraints are used for determining design descriptions as well as checking proposed descriptions for a structural engineering application. However, procedures are required for each intended use of the constraint. We also use constraints in different ways, but instead of setting out numerous procedures, we avail of simplified relationships represented by constraint networks. One of the ways in which we utilize the constraints, in addition to checking, is in establishing parameter-to-parameter dependency information, using local constraint relationships together with problem-specific heuristic factors. Thus we do not need to set out explicit dependency relationships as in more conventional systems [Dixon et al, 1987].

In [Mackworth, 1977] consistency algorithms are proposed for attaching feasible values to variables, and for filtering infeasible values in constraint networks.

Only unary and binary constraints are treated, and the inter-variable constraints are assumed to be inexpensive, unlike those represented by analysis programs in engineering design. Hence, our research focuses on the nature of the inter-variable constraints.

In [Dechter and Pearl, 1987], the variable-domains consist of a finite number of discrete values, enabling candidate-ordering and/or elimination to be used. These techniques cannot be used in engineering design where the parameters normally have continuous and therefore infinite domains. However, the continuous nature of the domain values permits movement from one candidate value to another, using only information concerning the extent to which the previous value was unsatisfactory, thus avoiding expensive re-computation procedures.

7 Conclusion

This system has shown how a constraint-based problem solver can improve the efficiency of parametric engineering design, by minimizing the use of large analysis programs and updating only the minimum number of parameters and variables in redesign. The fundamental difference between previous work and the present system is the following: in conventional systems the redesign process releases all intermediate variables from their currently assigned values, and after some alteration to the initial parameter(s), re-assigns these variables using the same methods as before. In our system, redesign does not release variables from their current values, but instead automatically adjusts the values of those variables related to the most critical performance parameter, by propagation through the constraint network. Forward propagation is exact, depending only on the binary constraints between the parameters and variables. Back propagation is generally approximate, due to loops or cycles in the constraint networks, and relies on *ad hoc* procedures. At best this will give exact accuracy (for single path, or no-loop dependencies) and good approximations for other multi-path dependencies. At worst, it will rely on the heuristic correction factor to steer the adjustment in the right course. The system at present uses a general hill-climbing strategy. A parametric design system should ideally have a variety of strategies, as shown in [Orelup et al, 1988]. However, this is a separate issue to the one considered in the present research.

The strategy used in this work is applicable to parametric design problems in which basic relationships between parameters and variables can be obtained from the behavioral equations, and in which multi-valued parameters can be rationalized into controlling single values for reasoning purposes. Although the system has been tested in only one domain, parametric design does not differ radically from domain to domain, so that our strategy has the potential to be applied generally.

References

[Brown and Breau, 1986] Brown, D.C., Breau, R., Types of Constraints in Routine Design Problem-Solving, in *Applications of Artificial Intelligence in Engineering Problems, Proceedings of the 1st International Conference*, Southampton U.K., 1986, Springer-Verlag.

[Chan and Paulson, 1987] Chan, W.T., Paulson, B.C. Jr, Exploratory Design using Constraints, in *Artificial Intelligence in Engineering Design, Analysis and Manufacturing* 1, 1987, pp. 59-71.

[Dechter and Pearl, 1987] Dechter, R., Pearl, J., Network Based Heuristics for Constraint-Satisfaction Problems, in *Artificial Intelligence* 34, 1987, pp.1-38.

[Dixon et al, 1987] Dixon, J.R., Howe, A., Cohen, P.R., Simmons, M.K., Dominic I: Progress Toward Domain Independence in Design by Iterative Redesign, in *Engineering with Computers* 2, 1987, pp. 137-145.

[Leler, 1988] Leler, Wm., *Constraint Programming Languages, their Specification and Generation*, 1988, Addison-Wesley Publishing Company.

[Mackworth, 1977] Mackworth, A., Consistency in Networks of Relations, in *Artificial Intelligence* 8, 1977, pp. 99-118.

[Montanari, 1974] Montanari, U., Networks of Constraints: Fundamental Properties and Applications to Picture Processing, in *Information Sciences* 7, 1974, pp. 95-132.

[Mostow, 1985] Mostow, J., Toward Better Models of the Design Process, in *AI Magazine* 6, 1985, Spring.

[Murtagh and Shimura, 1989] Murtagh, N., Shimura, M., A Constraint-Based Hybrid Engineering Design System, in *Proceedings of The Third IFIP WG 5.2 Workshop on Intelligent CAD*, 1989, Osaka, Japan, (to be published by North-Holland).

[Murthy and Addanki, 1987] Murthy, S., Addanki, S., PROMPT: An Innovative Design Tool, in *Proceedings of AAAI*, 1987, pp. 637-642.

[Orelup et al, 1988] Orelup, M.F., Dixon, J.R., Cohen, P.R., Simmons, M.K., Dominic II: Meta-Level Control in Iterative Redesign, in *Proceedings of AAAI*, 1988, pp. 25-29.

[Sriram and Maher, 1986] Sriram, D., Maher, M.L., The Representation and Use of Constraints in Structural Design, in *Applications of Artificial Intelligence in Engineering Problems, Proceedings of the 1st International Conference*, Southampton U.K., 1986, Springer-Verlag.

[Sussman and Steele, 1980] Sussman, G.J., Steele, G.L. Jr, Constraints –a Language for Expressing Almost Hierarchical Descriptions, in *Artificial Intelligence*, 14, 1980 pp. 1-39.