



XCODAMS: An engineering design system based on constraint propagation

E.K. Burke, D.G. Elliman & M.I. Heard

Department of Computer Science, University of Nottingham, University Park, Nottingham, NG7 2RD, UK

ABSTRACT

The investigation of engineering design problems inevitably involves searching through extremely large solution spaces. The designer has to make decisions about the product that is being designed so that it satisfies a large amount of constraints. These constraints are not limited to engineering equations but range from fundamental formulae to financial restrictions. The problem of automatically finding solutions to these problems is nearly always intractable because of the size of the search spaces. The designer necessarily adopts a strategy of intuitive guesswork which is unlikely to result in an optimal solution.

In this paper, we describe a powerful and useful engineering design system based on constraint propagation methods. It can quickly explore and reduce solution spaces. It is assumed that the user will have a good idea of the kind of solution required and can add and remove constraints in an informed manner to work with the system and *home* in on a suitable solution.

INTRODUCTION

A wide variety of engineering problems can be expressed by a set of variables which are subject to a set of constraining relations. We associate with each variable a particular domain (c.f. [1],[2]), or range of numbers. A valid solution is an instantiation of the variables such that all the constraints are satisfied.

A constraint satisfaction problem is concerned with finding solutions to systems of constraints. However, constraint satisfaction problems are, in general, NP-complete. That is to say, it is very strongly believed that there is no *short cut* to an optimal solution - in principle all possibilities have to be examined.

It is possible in many cases to reduce the domains (or search spaces) by omitting certain elements which are inconsistent with the set of constraints. Thus we obtain reduced domains which contain all the solutions but may also contain extra unwanted values. This technique is known as constraint propagation [1],[2]. However, constraint propagation is itself intractable.

This problem is overcome by allowing the user to interact with the machine, to backtrack, to add and remove constraints, to manually alter the domains and so work with the system to find a reasonable solution to many diverse engineering problems. By reasonable solution we mean one that satisfies the user, rather than an optimal one. We are assuming that the user has a good knowledge of the engineering design problem that is being worked upon and that the user is capable of guiding the system to such a solution and of recognising such a solution.

602 Artificial Intelligence in Engineering

This approach could be described as *informed search* and is much more likely to *home* in on a near optimal solution than the combination of guesswork and estimation which was all that was available before.

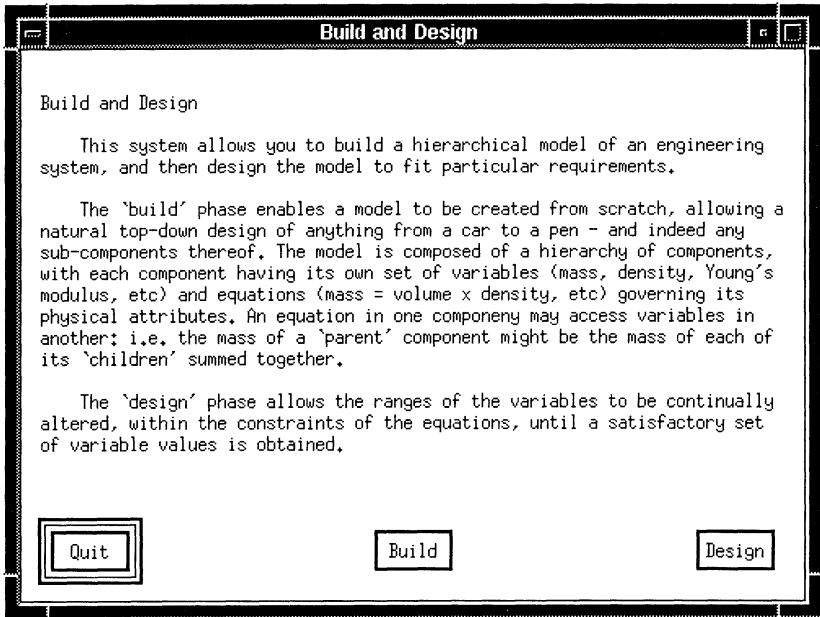


Fig.1

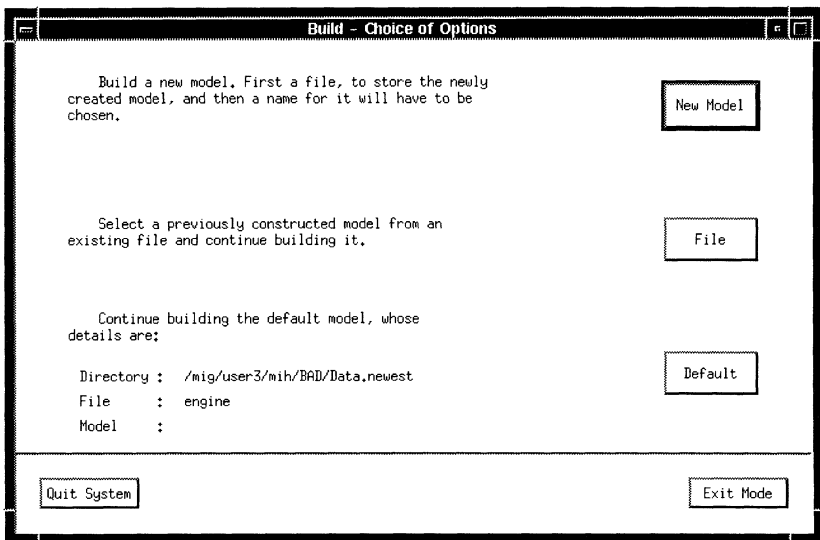


Fig.2

A DESCRIPTION OF THE SYSTEM

When first executing the XCODAMS system, the user is presented with the interface seen in fig.1. This presents a short textual introduction to the program and to the principle of modelling a system using parametric equations. The user is also presented with two choices at this stage. These are to allow the user to build a model or to design a model. Building a model consists of creating the variables and equations that describe a particular engineering problem. It is, in fact, the process of entering a problem into the system. Designing a model consists, essentially, of choosing (with the aid of the system) ranges of values for the variables within the constraints that define the particular problem being tackled.

When building a model, the user is faced with the choice of starting to build a new model or of continuing to build an existing model, as shown in fig.2. When building a new model, a file and a model name has to be chosen. The interface that helps the user choose the required file is shown in fig.3.

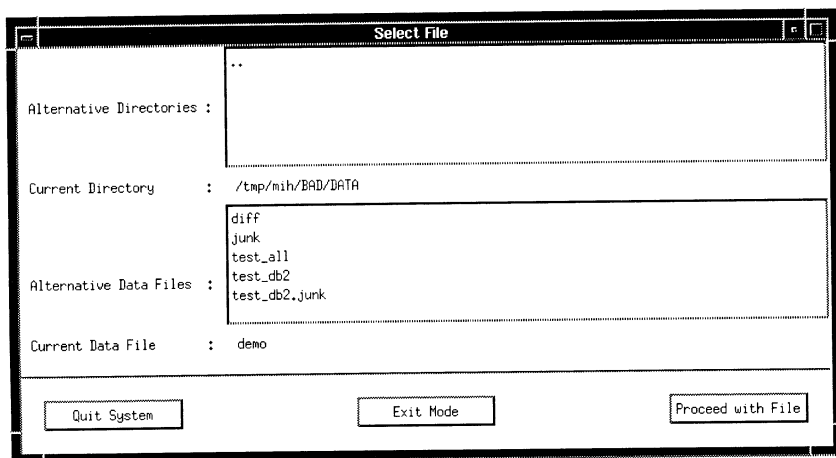


Fig.3

Having chosen the model, the user will move on to the task of building. The interface used is shown in fig.4. It consists of three separate windows, each of which operates on disparate aspects of model building. The "control window", at the top, is the focal point of the user's overall view of the model. It displays information on the model as a whole - the file in which it will be saved and its name - and allows the user to verify the model and save it to file. Verification is required before any model can be saved to file, as the design process, which works on the same model files, requires correctly entered equations and variables. Use of the "Verify" button will cause all errors in the current model to be reported to the user. These errors have to be corrected before the model can be saved to file. Verification is implicitly performed on the model before it is saved.

The variables and relations that define the model, or part of it, are shown in the "Primary Component" window on the bottom right. An example window, showing some of the variables and relations describing a beam model, can be seen in fig.6.

The diagram shows the variable declaration `den,0,_,0,kgm3` with five annotations pointing to its components:

- Variable Name**: Points to `den`.
- Minimum value of the variable**
(`_ == negative infinity`)
(`_ == positive infinity`) : Points to the underscore `_`.
- Maximum value of the variable**
(symbols same as for the minimum value) : Points to the first `0`.
- The increment:**
the variable's value is measured in increments of this value : Points to the second `0`.
- The units in which the variable is measured** : Points to `kgm3`.

Fig.5

In fig.6 the only relational operator that occurs is *equality*. However, the system can handle any relational operator that typically occurs in engineering design problems, for example, *less than* or *set membership*. The variables can be seen in the left hand *subwindow* of fig.6 The construction of a typical variable is presented in fig.5.

Note that a variable does not ordinarily have a single value, but a maximum and a minimum denoting a range of values for which that parameter is valid within the current problem. The main purpose of the system is to efficiently reduce these ranges. If the maximum and minimum elements of the range are reduced to identical values then the variable is taken to have that (single) value.

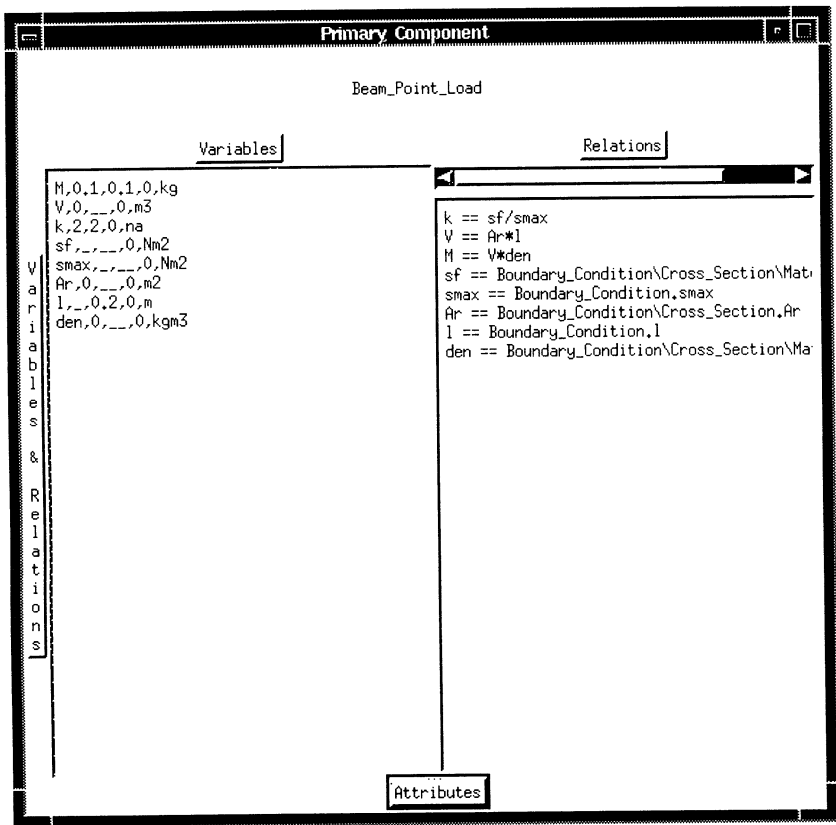


Fig.6



606 Artificial Intelligence in Engineering

When modelling simple systems, all the necessary variables and relations can be naturally grouped together and all can be simultaneously viewed with ease. For more complex models, such as the beam example presented here, the information is better viewed as a hierarchy of components or "nodes". Fig.7 shows the "Hierarchy Display" (on the bottom left of fig.4) for the beam model. The top node describes the overall beam. The "Boundary_Condition" is it's *child* node which describes the conditions on the end of the beam. The "Cross_Section" is the *grandchild* node which describes the cross-sectional geometry of the beam. The *great-grandchild* node "Material" (not on screen) describes the material from which the beam is constructed. Variables from one node may be referenced by any other, by using its *path name* which is analogous to a file's path name on UNIX, MS-DOS or any other operating system which uses a hierarchical directory structure.

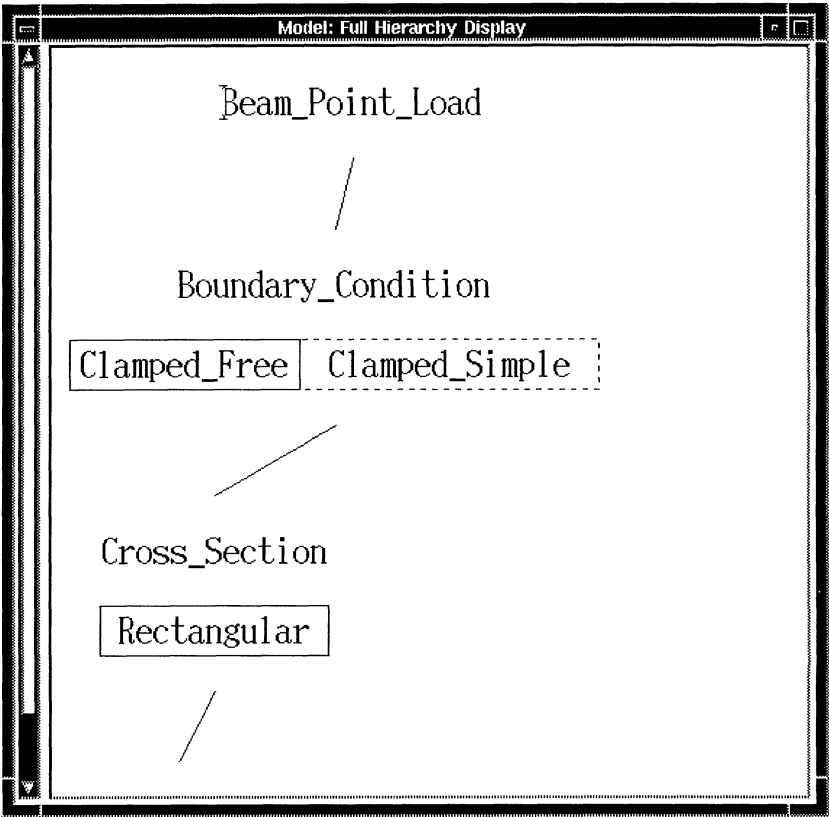


Fig.7

Some system parameters cannot be described easily by a variable with a numerical range value. For example, the cross-sectional geometry of the beam is most easily described using words such as *rectangular*, *square*, *circular*, *I-beam*. Furthermore, different equations may be valid for different values of the geometry parameter. This context-switching is allowed for by the provision of *optional nodes*. An optional node is similar to an ordinary node, in that it has variables and relations. However, it differs in that it can have no children and that none of its variables may be accessed by any other node, ordinary or otherwise. An optional node is grouped with an ordinary node, which may possess any number of optional nodes but only one of which may be *chosen*. Only the variables and relations from the chosen optional node will be included when designing the model.

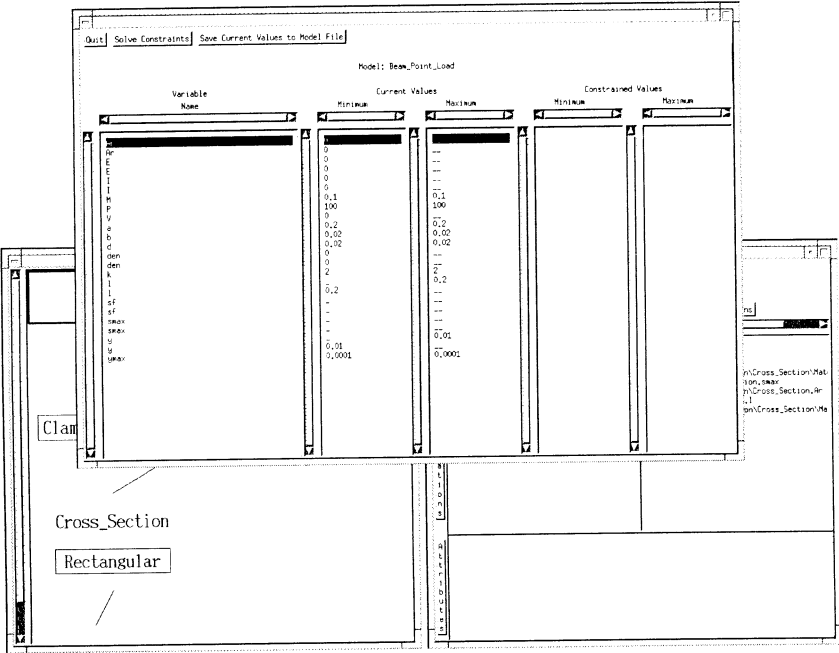


Fig.8

The designing of a model is accomplished by navigating a series of windows, similar to those that preface the build interface, and which will result in reaching the design interface presented in fig.8. This has a *Primary Component* window and a *Hierarchy Display* window (bottom right and bottom left respectively), which behave in exactly the same manner as the build versions except that no editing is allowed. The main window displays *all* the variables which occur in the problem, including those from the *chosen* optional nodes, and their current values. These values can be changed manually (if the user so desires), allowing the user and the system to explore different possibilities and different aspects of the problem under consideration. When required, the *Solve Constraints* button can be used to find a *solution* to the problem. The solution is displayed using the *Constrained Values* columns on the right side of the window, as can be seen in fig.9. This solution may, or may not be, suitable. It is up to the user to decide. If the solution is not suitable, the user can continue to explore the problem.



The screenshot shows a window titled "Design" with a menu bar containing "Quit", "Solve Constraints", and "Save Current Values to Model File". Below the menu bar, the text "Model: Beam_Point_Load" is displayed. The main area contains a table with four columns: "Variable Name", "Current Values", "Constrained Values", and "Maximum". The table lists various variables and their corresponding values.

Variable Name	Current Values		Constrained Values	
	Minimum	Maximum	Minimum	Maximum
Ar	0	---	0.0004	0.0004
E	0	---	2e+11	2e+11
L	0	---	2e+11	2e+11
M	0	---	1.33333e-08	1.33333e-08
P	0	---	1.33333e-08	1.33333e-08
V	0.1	0.1	0.1	0.1
a	100	100	100	100
b	0	---	8e-05	8e-05
d	0.2	0.2	0.2	0.2
den	0.02	0.02	0.02	0.02
den	0.02	0.02	0.02	0.02
k	0	---	1250	1250
l	2	2	2	2
sf	0.2	0.2	0.2	0.2
sf	---	---	3e+07	3e+07
snax	---	---	3e+07	3e+07
snax	---	---	1.5e+07	1.5e+07
y	---	---	1.5e+07	1.5e+07
y	0.01	0.01	0.01	0.01
y	0.0001	0.0001	0.01	0.01
y	---	---	0.0001	0.0001

Fig.9

CONCLUDING REMARKS

We have described an engineering design system which allows the user to define and investigate engineering problems in an efficient and powerful way.

The XCODAMS system was implemented under the UNIX operating system. Its graphical user interface (GUI) was implemented using the X window system, making the system available on a wide range of platforms including PCs, workstations and Macintoshes. All aspects of the implementation were written in the object-oriented language C++, which was used to design approximately 120 separate object classes. In practice, the implementation running on an unloaded machine was able to arrive at a solution to a system of 25 equations and 39 variables, 14 of which were set, in just over one minute; checking the results by hand took over two hours.

REFERENCES

- [1] L.Bai, E.K.Burke and D.G.Elliman, Constraint Propagation Over Ordered Domains, Proceedings of the 8th Conference on Artificial Intelligence for Applications, Monterey, California 2nd-6th March 1992 pages 35-43.
- [2] M.I.Heard, D.G.Elliman, E.K.Burke, R.Brooks, Constraint Propagation in Engineering Design, Proceedings of the Eighth International Conference on Applications of Artificial Intelligence in Engineering Vol.1: Design, Methods and Techniques, Toulouse, France 1993, pages 123-133.
- [3] M.C.Cooper, An Optimal k-Consistency Algorithm, Artificial Intelligence Vol.41, 1989-90, pages 89-95.
- [4] A.C.Mackworth and E.C.Freuder, The Complexity of Some polynomial Network Consistency Algorithms for Constraint Satisfaction Problems, Artificial Intelligence Vol.25 1985, Pages 65-74.