Symmetry Detection and Breaking in Constraint Satisfaction Problems

Andrew Grayland
andyg@cs.st-and.ac.uk

University of St Andrews

Microsoft Research Summer School 2007

Supervisors: Ian Miguel, Colva Roney-Dougal
ianm@cs.st-and.ac.uk colva@mcs.st-and.ac.uk

MSR Collaborator: Youssef Hamadi
youssefh@microsoft.com

Abstract

Constraint Programming (CP) enables computation of a wide variety of complex or time critical problems. The problem is characterised or modelled as a constraint satisfaction problem (CSP): a finite set of decision variables, each with a finite set of potential values, and a set of constraints on the allowed assignments of values to the variables. A solution is set of complete assignments that satisfies all constraints.

General methods of breaking symmetries by adding lexicographic ordering constraints can require a huge number of constraints. This adds an unacceptable overhead to the solving process [2]. In certain cases, this set can be reduced dramatically. The reduction process in itself can be prohibitively costly. We propose general formulae for the production of symmetry breaking constraints in linear time.

Symmetry

Consider the problem of finding a multi-set of coloured balls, such that each of 5 colours appears at least once, modelled as a one dimensional array of colours.

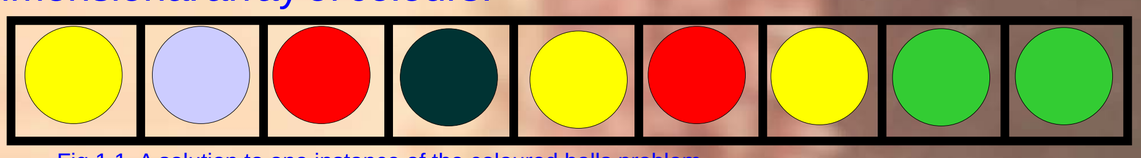

Fig 1.1. A solution to one instance of the coloured balls problem.

Since the problem requires a multi-set as the solution, modelling the solution as an array has introduced some symmetry. Clearly we can exchange any ball for any other and still have the same solution when viewed as a multiset.

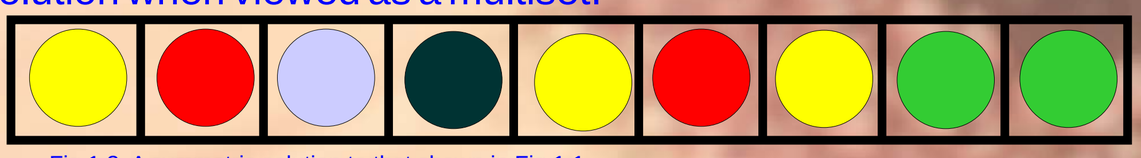

Fig 1.2. A symmetric solution to that shown in Fig 1.1.

Given any solution we can in fact exchange any ball with any other any number of times and always produce another solution. The symmetry in problems like these is known mathematically as the Symmetric Group, S_n.

Fig 1.3. A symmetric solution to that shown in Fig 1.1. and Fig 1.2 with more than one exchanging of

Fig 1.4 shows a small instance of the coloured balls problem. The values are red and blue and there are three variables. The left hand graph shows a search with no symmetry breaking, the right hand graph shows a search with symmetry breaking constraints.
Red is less than blue and a static variable ordering is used.

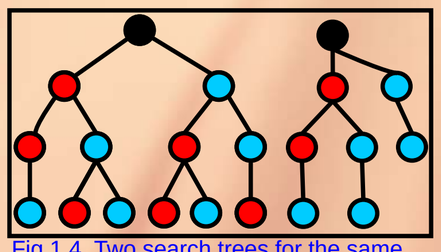

Fig 1.4. Two search trees for the same problem instance. The problem on the right has added symmetry breaking constraints.

Lex-Leader Constraints

One method to break symmetry is to add lex-ordering constraints. We first define an ordering on the variables. [1]

Fig 2.1. An ordering on the decision variables in the coloured balls problem.

Next we define an ordering on values, from left to right:

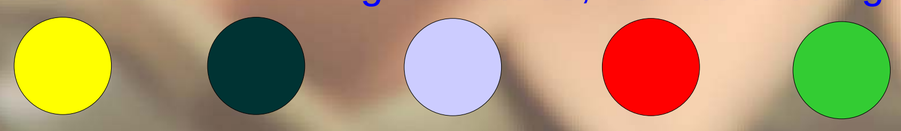

Fig 2.2. An ordering on the values in the coloured ball problem, smallest to greatest.

We then post one constraint for each permutation of the group. We can reduce the number and size of constraints by removing conditions that will already hold if they are ever tested. [2][4] Consider the following solution.

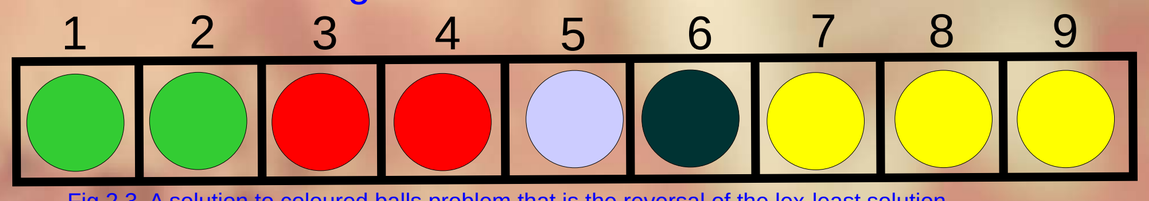

Fig 2.3. A solution to coloured balls problem that is the reversal of the lex-least solution.

To break all variable symmetries in the coloured ball problem we need only order the variables from left to right, smallest to greatest. We observe that the solution in Fig 2.3 is not the lex-least solution since the greatest value colour is first with lower value colours following. All solutions shown belong to the same equivalence class, Fig 2.4 shows the lex-least solution from that equivalence class.

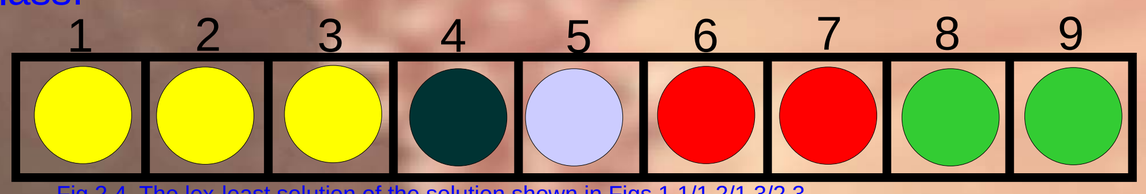

Fig 2.4. The lex-least solution of the solution shown in Figs 1.1/1.2/1.3/2.3

Symmetry Detection

Sometimes symmetries are introduced in the modelling process, as we discovered in the coloured ball problem. Unfortunately this does not usually cover all problem symmetries. Graph automorphism is a method to detect all problem symmetries in a given model. Consider a graph representing a problem over the variables *{x1,x2,x3,x4,x5,x6}*, where the edges represent constraints between the variables. We can permute the nodes whilst preserving the edges.

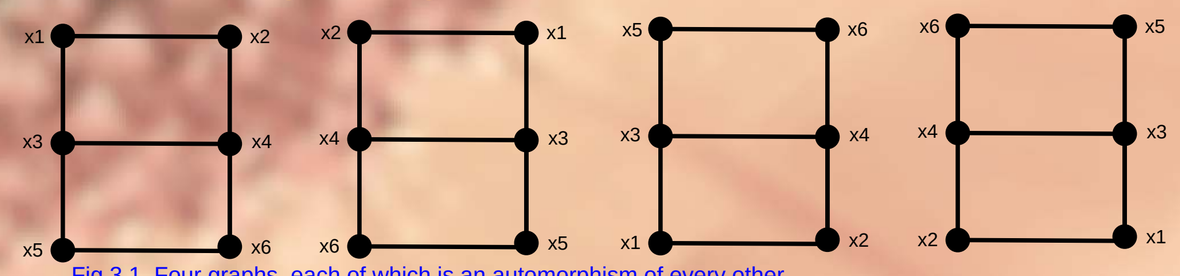

Fig 3.1. Four graphs, each of which is an automorphism of every other.

Automated tools, such as AUTOM [5], are available to automate the detection of graph autmorphisms.

Preliminary Results

For a sample problem exhibiting the symmetry group of the imprimitive wreath product of two symmetric groups the results are very promising. The minimal ordering constraints produced allow the solution of otherwise unsolvable problem instances. In many cases the production of the complete set of lex-leader constraints themselves were the prohibiting factor in finding a solution.

Future Work

The general formulae produced for commonly occurring groups combined with the detection of symmetries and deconstruction of those symmetries into products of the known general formulae will facilitate the introduction of symmetry breaking mechanisms in the automated modelling assistant, CONJURE. [3]

[1] J. Crawford, M. Ginsberg, E. Luks and A. Roy. Symmetry-Breaking Predicates for Search Problems Proc. KR, 148-159, 1996.
[2] A. M. Frisch and W. Harvey. Constraints for Breaking All Row and Column Symmetries in a Three-by-Two-Matrix. Proc. Symcon, 2003.
[3] A. M. Frisch, C. Jeerson, B. Martinez Hernandez and I. Miguel. The Rules of Constraint Modelling. Proc. IJCAI, 109-116, 2005.
[4] H. • Ohrman. Breaking Symmetries In Matrix Models. Masters Thesis, Dept. Information Technology, Uppsala University, 2005.
[5] J. F. Puget. Automatic detection of variable and value symmetries. CP 2005, 475-489, 2005

Acknowledgements Andrew Grayland is supported by an EPSRC CASE studentship, sponsored by Microsoft Research. Ian Miguel is supported by a Royal Academy of Engineering/EPSRC Research Fellowship. Colva Roney-Dougal is partly funded by the Nuffield Foundation.