Unique Symmetry breaking in CSPs using Group Theory

Iain McDonald*

University of St Andrews, Fife, Scotland,
iain@dcs.st-and.ac.uk

Abstract. In this paper I describe a new method of symmetry breaking using group theory which can represent very large sets of symmetries easily. I also define the class of unique symmetries which can be used to reduce the overhead of performing symmetry breaking during search. This makes it possible to break all symmetry on CSPs with a large number of symmetries.

1 Introduction

Many CSPs have an extremely large amount of symmetry. Take for example a competition scheduling problem, e.g. a sporting tournament. The input (the teams) are indistinguishable from each other in terms of the problem i.e. if we have a solution, we can exchange any two teams and we still have a solution. Thus for n teams we have n! symmetries. One approach to breaking symmetry in CSPs is presented in [1] and similarly SBDS[2]. These methods describe symmetries as functions which map assignments to assignments. This method of symmetry breaking is extremely limited due to the complexity of writing a large number of these functions and the limit of how many functions a compiler can compile in reasonable time.

The solution to this problem is a better method of implementing symmetry breaking. To do this we need to be able to describe large sets of symmetries efficiently and ensure we post the minimum number of symmetry breaking constraints needed to break all symmetry.

In this paper I present a new technique of efficiently storing symmetries for use in symmetry breaking during search. In doing so I define a new class of symmetries: unique symmetries, and I also propose a new method of symmetry breaking with reduced overheads compared to previous methods.

2 Background

We define a CSP C, with a **group** G, of symmetries acting on C with a full assignment of values to variables A. We say that $g \in G$ iff $\forall A$ that satisfies C, $g(A)$ satisfies C and $\forall A$ that doesn't satisfy C, $g(A)$ does not satisfy C.

* I would like to thank my supervisor Ian Gent and also Steve Linton and Barbara Smith for their help and encouragement. I would also like to thank the EPSRC for funding my research and finally the reviewers for their helpful comments.

Previous work has referred to a **set** of symmetries where each symmetry is described individually (e.g. SBDS requires the user to write a C++ function for every symmetry. This function takes a partial assignment as parameters and returns a symmetrical partial assignment). In this paper we deal with groups of symmetries and by doing so we can use the research in the field of group theory[3]. Gent and Smith[2] and Backofen and Will[1] perform symmetry breaking by adding constraints after backtracking that forbid the symmetric equivalents of failed assignments. I will be using the same technique of adding symmetry breaking constraints after backtracking.

3 Using Group Theory

A group is a non-empty set G (of elements) with a multiplication operator $\times$ with the properties that

1. G is closed under $\times$, i.e. $\forall g, h \in G, g \times h \in G$
2. $\times$ is associative, i.e. $\forall g, h, k \in G, (g \times h) \times k = g \times (h \times k)$
3. G has an identity id, i.e. $\forall g \in G, g \times id = id \times g = g$
4. every element g of G has an inverse g^{-1} such that $g \times g^{-1} = g^{-1} \times g = id$

We can represent the symmetries of any CSP using a group and in turn any group can be represented by permutations. A permutation is a bi-jection from a set to itself. The set of permutations P is said to be a generating set of G if we can repeatedly apply the permutations in P to themselves to reproduce all the elements in G. This is very beneficial as we can represent very large groups using only a few group generators, e.g. we can describe the symmetric group S_N using only two permutations: $(1, 2, 3, ..., N)$ and $(1, 2)$. There are N! elements in S_N. We can apply a group element of S_N to a set of numbers in the range of 1 to N. A set like this is called a set of points. Elements of a group are applied to a set of points in the same way we apply symmetries to partial assignments.

For a CSP with n variables with domains of size d, we can translate the assignment $var_i = val_j$ to the point $(n \times i) + j$. Given a partial assignment where k variables have been assigned values, we can translate this into a set of k points where each point of the set is a representative of the partial assignment. We find this representative using the above rule. We can then apply any element of the group to this set of points to produce a symmetrically equivalent set of points. This new set can then be used to produce a symmetrically equivalent partial assignment, i.e. point x translates to $var_{x/n} = val_{x \bmod n}$.

It is now possible to use group theory to find all the symmetrically equivalent partial assignments of a given partial assignment. I have used this to post symmetry breaking constraints of the form $(var \neq val) \Rightarrow g(var \neq val)$ to the search tree after a backtrack from trying $var = val$ in the same manner as [1] and [2].

4 Unique Symmetries

After every backtrack we post symmetry breaking constraints. It would be perfectly valid to use the generators to recreate all the elements of the group and post a constraint for every symmetry. This would incur a very high overhead by flooding the constraint solver with too many constraints. At every node in the search tree some of the symmetrical constraints are the same as each other. In many cases there are more duplicate constraints than distinct ones. In order to reduce the overhead of posting duplicate constraints, we must ensure that at each node in the search tree we only consider the unique symmetries.

Definition 1. *Given a partial assignment A, the maximal set of symmetries $G' \subseteq G$, is unique with respect to A iff $\forall g \in G' (\nexists h \in G' | h(A) = g(A))$.*

We want to find the result of applying the set of unique symmetries to a given partial assignment. Thankfully we can find this by using a group theoretic orbit finding algorithm. The orbit of G on a partial assignment A indicates how G acts on A by applying all the elements in G that map A to another assignment. The output of the orbit finding algorithm is a set of set of points and this can be translated into a set of partial assignments that we can post constraints for. There is an orbit finding algorithm in [3] with $O(g \times u)$ where g is the number of generators and u is the number of unique symmetries.

Using an orbit finding algorithm to perform unique symmetry breaking, I have solved the alien tiles problem featured in [4]. I used Ilog Solver version 4.4 and I over-rid the search routines to add constraints upon failure. The problem had 16 variables and 1152 symmetries. Using no symmetry breaking techniques, Solver took 1304.99 seconds and 57664 fails to solve the CSP while using the symmetry breaking technique described in this paper, Solver took 125.01 seconds and 479 fails to solve the CSP. In [4] the alien tiles problem was solved using SBDS but with better performance. They solved the problem in 56.589 seconds and 499 fails.

5 Discussion and Future Work

This paper is the first to have unique symmetry breaking. I have run experiments which have coped with groups of order larger than one thousand. These experiments have resulted in a 10 fold improvement in runtime. Less time is needed by the programmer to describe the symmetries as the system only needs the group generators and not a function for every symmetry.

[2] and [1] discard broken symmetries which my method does not. Broken symmetries are those that forbid assignments that we can guarantee will never take place. Both of these techniques reduce the overhead of breaking all symmetry. The overhead of breaking symmetry while discarding broken symmetries is greatest at the root whereas the overhead of breaking symmetries using only unique symmetries is greatest halfway down the search tree. Since there are exponentially more nodes halfway down the search tree than there are at the root,

it is most likely using just unique symmetries has a greater overhead than discarding broken symmetries. However, using group theory methods it should be possible to perform symmetry breaking using unique symmetries and discarding broken symmetries. Thus we would retain the efficient storage of the symmetries using group theory as well as minimizing the overhead as much as possible. This will hopefully allow us to break all symmetry in CSPs with n! symmetries for large instances of n.

References

1. R Backofen and S Will. Excluding symmetries in constraint-based search. In *Principles and Practice of Constraint Programming*, pages 73–87, 1999.
2. I P Gent and B M Smith. Symmetry breaking in constraint programming. In W. Horn, editor, *Proceedings of ECAI-2000*, pages 599–603. IOS Press, 2000.
3. G Butler. *Fundamental Algorithms for Permutation Groups*. Springer-Verlag, 1991.
4. I P Gent, S A Linton, and B M Smith. Symmetry breaking in the alien tiles puzzle. Technical Report APES-22-2000, APES Research Group, October 2000.