Breaking Value Symmetry*

Toby Walsh
NICTA and UNSW
Sydney, Australia
toby.walsh@nicta.com.au

Abstract

Symmetry is an important factor in solving many constraint satisfaction problems. One common type of symmetry is when we have symmetric values. In a recent series of papers, we have studied methods to break value symmetries (Walsh 2006a; 2007). Our results identify computational limits on eliminating value symmetry. For instance, we prove that pruning all symmetric values is NP-hard in general. Nevertheless, experiments show that much value symmetry can be broken in practice. These results may be useful to researchers in planning, scheduling and other areas as value symmetry occurs in many different domains.

Introduction

Many search problems contain symmetries. Symmetry occurs naturally in many problems (e.g. if we have identical machines to schedule, identical jobs to process, or equivalently skilled personnel to roster). Symmetry can also be introduced when we model a problem (e.g. if we name the elements in a set, we introduce the possibility of permuting their order). Unfortunately, symmetries increases the size of the search space. If we do not eliminate symmetries, we will waste much time visiting symmetric solutions, as well as those parts of the search tree which are symmetric to already visited states.

One common type of symmetry is when values are symmetric. For example, if we are assigning colors (values) to nodes (variables) in a graph coloring problem, we can uniformly swap the names of the colors throughout a coloring. As a second example if we are assigning nurses (values) to shifts (variables) in a rostering problems, and two nurses have the same skills, we may be able to interchange them uniformly throughout the schedule. In a recent series of papers (Walsh 2006a; 2007), we have studied methods to eliminate such value symmetries. A clear picture of breaking value symmetry is emerging from these studies. These results may be useful to researchers in other areas of AI as value symmetry is a common feature of many domains.

*The author is funded by the Australian Government's Department of Broadband, Communications and the Digital Economy and the Australian Research Council.

An example

To illustrate the ideas, we consider a simple problem from musical composition. The all interval series problem (prob007 in CSPLib.org) asks for a permutation of the numbers 0 to $n-1$ so that neighbouring differences form a permutation of 1 to $n-1$. For $n=12$, the problem corresponds to arranging the half-notes of a scale so that all musical intervals (minor second to major seventh) are covered. This is a simple example of a graceful graph problem in which the graph is a path.

We can model this as a constraint satisfaction problem in n variables with $X_i = j$ iff the ith number in the series is j. One solution for $n=11$ is:

$$X_1, X_2, \ldots, X_{11} \quad = \quad 3, 7, 4, 6, 5, 0, 10, 1, 9, 2, 8 \tag{1}$$

The all interval series problem has a number of different symmetries. First, we can reverse any solution and generate a new (but symmetric) solution:

$$X_1, X_2, \ldots, X_{11} \quad = \quad 8, 2, 9, 1, 10, 0, 5, 6, 4, 7, 3 \tag{2}$$

Second, the all interval series problem has a value symmetry as we can invert values. If we subtract all values in (1) from 10, we generate a second (but symmetric) solution:

$$X_1, X_2, \ldots, X_{11} \quad = \quad 7, 3, 6, 4, 5, 10, 0, 9, 1, 8, 2 \tag{3}$$

Third, we can do both and generate a third (but symmetric) solution:

$$X_1, X_2, \ldots, X_{11} \quad = \quad 2, 8, 1, 9, 0, 10, 5, 4, 6, 3, 7 \tag{4}$$

To eliminate such symmetric solutions from the search space, we can post additional constraints which eliminate all but one solution in each symmetry class. To eliminate the reversal of a solution, we can simply post the constraint:

$$X_1 < X_{11} \tag{5}$$

This eliminates solution (2) as it is a reversal of (1).

To eliminate the value symmetry which subtracts all values from 10, we can post:

$$X_1 \leq 10 - X_1, \quad X_1 = 10 - X_1 \Rightarrow X_2 < 10 - X_2 \tag{6}$$

This is equivalent to:

$$X_1 \leq 5, \qquad X_1 = 5 \Rightarrow X_2 < 5$$

This eliminates solutions (2) and (3). Finally, eliminating the third symmetry where we both reverse the solution and subtract it from 10 is more difficult. We can, for instance, post:

$$X_1 \leq 10 - X_{11},$$
$$X_1 = 10 - X_{11} \Rightarrow X_2 \leq 10 - X_{10},$$
$$X_1 = 10 - X_{11} \,\&\, X_2 = 10 - X_{10} \Rightarrow X_3 \leq 10 - X_9,$$
$$\vdots \tag{7}$$

Note that of the four symmetric solutions given earlier, only (4) with $X_1 = 2$, $X_2 = 8$ and $X_{11} = 7$ satisfies all three sets of symmetry breaking constraints: (5), (6) and (7). The other three solutions are eliminated. This leaves the question of where symmetry breaking constraints like (6) and (7) come from in general. Our work helps answer this question.

Formal background

A constraint satisfaction problem consists of a set of variables, each with a domain of values, and a set of constraints specifying allowed combinations of values for given subsets of variables. Variables take one value from a given finite set. A solution is an assignment of values to variables satisfying the constraints. Symmetry occurs in many constraint satisfaction problems. A *value symmetry* is a permutation of the values that preserves solutions. More formally, a value symmetry is a bijective mapping, σ of the values such that if $X_1 = d_1, \ldots, X_n = d_n$ is a solution then $X_1 = \sigma(d_1), \ldots, X_n = \sigma(d_n)$ is also. For example, in the all interval series problem, the value symmetry σ maps the value i onto $n - 1 - i$. A *variable symmetry*, on the other hand, is a permutation of the variables that preserves solutions. More formally, a variable symmetry is a bijective mapping, θ of the indices of variables such that if $X_1 = d_1, \ldots, X_n = d_n$ is a solution then $X_{\theta(1)} = d_1, \ldots, X_{\theta(n)} = d_n$ is also. For example, in the all interval series problem, the variable symmetry for reversing a series maps the index i onto $n - i + 1$. Symmetries are problematic as they increase the size of the search space. For instance, m interchangeable values increases the size of the search space by a factor of $m!$.

Lex-Leader method

One simple and effective mechanism to deal with symmetry is to add constraints which eliminate symmetric solutions (Puget 1993). For variable symmetries, Crawford *et al.* give a simple method that eliminates all symmetric solutions (Crawford *et al.* 1996). The basic idea is very simple. We pick an ordering on the variables, and then post symmetry breaking constraints to ensure that the final solution is lexicographically less than any symmetric re-ordering of the variables. For example, with a reversal symmetry which maps X_i to X_{n-i+1}, we post the lexicographical ordering constraint:

$$[X_1, \ldots, X_n] \quad \leq_{\text{lex}} \quad [X_n, \ldots, X_1]$$

This selects the "lex leader" assignment. Note that if X_i are all-different, as they are in the all interval series problem, we can simplify this lex leader constraint to give as in (5):

$$X_1 < X_n$$

The lex-leader method lies at the heart of most static methods for breaking variable symmetry. In their survey of symmetry breaking in constraint programming in the recent Handbook of Constraint Programming, Gent, Petrie and Puget observe that:

> "*...lex-leader remains of the highest importance, because there a number of ways it is used to derive new symmetry breaking methods ... [however] the lex-leader method is defined only for variable symmetries ... a proper generalisation of lex-leader to deal with value symmetries would be valuable, even if restricted to some special cases ...*" page 345 (Gent, Petrie, & Puget 2006)

This is the research challenge we have been tackling in our recent work (Walsh 2006a; 2007).

Value symmetry

In (Walsh 2006a), we propose a generalization of the lex-leader method that works with any type of value symmetry. Suppose we have a set Σ of value symmetries. We can eliminate all symmetric solutions with the constraint $\text{VALSYM}(\Sigma, [X_1, \ldots, X_n])$ which ensures for all $\sigma \in \Sigma$:

$$[X_1, \ldots, X_n] \quad \leq_{\text{lex}} \quad [\sigma(X_1), \ldots, \sigma(X_n)] \tag{8}$$

Where X_1 to X_n is any fixed ordering on the variables and $\sigma(X_i)$ represents the action of the symmetry σ on the value assigned to X_i. For example, if σ inverts values by mapping i onto $n - 1 - i$ as in the all interval series, we can simplify (8) to give:

$$[X_1, X_2, \ldots] \quad \leq_{\text{lex}} \quad [n-1-X_1, n-1-X_2, \ldots]$$

Expanding the definition of lex, and exploiting the fact that $X_1 \neq X_2$, we get as in (6):

$$X_1 \leq n-1-X_1, \quad X_1 = n-1-X_1 \Rightarrow X_2 < n-1-X_2$$

We give a linear time propagator for lex-leader constraints like (8), and show how to extend the lex-leader method to symmetries which act on both variables and values simultaneously.

In theory, this generalization of the lex-leader method solves the problem of value symmetries as it eliminates all symmetric solutions. Unfortunately, several problems remain. First, the set of symmetries Σ can be exponentially large (for example, there are $m!$ symmetries if we have m interchangeable values) requiring us to post a large number of lex ordering constraints. Second, decomposing VALSYM into separate lex ordering constraints hinders propagation and may prevent all symmetric values from being pruned. In (Walsh 2006a), we give a simple example where this decomposition hinders propagation. Somewhat surprisingly, there is little hope to overcome this second problem. We prove in (Walsh 2007) that pruning all symmetric values is intractable in general even with a small number of symmetries (assuming $P \neq NP$).

Theorem 1 *Pruning all inconsistent values for* $\text{VALSYM}(\Sigma, [X_1, ., X_n])$ *is NP-hard, even when* $|\Sigma|$ *is linearly bounded.*

Proof: In (Walsh 2007), we reduce 3-SAT to deciding if a particular VALSYM constraint has a solution. □

With value symmetry, whilst pruning all symmetric values is NP-hard, method likes those in (Roney-Dougal *et al.* 2004; Puget 2005) will eliminate all symmetric solutions in polynomial time. Despite the negative result given in Theorem 1, value symmetry appears easier to break than variable symmetry both in theory and in practice. For example, with variable symmetry, we prove in (Bessiere *et al.* 2004) that just eliminating all symmetric solutions is itself NP-hard.

Interchangeable values

So far, we have considered symmetries in general and ignored any special properties of the symmetry group. A common type of value symmetry with special properties that we can exploit is that due to interchangeable values. We can break all such value symmetry using value *precedence* (Law & Lee 2004), an idea which has been used in many contexts including the least number heuristic (Zhang 1996). To ensure value precedence, we can use the global constraint $\text{PRECEDENCE}([X_1, ., X_n])$. This holds iff the first time we use j is before the first time we use k for all $j < k$. For example, consider an assignment like:

$$X_1, X_2, X_3, \ldots, X_n \quad = \quad 1, 1, 2, 1, 3, \ldots, 2 \qquad (9)$$

This satisfies value precedence as 1 first occurs before 2, 2 first occurs before 3, etc. Now consider the symmetric assignment in which we swap 2 with 3:

$$X_1, X_2, X_3, \ldots, X_n \quad = \quad 1, 1, 3, 1, 2, \ldots, 3 \qquad (10)$$

This does not satisfy value precedence as 3 first occurs before 2. Posting a PRECEDENCE constraint eliminates all symmetric solutions due to interchangeable values. In (Walsh 2006b), we give a linear time propagator for the PRECEDENCE constraint. In (Walsh 2007), we argue that PRECEDENCE is in fact equivalent to VALSYM.

Another way to ensure value precedence is to channel into dual variables, Z_j which record the first index using each value j (Puget 2005). This maps value symmetry into variable symmetry on the Z_j. We can then break this variable symmetry by posting simple ordering constraints:

$$Z_1 < Z_2 < Z_3 < \ldots < Z_m \qquad (11)$$

This ensures that the first occurrence of 1 is before that of 2, that of 2 is before 3, etc. For example, the assignment in (9) satisfies (11) as $Z_1 = 1$, $Z_2 = 3$ and $Z_3 = 5$. However, the symmetric assignment in (10) does not satisfy (11) as $Z_2 = 5$ but $Z_3 = 3$. Puget proves that we can, in fact, break *any* value symmetry with a linear number of ordering constraints on the Z_j. Unfortunately, even with just two value symmetries, Puget's method hinders propagation (Walsh 2007). This is supported by the experiments in (Walsh 2007) where we see faster and more effective symmetry breaking with the global PRECEDENCE constraint. This is therefore a promising method to eliminate the symmetry due to interchangeable values.

Dynamic methods

An alternative to static methods which add constraints to eliminate symmetric solutions are dynamic methods which modify the search procedure to ignore symmetric branches. For example, with value symmetries, the GE-tree method dynamically eliminates all symmetric solutions from a backtracking search procedure in $O(n^4 \log(n))$ time (Roney-Dougal *et al.* 2004). Dynamic methods have the advantage that they do not conflict with the branching heuristic. However, dynamic methods do not prune the search space as much. Suppose we are at a particular node in the search tree explored by the GE-tree method. The GE-tree method essential performs only forward checking, pruning symmetric assignments from the domain of the *next* branching variable. Unlike static methods, the GE-tree method does not prune *deeper* variables. By comparison, static symmetry breaking constraints can prune deeper variables, resulting in interactions between the problem constraints and additional domain prunings. For this reason, static symmetry breaking methods can solve certain problems exponentially quicker than dynamic methods.

Theorem 2 *There exists a model of the pigeonhole problem in n variables and $n+1$ interchangeable values such that, given any variable and value ordering, the GE-tree method explores $O(2^n)$ branches, but which static symmetry breaking methods like value precedence solve in just $O(n^2)$ time.*

Proof: See (Walsh 2007). □

This theoretical result supports the experimental results in (Puget 2005) showing that static methods for breaking value symmetry outperform dynamic methods. Nevertheless, dynamic methods have been found useful as they may not conflict with the branching heuristic (Gargani & Refalo 2007; Van Hentenryck & Michel 2008). An interesting direction for future work are hybrid methods that combine the best features of static and dynamic symmetry breaking.

Related work

Puget proved that symmetric solutions can be eliminated by the addition of suitable constraints (Puget 1993). Crawford *et al.* presented the first general method for constructing variable symmetry breaking constraints (Crawford *et al.* 1996). To deal with large number of symmetries, Aloul *et al.* suggest breaking only those symmetries which are generators (Aloul *et al.* 2002). Puget and Walsh independently proposed propagators for value symmetry breaking constraints (Puget 2006; Walsh 2006a). To deal with the exponential number of such value symmetry breaking constraints, Puget proposed a global propagator which does forward checking in polynomial time (Puget 2006).

To eliminate symmetric solutions due to interchangeable values, Law and Lee defined value precedence for finite domain and set variables and proposed a specialized propagator for a pair of interchangeable values (Law & Lee 2004). Walsh extended this to a propagator for any number of interchangeable values (Walsh 2006b). Value precedence enforces the so-called "least number heuristic" (Zhang 1996). Finally, an alternative way to break value symmetry statically is to convert it into a variable symmetry by channelling into a dual Boolean viewpoint in which $B_{ij} = 1$ iff $X_i = j$, and using lexicographical ordering constraints on the Boolean variables (Flener *et al.* 2002; Law & Lee 2006). More recently, static symmetry breaking constraints have been proposed to eliminate the symmetry due to both interchangeable variables and values (Flener *et al.* 2006; Law *et al.* 2007).

A number of dynamic methods have been proposed to deal with value symmetry. Van Hentenryck *et al.* gave a labelling schema for eliminating all symmetric solutions due to interchangeable values (Van Hentenryck *et al.* 2003). Inspired by this method, Roney-Dougal *et al.* gave a polynomial method to construct a GE-tree, a search tree without value symmetry (Roney-Dougal *et al.* 2004). Finally, Sellmann and van Hentenryck gave a $O(nd^{3.5} + n^2d^2)$ dominance detection algorithm for eliminating all symmetric solutions when both variables and values are interchangeable (Sellmann & Van Hentenryck 2005).

Conclusions

Value symmetries can be broken either statically (by adding constraints to prune symmetric solutions) or dynamically (by modifying the search procedure to avoid symmetric branches). We have shown that both approaches have computational limitations. With static methods, we can eliminate all symmetric solutions in polynomial time but pruning all symmetric values is NP-hard in general (or equivalently, we can avoid visiting symmetric leaves of the search tree in polynomial time but avoiding symmetric subtrees is NP-hard). With dynamic methods, we can take exponential time on problems which static methods solve without search. Nevertheless, experimental results in (Puget 2006; Walsh 2006b; 2006a; 2007) and elsewhere show that value symmetry can be dealt with effectively in practice.

These results may be useful to researchers in closely related areas like planning and scheduling where value symmetries arise and need to be eliminated. They may also be useful in areas like model counting, verification, and automated reasoning as value symmetry arises in many of these problems too. There are many open questions raised by this research. For example, are there symmetries where all symmetric values can be pruned tractably? How do we combine the best features of static and dynamic symmetry breaking?

References

Aloul, F.; Ramani, A.; Markov, I.; and Sakallah, K. 2002. Solving difficult SAT instances in the presence of symmetries. In *Proc. of the Design Automation Conf.*, 731–736.

Bessiere, C.; Hebrard, E.; Hnich, B.; and Walsh, T. 2004. The complexity of global constraints. In *Proc. of the 19th National Conf. on AI.* AAAI.

Crawford, J.; Luks, G.; Ginsberg, M.; and Roy, A. 1996. Symmetry breaking predicates for search problems. In *Proc. of the 5th Int. Conf. on Knowledge Representation and Reasoning, (KR '96)*, 148–159.

Flener, P.; Frisch, A.; Hnich, B.; Kiziltan, Z.; Miguel, I.; Pearson, J.; and Walsh, T. 2002. Breaking row and column symmetry in matrix models. In *8th Int. Conf. on Principles and Practices of Constraint Programming (CP-2002)*.

Flener, P.; Pearson, J.; Sellmann, M.; and Van Hentenryck, P. 2006. Static and dynamic structural symmetry breaking. In *Proc. of 12th Int. Conf. on Principles and Practice of Constraint Programming (CP2006)*.

Gargani, A., and Refalo, P. 2007. An efficient model and strategy for the steel mill slab design problem. In *Proc. of 13th Int. Conf. on Principles and Practice of Constraint Programming (CP2007)*.

Gent, I.; Petrie, K.; and Puget, J.-F. 2006. Symmetry in constraint programming. In *Handbook for Constraint Programming*. Elsevier.

Law, Y., and Lee, J. 2004. Global constraints for integer and set value precedence. In *Proc. of 10th Int. Conf. on Principles and Practice of Constraint Programming (CP2004)*, 362–376.

Law, Y., and Lee, J. 2006. Symmetry Breaking Constraints for Value Symmetries in Constraint Satisfaction. *Constraints* 11(2–3):221–267.

Law, Y.-C.; Lee, J.; Walsh, T.; and Yip, J. 2007. Breaking symmetry of interchangeable variables and values. In *13th Int. Conf. on Principles and Practices of Constraint Programming (CP-2007)*.

Puget, J.-F. 1993. On the satisfiability of symmetrical constrained satisfaction problems. In *Proc. of ISMIS'93*, LNAI 689, 350–361.

Puget, J.-F. 2005. Breaking all value symmetries in surjection problems. In *Proc. of 11th Int. Conf. on Principles and Practice of Constraint Programming (CP2005)*.

Puget, J.-F. 2006. An efficient way of breaking value symmetries. In *Proc. of the 21st National Conf. on AI.* AAAI.

Roney-Dougal, C.; Gent, I.; Kelsey, T.; and Linton, S. 2004. Tractable symmetry breaking using restricted search trees. In *Proc. of ECAI-2004*. IOS Press.

Sellmann, M., and Van Hentenryck, P. 2005. Structural symmetry breaking. In *Proc. of 19th IJCAI*.

Van Hentenryck, P., and Michel, L. 2008. The steel mill slab design problem revisited. In *Proc. of 5th Int. Workshop on Integration of AI and OR Techniques in Constraint Programming (CP-AI-OR)*.

Van Hentenryck, P.; Agren, M.; Flener, P.; and Pearson, J. 2003. Tractable symmetry breaking for CSPs with interchangeable values. In *Proc. of the 18th IJCAI*.

Walsh, T. 2006a. General symmetry breaking constraints. In *12th Int. Conf. on Principles and Practices of Constraint Programming (CP-2006)*.

Walsh, T. 2006b. Symmetry breaking using value precedence. In *Proc. of the 17th ECAI*. European Conf. on Artificial Intelligence.

Walsh, T. 2007. Breaking value symmetry. In *13th Int. Conf. on Principles and Practices of Constraint Programming (CP-2007)*.

Zhang, J. 1996. Constructing finite algebras with FALCON. *Journal of Automated Reasoning* 17(1):1–22.