Breaking Value Symmetry[*]

Toby Walsh
NICTA and UNSW
Sydney, Australia
tw@cse.unsw.edu.au

Abstract

Symmetry is an important factor in solving many constraint satisfaction problems. One common type of symmetry is when we have symmetric values. We can eliminate such value symmetry either statically by adding constraints to prune symmetric solutions or dynamically by modifying the search procedure to avoid symmetric branches. We show that either method has computational limitations. With static methods, pruning all symmetric values is NP-hard in general. With dynamic methods, we can take exponential time on problems which static methods solve without search. Finally, we consider a common type of value symmetry, that due to interchangeable values, where polynomial methods have been proposed to break symmetries. We show that despite these theoretical limitations, the methods proposed to break the symmetries introduced by interchangeable values are both effective in theory and in practice.

1 Introduction

Many search problems contain symmetries. Symmetry occurs naturally in many problems (e.g. if we have identical machines to schedule, or identical jobs to process). 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. We must therefore try to eliminate symmetry or 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. For a problem with value symmetries, all symmetric solutions can be eliminated in polynomial time [Roney-Dougal *et al.*, 2004; Puget, 2005]. However, as we show here, pruning *all* symmetric values is NP-hard in general. Nevertheless, methods that have been proposed, like those in [Law and Lee, 2004; Puget, 2005; Aloul, 2006], appear to be effective at dealing with common types of value symmetry.

2 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, suppose we wish to assign colors (values) to nodes (variables) in a graph coloring problem. This problem has a value symmetry that permits us to interchange any two colors uniformly throughout a coloring. 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, suppose we wish to assign times (values) to exams (variables) in an exam scheduling problem and we have two exams taken by the same set of students. This variable symmetry permits us to interchange the two exams.

Symmetries are problematic as they increase the size of the search space. For instance, if we have m interchangeable values, symmetry increases the size of the search space by a factor of $m!$. The set of symmetries of a constraint satisfaction problem form a group under composition. In this work, we place no restrictions on the type of group. In particular, we are not restricted to products of the symmetry group, S_n.

[*]NICTA is funded by the Australian Government's Department of Communications, Information Technology and the Arts and the Australian Research Council DCITA and ARC through Backing Australia's Ability and the ICT Centre of Excellence program. Thanks to Chris Jefferson and Jean-Francois Puget for useful comments. A shorter version of this paper without experimental results appears in the Proceedings of the *13th International Conference on Principles and Practices of Constraint Programming (CP-2007).*

However, we do assume that the symmetries of the constraint satisfaction problem are known in advance. For instance, if we are coloring a graph and use a straight forward model with variables for nodes and values for colors, we know that the values are fully interchangeable. A number of methods have been developed to find symmetries in a constraint satisfaction problem automatically.

Many constraint solvers explore the space of partial assignments enforcing some local consistency. We consider three local consistencies. Given a constraint C, a *support* is assignment to each variable of a value in its domain which satisfies C. A constraint is *generalized arc consistent* (*GAC*) iff for each variable, every value in its domain belongs to a support. A set of constraints is GAC iff each constraint is GAC. On binary constraints, GAC is simply called arc consistency (AC). A set of binary constraints is *singleton arc consistent* (*SAC*) iff we can assign every variable with each value in its domain and make the resulting problem arc consistent (AC). Finally, a set of binary constraint is *k-consistent* iff each $k-1$ assignment can be consistently extended to a kth variable, and is *strongly k-consistent* iff it is j-consistency for all $j \leq k$. We will compare local consistency properties applied to sets of constraints, c_1 and c_2 which are logically equivalent. As in [Debruyne and Bessière, 1997], a local consistency property Φ on c_1 is as strong as Ψ on c_2 iff, given any domains, if Φ holds on c_1 then Ψ holds on c_2; Φ on c_1 is stronger than Ψ on c_2 iff Φ on c_1 is as strong as Ψ on c_2 but not vice versa; Φ on c_1 is equivalent to Ψ on c_2 iff Φ on c_1 is as strong as Ψ on c_2 and vice versa.

3 Static methods

One simple and common mechanism to deal with symmetry is to add constraints which eliminate symmetric solutions [Puget, 1993]. Suppose we have a set Σ of value symmetries. Based on [Crawford *et al.*, 1996], we can eliminate all symmetric solutions by posting a global constraint which ensures that the solution is ordered lexicographically before any of its symmetries. More precisely, we post the global constraint $\textsc{ValSymBreak}(\Sigma, [X_1, \ldots, X_n])$ which ensures $[X_1, \ldots, X_n] \leq_{\text{lex}} [\sigma(X_1), \ldots, \sigma(X_n)]$ for all $\sigma \in \Sigma$ where X_1 to X_n is a fixed ordering on the variables and $\sigma(X_i)$ represents the action of the symmetry σ on the value assigned to X_i. Unfortunately, pruning *all* values from such a symmetry breaking constraint is NP-hard.

Theorem 1 *Deciding if* $\textsc{ValSymBreak}(\Sigma, [X_1, \ldots, X_n])$ *is GAC is NP-complete, even when* $|\Sigma|$ *is linearly bounded.*

Proof: Membership in NP follows by giving a support for every possible assignment. To prove it is NP-hard, we give a reduction from a 3-SAT problem in N Boolean variables and M clauses. We construct a CSP with $N + M + 1$ variables over $4N + 2$ possible values. The first $4N$ values partition into $2N$ interchangeable pairs. The values $4i - 3$ and $4i - 2$ are interchangeable, as are $4i - 1$ and $4i$ for $1 \leq i \leq N$. The values $4i - 3$ and $4i - 2$ represent the SAT variable x_i being true, whilst the values $4i-1$ and $4i$ represent the SAT variable x_i being false. The final two values, $4N + 1$ and $4N + 2$ are not interchangeable. The first N CSP variables represent a "truth assignment". We have $X_i \in \{4i-3, 4i-2, 4i-1, 4i\}$ for $1 \leq i \leq N$. The next M CSP variables ensure at least one literal in each clause is true. For example, if the ith clause is $x_j \vee \neg x_k \vee x_l$, then the domain of X_{N+i} is $\{4j - 3, 4j - 2, 4k - 1, 4k, 4l - 3, 4l - 2\}$. The final variable X_{N+M+1} is a "switch" and has the domain $\{4N + 1, 4N + 2\}$. Note that if a value is in the domain of a variable then so is every symmetry of this value.

We have two sets of constraints. First, we have the binary constraints $odd(X_{N+M+1}) \rightarrow odd(X_i)$ for $1 \leq i \leq N$ and $odd(X_{N+M+1}) \rightarrow even(X_{N+j})$ for $1 \leq j \leq M$. Second, we have the constraints $odd(X_{N+M+1}) \rightarrow PHP(N, N+1)$ and $even(X_{N+M+1}) \rightarrow PHP(N, N)$ where $PHP(i, j)$ is a pigeonhole constraint which holds iff the variables X_1 to X_i take j distinct values. Note that $PHP(N, N + 1)$ is unsatisfiable and that $PHP(N, N)$ is satisfiable. Thus, the constructed CSP is unsatisfiable if $X_{N+M+1} = 4N + 1$ and satisfiable if $X_{N+M+1} = 4N + 2$. Note that if we take any solution of the CSP and permute any of the interchangeable values, we still have a solution. Thus, if Σ is the set of symmetries induced by these interchangeable values, it is permissible to add $\textsc{ValSymBreak}(\Sigma, [X_1, \ldots, X_n])$ to this constraint satisfaction problem to eliminate value symmetry.

Suppose our branching heuristic sets the switch variable X_{N+M+1} to $4N + 1$. Enforcing AC on the binary constraints prunes the domains of X_i to $\{4i - 3, 4i - 1\}$ for $1 \leq i \leq N$. Similarly, the domain of X_{N+i} is reduced to $\{4j - 2, 4k, 4l - 2\}$. Consider now finding a support for $\textsc{ValSymBreak}$ given this particular subproblem. Now, $\textsc{ValSymBreak}(\Sigma, [X_1, \ldots, X_n])$ ensures that the interchangeable values first appear in order. In particular, X_{N+i} can only take the value $4j - 2$ if X_j had previously been assigned $4j - 3$. In other words, X_{N+i} can only take the value $4j - 2$ if x_j is set to true in the "truth assignment". Similarly, X_{N+i} can only take the value $4k$ if X_k had previously been assigned $4k - 1$. In other words, X_{N+i} can only take the value $4k$ if x_k is set to false in the "truth assignment". Finally, X_{N+i} can only take the value $4l - 2$ if X_j had previously been assigned $4l - 3$. In other words, X_{N+i} can only take the value $4l - 2$ if x_l is set to true in the "truth assignment". Thus, at least one of the literals in the ith clause must have been set to true in the "truth assignment". Hence, there is a support for $\textsc{ValSymBreak}$ iff the original 3-SAT problem is satisfiable. By Theorem 3, $|\Sigma|$ can be linearly bound. $\square$

This is a somewhat surprising result. Whilst it is polynomial to eliminate all symmetric solutions either statically [Puget, 2005] or dynamically [Roney-Dougal *et al.*, 2004], it is NP-hard to lookahead and prune all symmetric values. Equivalently, whilst we can avoid visiting symmetric leaves of the search tree in polynomial time, avoiding symmetric subtrees is NP-hard.

4 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]. However, as we show now, such dynamic methods may not prune *all* the symmetric values which static methods can do. Suppose we are at a particular node in the search tree explored by the GE-tree method. Consider the current and all past variables seen so far. The GE-tree method can be seen as performing forward checking on a static symmetry breaking constraint over this set of variables. This prunes 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 with 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 can solve in just $O(n^2)$ time.*

Proof: The $n+1$ constraints in the CSP are $\bigvee_{i=1}^{n} X_i = j$ for $1 \leq j \leq n+1$, and the domains are $X_i \in \{1, \dots, n+1\}$ for $1 \leq i \leq n$. The problem is unsatisfiable by a simple pigeonhole argument. Any of the static methods for breaking value symmetry presented later in this paper will prune $n+1$ from every domain in $O(n^2)$ time. Enforcing GAC on the constraint $\bigvee_{i=1}^{n} X_i = n+1$ then proves unsatisfiability. On the other hand, the GE-tree method irrespective of the variable and value ordering, will only terminate each branch when $n-1$ variables have been assigned (and the last variable is forced). A simple calculation shows that the size of the GE-tree more than doubles as we increase n by 1. Hence we will visit $O(2^n)$ branches before declaring the problem is unsatisfiable. □

This theoretical result supports the experimental results in [Puget, 2005] showing that static methods for breaking value symmetry can outperform dynamic methods. Given the intractability of pruning all symmetric values in general, we focus in the rest of the paper on a common and useful type of value symmetry where polynomial symmetry breaking methods have been proposed: we will suppose that values are ordered into partitions, and values within each partition are uniformly interchangeable.

5 Generator symmetries

One way to propagate ValSymBreak is to decompose it into individual lexicographical ordering constraints, $[X_1, \dots, X_n] \leq_{\text{lex}} [\sigma(X_1), \dots, \sigma(X_n)]$ and use one of the propagators proposed in [Puget, 2006] or [Walsh, 2006a]. Even if we ignore the fact that such a decomposition may hinder propagation (see Theorem 2 in [Walsh, 2006a]), we have to cope with Σ, the set of symmetries being exponentially large in general. For instance, if we have m interchangeable values, then Σ contains $m!$ symmetries. To deal with large number of symmetries, Aloul *et al.* suggest breaking only those symmetries corresponding to generators of the group [Aloul *et al.*, 2002]. Consider the generators which interchange adjacent values within each partition. If the m values partition into k classes of interchangeable values, there are just $m-k$ such generators. We prove here that breaking *just* these generator symmetries eliminates *all* symmetric solutions. Thus, we have identified a special class of symmetries where using just generators is complete.

Theorem 3 *If Σ is a set of symmetries induced by interchangeable values, and Σ_g is the set of generators corresponding to interchanging adjacent values then posting* $\text{ValSymBreak}(\Sigma_g, [X_1, \dots, X_n])$ *eliminates all symmetric solutions.*

Proof: Assume $\text{ValSymBreak}(\Sigma_g, [X_1, \dots, X_n])$. Consider any two interchangeable values, j and k where $j < k$, Let $\sigma_j \in \Sigma_g$ be the symmetry which swaps just j with $j+1$. To ensure $[X_1, \dots, X_n] \leq_{\text{lex}} [\sigma_j(X_1), \dots, \sigma_j(X_n)]$, j must occur before $j+1$ in X_1 to X_n. By transitivity, j therefore occurs before k. Thus, for the symmetry σ' which swaps just j with k, $[X_1, \dots, X_n] \leq_{\text{lex}} [\sigma'(X_1), \dots, \sigma'(X_n)]$. Consider now any symmetry $\sigma \in \Sigma$. The proof proceeds by contradiction. Suppose $[X_1, \dots, X_n] >_{\text{lex}} [\sigma(X_1), \dots, \sigma(X_n)]$. Then there exists some j with $X_j > \sigma(X_j)$ and $X_i = \sigma(X_i)$ for all $i < j$. Consider the symmetry σ' which swaps just X_j with $\sigma(X_j)$. As argued before, $[X_1, \dots, X_n] \leq_{\text{lex}} [\sigma'(X_1), \dots, \sigma'(X_n)]$. But this contradicts $[X_1, \dots, X_n] >_{\text{lex}} [\sigma(X_1), \dots, \sigma(X_n)]$ as σ and σ' act identically on the first j variables in X_1 to X_n. Hence, $[X_1, \dots, X_n] \leq_{\text{lex}} [\sigma(X_1), \dots, \sigma(X_n)]$. □

Not surprisingly, reducing the number of symmetry breaking constraints to linear is not without consequence. Whilst restricting symmetry breaking to just these generators eliminates all symmetric solutions, we may not necessarily prune all symmetric values. Equivalently, whilst restricting symmetry breaking to just these generators eliminates all symmetric leaves of the search tree, we may not necessarily avoid all symmetric subtrees.

Theorem 4 *If Σ is a set of symmetries induced by interchangeable values, and Σ_g is the set of generators corresponding to interchanging adjacent values then GAC on* $\text{ValSymBreak}(\Sigma, [X_1, \dots, X_n])$ *is stronger than GAC on* $[X_1, \dots, X_n] \leq_{\text{lex}} [\sigma(X_1), \dots, \sigma(X_n)]$ *for all* $\sigma \in \Sigma_g$.

Proof: Clearly it is at least as strong. To show it is stronger, suppose all values are interchangeable with each other. Consider $X_1 = 1$, $X_2 \in \{1, 2\}$, $X_3 \in \{1, 3\}$, $X_4 \in \{1, 4\}$ and $X_5 = 5$. Then enforcing GAC on $\text{ValSymBreak}(\Sigma, [X_1, \dots, X_5])$ prunes 1 from X_2, X_3 and X_4. However, $[X_1, \dots, X_5] \leq_{\text{lex}} [\sigma(X_1), \dots, \sigma(X_5)]$ is GAC for all $\sigma \in \Sigma_g$. . □

It is not hard to see that there are other sets of generators for the symmetries induced by interchangeable values which do not necessarily eliminate all symmetric solutions (e.g. with the generators which interchange the value 1 with any i, we do not eliminate either the assignment $X_1 = 1$, $X_2 = 2$ or the symmetric assignment $X_1 = 1$, $X_2 = 3$). Thus we have shown that the choice of generators is important.

6 Puget's decomposition

With value symmetries, a second method that eliminates all symmetric solutions is a decomposition due to [Puget, 2005]. Consider a surjection problem (where each value is used at least once) with interchangeable values. We can channel into dual variables, Z_j which record the first index using the value j by posting the binary constraints: $X_i = j \rightarrow Z_j \leq i$ and $Z_j = i \rightarrow X_i = j$ for all $1 \leq i \leq n$, $1 \leq j \leq m$. We can then eliminate all symmetric solutions by insisting that interchangeable values *first* occur in some given order. That is, we place strict ordering constraints on the Z_k within each class of interchangeable values. Puget notes that any problem can be made into a surjection by introducing m additional new variables, X_{n+1} to X_{n+m} where $X_{n+i} = i$. These variables ensure that each value is used at least once. In fact, we don't need additional variables. It is enough to ensure that each Z_j has a dummy value, which means that j is not assigned, and to order (dummy) values appropriately. Unfortunately, Puget's decomposition into binary constraints hinders propagation.

Theorem 5 *If Σ is a set of symmetries induced by interchangeable values, then GAC on* $\textsc{ValSymBreak}(\Sigma, [X_1, \ldots, X_n])$ *is stronger than AC on Puget's decomposition into binary constraints.*

Proof: It is clearly at least as strong. To show it is stronger, suppose all values are interchangeable with each other. Consider $X_1 = 1$, $X_2 \in \{1, 2\}$, $X_3 \in \{1, 3\}$, $X_4 \in \{3, 4\}$, $X_5 = 2$, $X_6 = 3$, $X_7 = 4$, $Z_1 = 1$, $Z_2 \in \{2, 5\}$, $Z_3 \in \{3, 4, 6\}$, and $Z_4 \in \{4, 7\}$. Then all Puget's symmetry breaking constraints are AC. However, enforcing GAC on $\textsc{ValSymBreak}(\Sigma, [X_1, \ldots, X_5])$ will prune 1 from X_2. □

If *all* values are interchangeable with each other, we only need to enforce a slightly stronger level of local consistency to prune all symmetric values. More precisely, enforcing singleton arc consistency on Puget's binary decomposition will prune all symmetric values.

Theorem 6 *If all values are interchangeable and Σ is the set of symmetries induced by this then GAC on* $\textsc{ValSymBreak}(\Sigma, [X_1, \ldots, X_n])$ *is equivalent to SAC on Puget's decomposition into binary constraints.*

Proof: Suppose Puget's encoding is AC. We will show that there is at least one support for $\textsc{ValSymBreak}$. We assign Z_1 to Z_m in turn, giving each the smallest remaining value in their domain, and enforcing AC on the encoding after each assignment. This will construct a support without the need for backtracking. At each choice point, we ensure that a new value is used as soon as possible, thus giving us the most freedom to use values in the future. Suppose now that Puget's encoding is SAC. Then, by the definition of SAC, we can assign any variable with any value in its domains and be sure that the problem can be made AC without a domain wipeout. But if the problem can be made AC, it has support. Thus every value in every domain has support. Hence enforcing SAC on Puget's decomposition ensures that $\textsc{ValSymBreak}$ is GAC. □

We might wonder if singleton arc-consistency is enough for arbitrary value symmetries. That is, does enforcing SAC on Puget's encoding prune all symmetric values? We can prove that no fixed level of local consistency is sufficient. Given the intractability of pruning all symmetric values in general, this result is not surprising.

Theorem 7 *For any given k, there exists a value symmetry and domains for which Puget's encoding is strongly k-consistent but is not $k+1$-consistent.*

Proof: We construct a CSP problem with $2k+1$ variables over $2(k+1)$ possible values. The $2(k+1)$ values partition into $k+1$ pairs which are interchangeable. More precisely, the values i and $k+1+i$ are interchangeable for $1 \leq i \leq k+1$. The first k variables of the CSP have $k+1$ values between them (hence, one value is not taken). More precisely, $X_i \in \{i, i+1\}$ for $1 \leq i \leq k$. The remaining $k+1$ variables then take the other $k+1$ values. More precisely, $X_{k+i} = k+1+i$ for $1 \leq i \leq k+1$. The values 1 to $k+1$ need to be used by the first k variables, X_1 to X_k so that the last $k+1$ variables, X_{k+1} to $X_{2(k+1)}$ can use the values $k+2$ to $2(k+1)$. But this is impossible by a pigeonhole argument. Puget's encoding of this is strongly k-consistent. since any assignment of $k-1$ or less variables can be extended to an additional variable. On the other hand, enforcing $k+1$-consistency will discover that the CSP has no solution. □

Finally, we compare this method with the previous method based on breaking the symmetries corresponding to the generators which interchange adjacent values.

Theorem 8 *If Σ is a set of symmetries induced by interchangeable values, and Σ_g is the set of generators interchanging adjacent values then AC on Puget's decomposition for Σ is stronger than GAC on $[X_1, \ldots, X_n] \leq_{\text{lex}} [\sigma(X_1), \ldots, \sigma(X_n)]$ for all $\sigma \in \Sigma_g$.*

Proof: Suppose Puget's decomposition is AC. Consider the symmetry σ which interchanges j with $j+1$. Consider any variable and any value in its domain. We show how to construct a support for this assignment. We assign every other variable with j if it is in its domain, otherwise any value other than $j+1$ and failing this, $j+1$. Suppose this is not a support for $[X_1, \ldots, X_n] \leq_{\text{lex}} [\sigma(X_1), \ldots, \sigma(X_n)]$. This means that in the sequence from X_1 to X_n, we had to use the value $j+1$ before the value j. However, as Puget's decomposition is AC, there is a value in the domain of Z_j smaller than Z_{j+1}. This contradicts $j+1$ having to be used before j. Hence, this must be a support. Thus $[X_1, \ldots, X_n] \leq_{\text{lex}} [\sigma(X_1), \ldots, \sigma(X_n)]$ is GAC for all $\sigma \in \Sigma_g$. To show that AC on Puget's decomposition is stronger consider again the example used in the proof of Theorem 4. The lexicographical ordering constraint for each generator $\sigma \in \Sigma_g$ is GAC without any domain pruning. However, enforcing AC on Puget's decomposition prunes 1 from X_2, X_3 and X_4. □

7 Value precedence

A third method to eliminate all symmetric solutions induced by interchangeable values uses the global *precedence* constraint [Law and Lee, 2004]. The constraint $\textsc{Precedence}([X_1, \ldots, X_n])$ holds iff $\min\{i \mid X_i = j \vee i = n+1\} < \min\{i \mid X_i = k \vee i = n+2\}$ for all $j < k$. That is, the first time we use j is before the first time we use k for all $j < k$. Posting such a precedence constraint eliminates all symmetric solutions due to interchangeable values. In [Walsh, 2006b], a GAC propagator for such a precedence constraint is proposed which takes $O(nm)$ time. It is not hard to show that $\text{PRECEDENCE}([X_1, \ldots, X_n])$ is equivalent to $\text{VALSYMBREAK}(\Sigma, [X_1, \ldots, X_n])$. Hence, enforcing GAC on such a precedence constraint prunes all symmetric values in polynomial time.

Precedence constraints can also be defined when values partition into more than one interchangeable class. We just insist that the values within each class first occur in a fixed order. In [Walsh, 2006b], a propagator for such a precedence constraint is proposed which takes $O(n \prod_i m_i)$ time where m_i is the size of the ith class of interchangeable values. Whilst this prunes all symmetric values, it is only polynomial if we can bound the number of classes of interchangeable values. This complexity is now not surprising. We have shown that pruning all symmetric values is NP-hard when the number of classes of interchangeable values is unbounded.

8 Breaking variable and value symmetry

Variable symmetries can also be broken statically by posting constraints. Following [Crawford *et al.*, 1996], we can eliminate all symmetric solutions with a global constraint which ensures that the solution is ordered lexicographically before any of the symmetries of the solution. More precisely, give a set of variable symmetries Θ, we can eliminate all symmetric solutions with a global constraint which ensures $[X_1, \ldots, X_n] \leq_{\text{lex}} [X_{\theta(1)}, \ldots, X_{\theta(n)}]$ for all $\theta \in \Theta$ where X_1 to X_n is a fixed ordering on the variables. Pruning *all* values from such a symmetry breaking constraint is NP-hard in general [Crawford *et al.*, 1996; Bessiere *et al.*, 2004]. Consider, for example, a model of the rehearsal problem (prob039 in CSPLib) where we have a variable for each time slot whose value is the piece to rehearse. This model has a variable symmetry as we can invert any rehearsal scheduling without violating any constraints. This is equivalent to swapping X_i with X_{n-i+1}. We eliminate this symmetry by posting the constraint: $[X_1, \ldots, X_n] \leq_{\text{lex}} [X_n, \ldots, X_1]$. Such variable symmetry breaking constraints are consistent with the value symmetry breaking constraints discussed here. We must, however, ensure that all are based on the same fixed variable ordering. Whilst we can consistently post both variable and value symmetry breaking constraints, this may not eliminate all symmetric solutions resulting from the interaction of variable and value symmetry (see, for example, Theorem 3 in [Walsh, 2006a]).

9 Experimental results

We now compare these value symmetry breaking methods experimentally. Puget has shown that his static symmetry breaking method significantly outperforms the dynamic GE-tree method. We therefore look at just the three static methods.

Generator symmetries: we post lexicographical ordering constraints for the generators of the symmetry group that interchange adjacent values and enforce GAC using a linear time propagator [Walsh, 2006a].

Puget's decomposition: we enforce AC using the solver's built-in propagators.

Value precedence: we post a single global value precedence constraint and enforce GAC using a linear time propagator [Walsh, 2006b].

Graph coloring

As in an earlier study of interchangeable values [Law and Lee, 2006], we experimented with graph coloring problems from the DIMACS benchmark suite. We use a straight forward model with one decision variable for each vertex. The values represent colors and are completely interchangeable. We coded the graph coloring problems in BProlog and ran them on a PowerPC 1GHz G4 processor with 1.25 GB RAM. Table 1 gives results. There is little to choose between the different symmetry breaking methods. However, the generator symmetry method is never fastest on any problem so can perhaps be excluded on these grounds. Although the differences between Puget's method and the global value precedence constraint are slight, breaking symmetry using a global value precedence constraint solves the most problems within the 2 hour time-out. It also often has the fewest branches or the shortest runtime.

We will make some additional observations about these results. On the fifth problem `DSJC125.1`, we actually find the optimal coloring faster without any symmetry breaking constraints. We conjecture that symmetry breaking in this case is conflicting with the branching heuristic. However, this problem is exceptional and symmetry breaking constraints are usually essential to be able to prove optimality within the 2 hour time-out. On the final problem `ash331GPIA`, we also find the optimal coloring and prove optimality quicker without symmetry breaking constraints. However, when we have an additional color available, the symmetry breaking constraints help keep the branching heuristic on track.

Note that as the number of colors (and thus the number of interchangeable values) increases, we see greater benefits using the global value precedence constraint compared to Puget's method or using generator symmetries. With the largest number of interchangeable values, the global value precedence constraint can be up to two orders of magnitude faster at reaching its fixed point than the other symmetry breaking methods. Finally, whilst the different symmetry breaking methods in theory prune the search tree differently, this is only seen on a few problem instances.

9.1 n by n queens

As in the first experiment on value symmetry breaking in [Puget, 2005], we used a simple model of the n by n queens problem. The aim is to color each square in a n by n chessboard with one of n colors so that no line (row, column or diagonal) has the same color twice. This is equivalent to finding n non-intersecting solutions to the n-queens problems. This is a difficult combinatorial problem. The existence of a solution for $n = 12$ was open until recently. We model this with n^2 variables, each with n possible values, and an all different constraint along each line. The model has 8 variable symmetries corresponding to the rotations and reflections of the chessboard. We break these by posting the ordering constraints: $X_1 < X_n$, $X_1 < X_{n^2-n+1}$, $X_1 < X_{n^2}$ and $X_2 < X_{n+1}$. The model also has $n!$ value symmetries as

`problem`	colors	value symmetry breaking							
vertices/	k	none		generator symmetries		Puget's method		value precedence	
edges		**b**	**t**	**b**	**t**	**b**	**t**	**b**	**t**
`myciel5`	4	6,264	0.31	**261**	0.08	**261**	**0.04**	**261**	0.06
47/236	5	43,001,880	1,334.71	**286,994**	53.66	**286,994**	**16.41**	**286,994**	21.45
	6^*	**1**	**0.01**	**1**	0.02	**1**	0.02	**1**	**0.01**
`GEOM50_5a`	7			5,504	0.89	**3,047**	0.83	**4,928**	**0.33**
50/238	8			**61,430**	7.81	**61,430**	10.24	65,398	**3.11**
	9^*	**1**	**0.01**	25	0.03	257	0.03	**1**	**0.01**
`R50_5gb8`	8			**3,047**	1.57	**3,047**	0.83	3,061	**0.41**
50/612	9			92,302	48.59	**92,202**	19.98	95,685	**12.11**
	10^*	**2**	**0.01**	1,901	0.89	1,897	0.43	2,139	0.26
`queen8_8`	7	5,040	0.31	**0**	0.04	**0**	0.04	**0**	**0.03**
64/1456	8			**149,573**	159.09	**149,573**	25.89	**149,573**	**15.87**
	9^*	**12,674**	**0.71**	**12,674**	8.31	**12,674**	1.83	**12,674**	1.40
`DSJC125.1`	4	**1,463**	**0.15**	3,093	0.92	3,093	0.84	3,094	0.89
125/736	5	**79,990**	**5.69**	401,777	290.58	401,777	77.40	408,691	110.30
	6^*	**1**	**0.03**	**1**	0.06	**1**	0.07	3	0.04
`miles250`	3	**6**	**0.03**	11	0.05	11	0.06	11	0.05
128/774	4	26,542,104	1,674.00	**770**	0.17	**770**	0.44	**770**	**0.12**
	5			**73,666**	11.68	**73,666**	34.24	**73,666**	**7.20**
`mulsol.i.1`	23			**5,040**	14.66	**5,040**	8.58	**5,040**	**1.52**
197/3925	24			**40,320**	133.15	**40,320**	77.73	**40,320**	**14.90**
	25					7,869,767	6,971.80	**362,880**	**129.76**
	26							**3,628,800**	**886.12**
`school1`	13			**762**	209.83	**762**	**13.82**	**762**	50.46
385/19095	14			**2,358**	1,079.66	**2,358**	**27.14**	**2,358**	56.71
	15^*			130	29.95	130	3.81	**105**	**3.16**
`fpsol2.i.2`	22			**720**	4.14	**720**	71.01	**720**	**0.77**
451/8691	23			**5,040**	20.00	**5,040**	292.47	**5,040**	**2.23**
	24			**40,320**	133.55	**40,320**	1,517.75	**40,320**	**14.38**
	25							**362,880**	**126.86**
`ash331GPIA`	3	24	**0.43**	**4**	0.47	**4**	0.77	**4**	0.49
662/4185	4^*	**1,563**	**1.01**	67,952	26.90	67,952	18.04	67,952	16.71
	5	125,240	47.15	**3,815**	3.51	**3,815**	2.86	**3,815**	**2.06**
TOTALS									
Instances solved/total			15/32		29/32		30/32		**32/32**
Best method/instances		9/32	10/32	20/32	0/32	22/32	4/32	**23/32**	**20/32**
Average (solved by all)		4,585,083	204.30	**51,952**	25.71	51,967	**7.95**	52,427	10.26
Average (all solved)		4,585,083	204.30	**43,430**	77.25	304,231	306.52	175,873	**46.24**

Table 1: Graph coloring problems: **b**ranches and **t**ime in seconds to find a proper coloring or prove one does not exist. Blank entries are problems not solved in 2 hours. Colorings marked with * are optimal, "Best method/instances" is the fraction of instances on which the method is best, "(solved by all)" is the average over the 15 instances all methods solve before the time-out, whilst "(all solved)" is the average over all instances solved by the given method.

all colors are interchangeable. We break these with one of the three methods mentioned above.

Results are given in Table 2. To look in more detail at the propagation, we ran this experiment using SICSTUS 3.12.7. This also provides statistics on the number of domain prunings performed in propagating constraints. For $n = 5$ and 7, there is an unique solution up to symmetry. For $n = 6$ and 8, there are no solutions. Despite the theoretical differences between the three static symmetric breaking methods identified in Theorems 4 and 5, we see no difference in the size of the search trees explored on these n by n queens problems. The specialized propagator for value precedence is, however, two or so times faster than Puget's method which itself is two or so time faster than the generator symmetry method. The more detailed statistics suggest that the value precedence constraint often reaches the same fixed point as Puget's method but with fewer domain prunings.

10 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 corresponding to generators of the group [Aloul *et al.*, 2002]. Aloul *et al.* also improved the runtime of this method by reducing the size of a CNF encoding of such a symmetry breaking constraint from quadratic to linear [Aloul *et al.*, 2003]. Petrie and Smith adapted this symmetry breaking method to value symmetries by posting a suitable lexicographical ordering constraint for each value symmetry [Petrie and Smith, 2003]. Puget and Walsh independently proposed propagators for such 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 formally defined value precedence for finite domain and set variables and proposed a specialized propagator for a pair of interchangeable values [Law and Lee, 2004]. Walsh extended this to a propagator for any number of interchangeable values [Walsh, 2006b]. Value precedence enforces the so-called "lowest index color ordering" which eliminates value symmetry in graph coloring problems [Aloul, 2006]. Finally, an alternative way to break value symmetry statically is to convert it into a variable symmetry by channelling into a dual viewpoint and using lexicographical ordering constraints on this dual view [Flener *et al.*, 2002; Law and Lee, 2006].

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 [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 and Hentenryck, 2005].

11 Conclusion

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 typically only perform forward checking and can take exponential time on problems which static methods solve without search. We have studied a common type of value symmetry where values are interchangeable and static methods are polynomial. We considered three different symmetry breaking constraints for interchangeable values: lexicographical ordering constraints based on generators of the symmetry group, symmetry breaking constraints proposed by Puget [Puget, 2005], and a specialized precedence constraint [Law and Lee, 2004; Walsh, 2006b]. We have shown that despite theoretical differences in their ability to prune symmetric values, the three methods appear to explore very similar search spaces in practice. However, the specialized propagator offers runtime savings by reaching its fixed point quicker. There are many open questions raised by this research. For example, are there other types of symmetry where all symmetric values can be pruned tractably? Are there other types of symmetry where it is enough to use just generators?

References

[Aloul *et al.*, 2002] F.A. Aloul, A. Ramani, I. Markov, and K.A. Sakallah. Solving difficult SAT instances in the presence of symmetries. In *Proceedings of the Design Automation Conference*, pages 731–736, 2002.

[Aloul *et al.*, 2003] F.A. Aloul, K.A. Sakallah, and I.L. Markov. Efficient symmetry breaking for Boolean satisfiability. In *Proceedings of the 18th International Joint Conference on AI*, pages 271–276. International Joint Conference on Artificial Intelligence, 2003.

[Aloul, 2006] F.A. Aloul. Breaking instance-independent symmetries in exact graph coloring. *Journal of Artificial Intelligence Research*, 26:289–322, 2006.

[Bessiere *et al.*, 2004] C. Bessiere, E. Hebrard, B. Hnich, and T. Walsh. The complexity of global constraints. In *Proceedings of the 19th National Conference on AI*. American Association for Artificial Intelligence, 2004.

[Crawford *et al.*, 1996] J. Crawford, G. Luks, M. Ginsberg, and A. Roy. Symmetry breaking predicates for search problems. In *Proceedings of the 5th International Conference on Knowledge Representation and Reasoning, (KR '96)*, pages 148–159, 1996.

problem	value symmetry breaking															
n	none				generator symmetries				Puget's method				value precedence			
	c	**b**	**p**	**t**	**c**	**b**	**p**	**t**	**c**	**b**	**p**	**t**	**c**	**b**	**p**	**t**
4	22	7	**219**	0.01	444	**1**	628	0.02	399	**1**	591	0.02	156	**1**	317	**0.00**
5	28	59	2,781	**0.02**	928	**2**	1,651	**0.02**	782	**2**	1,251	0.03	253	**2**	**601**	**0.02**
6	34	3949	200,395	0.65	1,654	**30**	9,624	0.07	1,335	**30**	7,245	0.07	358	**30**	**3,611**	**0.02**
7	40	882,813	53,528,368	170.75	2,686	**838**	278,678	1.20	2,104	**838**	193,901	0.67	481	**838**	**130,695**	**0.28**
8					4,078	**148,564**	54,091,553	238.52	3,125	**148,564**	36,865,615	119.83	622	**148,564**	**19,899,573**	**50.12**
9																

Table 2: n by n queens problem: **c**onstraints posted, **b**ranches, domain **p**runings and **t**ime to find all solutions in secs using the fail first heuristic. Blank entries are problems not solved in 1 hour. Results are similar to find first solution.

[Debruyne and Bessière, 1997] R. Debruyne and C. Bessière. Some practicable filtering techniques for the constraint satisfaction problem. In *Proceedings of the 15th IJCAI*, pages 412–417. International Joint Conference on Artificial Intelligence, 1997.

[Flener *et al.*, 2002] P. Flener, A. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, J. Pearson, and T. Walsh. Breaking row and column symmetry in matrix models. In *8th International Conference on Principles and Practices of Constraint Programming (CP-2002)*. Springer, 2002.

[Hentenryck *et al.*, 2003] P. Van Hentenryck, M. Agren, P. Flener, and J. Pearson. Tractable symmetry breaking for CSPs with interchangeable values. In *Proceedings of the 18th International Conference on AI*. International Joint Conference on Artificial Intelligence, 2003.

[Law and Lee, 2004] Y.C. Law and J.H.M. Lee. Global constraints for integer and set value precedence. In *Proceedings of 10th International Conference on Principles and Practice of Constraint Programming (CP2004)*, pages 362–376. Springer, 2004.

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

[Petrie and Smith, 2003] Karen E. Petrie and Barbara M. Smith. Symmetry Breaking in Graceful Graphs. Technical Report APES-56a-2003, APES Research Group, June 2003.

[Puget, 1993] J.-F. Puget. On the satisfiability of symmetrical constrained satisfaction problems. In J. Komorowski and Z.W. Ras, editors, *Proceedings of ISMIS'93*, LNAI 689, pages 350–361. Springer-Verlag, 1993.

[Puget, 2005] J-F. Puget. Breaking all value symmetries in surjection problems. In P. van Beek, editor, *Proceedings of 11th International Conference on Principles and Practice of Constraint Programming (CP2005)*. Springer, 2005.

[Puget, 2006] J-F. Puget. An efficient way of breaking value symmetries. In *Proceedings of the 21st National Conference on AI*. American Association for Artificial Intelligence, 2006.

[Roney-Dougal *et al.*, 2004] C. Roney-Dougal, I. Gent, T. Kelsey, and S. Linton. Tractable symmetry breaking using restricted search trees. In *Proceedings of ECAI-2004*. IOS Press, 2004.

[Sellmann and Hentenryck, 2005] M. Sellmann and P. Van Hentenryck. Structural symmetry breaking. In *Proceedings of 19th IJCAI*. International Joint Conference on Artificial Intelligence, 2005.

[Walsh, 2006a] T. Walsh. General symmetry breaking constraints. In *12th International Conference on Principles and Practices of Constraint Programming (CP-2006)*. Springer-Verlag, 2006.

[Walsh, 2006b] T. Walsh. Symmetry breaking using value precedence. In *Proceedings of the 17th ECAI*. European Conference on Artificial Intelligence, IOS Press, 2006.