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. We show that despite these theoretical limitations, methods proposed to deal with interchangeable values are both effective in theory and in practice.

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. 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 & Lee 2004; Puget 2005), appear to be effective at dealing with common types of value symmetry.

[*]Funded by 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.

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. A solution is an assignment of values to variables satisfying the constraints. Finite domain variables take one value from a given finite set. Set variables take sets of such values and are typically defined by a lower bound on the definite elements in the set and an upper bound on the definite and potential elements.

Many constraint solvers explore the space of partial assignments enforcing some local consistency. We consider three local consistencies for finite domain variables as well as the most common local consistency for set variables. Given a constraint C on finite domain variables, a *support* is assignment to each variable of a value in its domain which satisfies C. A constraint on finite domains variables 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. Given a constraint C on set variables, a *bound support* on C is an assignment of a set to each set variable between its lower and upper bounds which satisfies C. A constraint is *bound consistent* (*BC*) iff for each set variable S, the values in $ub(S)$ belong to S in at least one bound support and the values in $lb(S)$ belong to S in all bound supports. A set of constraints is BC iff each constraint is BC.

Symmetry occurs in many constraint satisfaction problems. A *value symmetry* is a bijection on values that preserves solutions. For example, suppose we wish to assign colors (values) to nodes (variables) in a graph coloring problem. Value symmetry permits us to interchange any two colors uniformly throughout a coloring. A *variable symmetry*, on the other hand, is a bijection on variables that preserves solutions. 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 $m!$.

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 $\textsc{ValSymBreak}(\Sigma, [X_1, .., X_n])$ which ensures $[X_1, .., X_n] \leq_{\text{lex}} [\sigma(X_1), .., \sigma(X_n)]$ for all $\sigma \in \Sigma$ and X_1 to X_n is a fixed ordering on the variables. Unfortunately, pruning *all* values from such a symmetry breaking constraint is NP-hard.

Theorem 1 *Deciding if* $\textsc{ValSymBreak}(\Sigma, [X_1, .., X_n])$ *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 first construct a CSP with $N + M + 1$ variables over $4N$ possible values which partition into $2N$ interchangeable pairs. $4i - 3$ and $4i - 2$ are interchangeable, as are $4i-1$ and $4i$ for $1 \leq i \leq N$. $4i-3$ and $4i-2$ represent x_i being true, whilst $4i-1$ and $4i$ represent x_i being false. 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} has the domain $\{1, 2\}$. Note that all variables have symmetric domains. If a value is in the domain of a variable then so are all its symmetries. We next add two sets of constraints. First, we have the constraints $odd(X_{N+M+1}) \rightarrow odd(X_i)$ and $odd(X_{N+M+1}) \rightarrow even(X_{N+j})$ for $1 \leq i \leq N$ and $1 \leq j \leq M$. Second, we introduce constraints over fresh variables and values encoding an unsatisfiable CSP (say, $N + 1$ pigeons in N pigeonholes). Note that the constructed CSP is unsatisfiable. Thus, it trivially has the property that any symmetry of a solution is also a solution.

Suppose our branching heuristic assigns $X_{N+M+1} = 1$. Enforcing AC on the 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 $\textsc{ValSymBreak}$. 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, there is a support for $\textsc{ValSymBreak}$ iff the original 3-SAT problem is satisfiable. By Theorem 3, $|\Sigma|$ can be linearly bound. $\diamond$

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.

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 can dynamically eliminate all symmetric solutions 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* variable. Unlike static methods, the GE-tree method does not prune *deeper* variables. By pruning the domains of deeper variables, static symmetry breaking methods can solve certain problems exponentially quicker than dynamic methods.

Theorem 2 *There exists a class of CSP 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 $X_i \in \{1, .., n + 1\}$ 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. Eenforcing 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. $\diamond$

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: we will suppose that values are ordered into partitions, and values within each partition are uniformly interchangeable. We will consider three static methods proposed to break such symmetry.

Generator symmetries

One way to propagate $\textsc{ValSymBreak}$ is to decompose it into individual lexicographical ordering constraints,

$[X_1, .., X_n] \leq_{\text{lex}} [\sigma(X_1), .., \sigma(X_n)]$ 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, for instance, 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. Breaking *just* these symmetries eliminates *all* symmetric solutions.

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

Proof: Assume $\text{VALSYMBREAK}(\Sigma_g, [X_1, .., X_n])$. 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 $[X_1, .., X_n] \leq_{\text{lex}} [\sigma_j(X_1), .., \sigma_j(X_n)]$, 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, $[X_1, .., X_n] \leq_{\text{lex}} [\sigma'(X_1), .., \sigma'(X_n)]$. Consider now any symmetry $\sigma \in \Sigma$. Suppose $[X_1, .., X_n] >_{\text{lex}} [\sigma(X_1), .., \sigma(X_n)]$. 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, $[X_1, .., X_n] \leq_{\text{lex}} [\sigma'(X_1), .., \sigma'(X_n)]$. But this contradicts $[X_1, .., X_n] >_{\text{lex}} [\sigma(X_1), .., \sigma(X_n)]$ as σ and σ' act identically on the first j variables in X_1 to X_n. Hence, $[X_1, .., X_n] \leq_{\text{lex}} [\sigma(X_1), .., \sigma(X_n)]$ for any $\sigma \in \Sigma$. $\diamond$

Not surprisingly, reducing the number of symmetry breaking constraints to linear comes at a cost. We may not prune all symmetric values.

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

Proof: 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 $\text{VALSYMBREAK}(\Sigma, [X_1, .., X_5])$ prunes 1 from X_2 to X_4. However, consider the generator $\sigma \in \Sigma_g$ which interchanges i with $i+1$ where $1 \leq i < 5$. Then $[X_1, .., X_5] \leq_{\text{lex}} [\sigma(X_1), .., \sigma(X_5)]$ is GAC without any domain pruning. $\diamond$

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 *Enforcing GAC on* $\text{VALSYMBREAK}(\Sigma, [X_1, .., X_n])$ *is strictly stronger than enforcing AC on Puget's decomposition.*

Proof: It is not hard to show it as least as strong. To show it is strictly 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 $\text{VALSYMBREAK}(\Sigma, [X_1, .., X_5])$ will prune 1 from X_2. $\diamond$

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 SAC on Puget's binary decomposition will prune all symmetric values. The argument is as follows. Enforcing AC on Puget's encoding ensures that there is at least one support for VALSYMBREAK in the domain of every variable. Hence enforcing SAC on Puget's decomposition ensures that VALSYMBREAK is GAC. More generally, if values partition into j interchangeable classes, we may need to make the problem k-consistent for all $k \leq j+1$ to prune all symmetric values. As j may not be bounded, *local* consistency is not enough to prune all symmetric values.

Finally, we compare this method with the previous method based on breaking the symmetries corresponding to each generator. We can show that we may prune more symmetric values using Puget's decomposition.

Theorem 6 *If Σ is a set of symmetries induced by interchangeable values, and Σ_g is the set of generators interchanging adjacent values then enforcing AC on Puget's decomposition is strictly stronger than enforcing GAC on $[X_1, .., X_n] \leq_{\text{lex}} [\sigma(X_1), .., \sigma(X_n)]$ for each $\sigma \in \Sigma_g$.*

Proof: Consider the example used in the proof of Theorem 4. Enforcing AC on Puget's decomposition prunes 1 from X_2 to X_4. However, the lexicographical ordering constraint for each generator is GAC without any domain pruning. $\diamond$

Value precedence

A third method to eliminate all symmetric solutions statically is based on global *precedence* constraints (Law & Lee 2004). Suppose Σ is the set of symmetries induced by all values being interchangeable. The global constraint $\text{PRECEDENCE}([X_1, .., X_n])$ 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 $\text{PRECEDENCE}([X_1, .., X_n])$ is equivalent to $\text{VARSYMBREAK}(\Sigma, [X_1, .., X_n])$. 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 interchangeable classes. 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.

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 Σ, we post the global constraint $\text{VARSYMBREAK}(\Sigma, [X_1, .., X_n])$ which ensures $[X_1, .., X_n] \leq_{\text{lex}} [X_{\sigma(1)}, .., X_{\sigma(n)}]$ for all $\sigma \in \Sigma$ 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 (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: $[X_1, .., X_n] \leq_{\text{lex}} [X_n, .., X_1]$.

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)).

Set variables

Value symmetry can be eliminated from problems containing set variables in similar ways. For example, we can post a (global) constraint which ensures that the solution is ordered lexicographically before any of the symmetries of the solution. More precisely, we post $\text{VALSYMBREAK}(\Sigma, [S_1, .., S_n])$ which ensures $[S_1, .., S_n] \leq_{\text{lex}} [\sigma(S_1), .., \sigma(S_n)]$ for all $\sigma \in \Sigma$ where S_1 to S_n is a fixed ordering on the set variables and $\leq_{\text{lex}}$ is the lexicographical extension of the multiset ordering on sets. The multiset ordering on sets is identical to the lexicographical ordering on the characteristic function of the sets.

For set variables taking interchangeable values, pruning all symmetric values is polynomial. In (Walsh 2006b), it is shown that all symmetric solutions introduced by set variables taking interchangeable values can be eliminated by lexicographically ordering the columns of the 2-d matrix constructed by mapping the sequence of set variables onto their characteristic functions. If values partition into interchangeable classes, we need merely to order lexicographically those pairs of columns which are interchangeable. We can achieve BC on this representation (and thus prune all symmetric values) in polynomial time. On the other hand, with arbitrary value symmetries, pruning all symmetric values is NP-hard.

Theorem 7 *For set variables taking symmetric values, enforcing BC on* $\text{VALSYMBREAK}(\Sigma, [S_1, .., S_n])$ *is NP-hard.*

Proof: Consider finding an assignment to a single set variable that is lexicographically smaller than or equal to any of its symmetries. We give a reduction from 3-SAT. Given a 3-SAT problem in N variables, x_i to x_N and M clauses, we construct a set variable S with lower bound $\{4i-1 | 1 \leq i \leq N\} \cup \{4N+2\}$ and upper bound $\{4i, 4i-1, 4i-2 \mid 1 \leq i \leq N\} \cup \{4N+1, 4N+2, 4N+3, 4N+4, 4N+5\}$. The interpretation of $4i \in S$ for $1 \leq i \leq N$ is that x_i is true, and of $4i-2 \in S$ for $1 \leq i \leq N$ is that x_i is false. $4i-1$ for $1 \leq i \leq N$ are "dummy" values used to ensure that we don't have both $4i \in S$ and $4i-2 \in S$ (in other words, x_i is not assigned to both true and false). $4N+1$ is an additional "dummy" value used to ensure that at least one literal in every clause is true.

For each i from 1 to N, we construct the value symmetry which permutes $4i$ with $4i-1$, $4i-2$ with $4i-3$, and leaves all other values are unchanged. Since $4i-1 \in lb(S)$, for S to be lexicographical smaller than or equal to $\sigma(S)$, $4i \in S$ implies $4i-2 \notin S$, and $4i-2 \in S$ implies $4i \notin S$. That is, we cannot have both x_i set to true and to false. It may be that both $4i \notin S$ and $4i-2 \notin S$. In other words, we may not have assigned any truth value to x_i. In this case, all clauses will be satisfied without us needing to chose a truth value for x_i.

To ensure that the truth assignment represented by S satisfies the clauses, we construct a value symmetry corresponding to each clause. Suppose we have the clause: $x_i \vee \neg x_j \vee x_k$. Then we construct the value symmetry which permutes $4i$ with $4N+3$, $4j-2$ with $4N+4$, $4k$ with $4N+5$, and $4N+1$ with $4N+2$, leaving all other values unchanged. Since $4N+2 \in lb(S)$, $S \leq_{\text{lex}} \sigma(S)$ and σ swaps $4N+1$ with $4N+2$ it follows that at least one of $4i$, $4j-2$ or $4k$ must also be in S. That is, at least one of x_i, $\neg x_j$ or x_k must be set to true.

Thus, there is an assignment to S which is lexicographically smaller than or equal to any of its symmetries iff the original 3SAT problem has a satisfying assignment. Hence, finding a support (and thus enforcing BC on a complete symmetry breaking constraint) is NP-hard. $\diamond$

As the proof only used a single set variable, it also follows that dynamic methods which do not assign symmetric values to set variables are NP-hard to compute in general.

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).

We coded all problems in SICSTUS 3.12.7, and ran them on a PowerPC 1GHz G4 processor with 1.25 GB RAM.

n by n queens

As in the first experiment 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 colour 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 finite domain 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 all colors are interchangeable. We break these with one of the three methods mentioned above.

Results are given in Table 1. 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 in practice on these 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.

Schur numbers

We also ran experiments on the Schur numbers problem (prob015 in CSPLib). This problem was used in previous experimental studies of value precedence (Law & Lee 2004; Walsh 2006b). The Schur number $S(k)$ is the largest integer n for which the interval $[1, n]$ can be partitioned into k sum-free sets. S is sum-free iff $\forall a, b, c \in S \,.\, a \neq b + c$. Schur numbers are related to Ramsey numbers, $R(n)$ through the identity: $S(n) \leq R(n) - 2$. Schur numbers were proposed by the famous German mathematician Isaai Schur in 1918. $S(4)$ was open until 1961 when it was first calculated by computer. $S(3)$ is 13, $S(4)$ is 44, and $160 \leq S(5) \leq 315$. We consider the corresponding decision problem, $S(n, k)$ which asks if the interval $[1, n]$ can be partitioned into k sum-free sets. A simple model of this uses n finite domain variables with k interchangeable values.

Results are given in Table 2. We now see slight difference in the size of the search trees explored using the different static symmetry breaking methods. We conjecture that these differences are probably a consequence of the fail first heuristic deciding to branch on the dual variables.

To conclude, over the two problem sets, the specialized propagator for value precedence is either as fast or faster than Puget's method, and both are typically two or more times faster than the generator symmetry method. It is an interesting open question how the methods will compare on problems with more classes of interchangeable values. We conjecture that we may then see more significant differences in the size of the search trees.

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). Petrie and Smith adapted this method to value symmetries by posting a suitable lexicographical ordering constraint for each value symmetry (Petrie & 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 symmetry breaking constraints, Puget proposed a global propagator which does forward checking (Puget 2006).

To eliminate symmetric solutions due to interchangeable values, Law and Lee formally defined value precedence 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). 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 & 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 & Hentenryck 2005).

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	2781	**0.02**	928	**2**	1651	**0.02**	782	**2**	1251	0.03	253	**2**	**601**	**0.02**
6	34	3949	200395	0.65	1654	**30**	9624	0.07	1335	**30**	7245	0.07	358	**30**	**3611**	**0.02**
7	40	882813	53528368	170.75	2686	**838**	278678	1.20	2104	**838**	193901	0.67	481	**838**	**103695**	**0.28**
8					4078	**148564**	54091553	238.52	3125	**148564**	36865615	119.83	622	**148564**	**19899573**	**50.12**
9																

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

problem	value symmetry breaking															
$S(n,k)$	none				generator symmetries				Puget’s method				value precedence			
	c	**b**	**p**	**t**	**c**	**b**	**p**	**t**	**c**	**b**	**p**	**t**	**c**	**b**	**p**	**t**
$S(13,3)$	126	294	9253	0.04	360	**49**	3435	0.04	362	**49**	2739	**0.03**	243	**49**	**2449**	**0.03**
$S(13,4)$	126	1331688	14584107	36.00	477	**51099**	1549607	3.52	441	**51099**	955356	1.96	243	**51099**	**805273**	**1.94**
$S(13,5)$					594	**691700**	22014534	48.97	520	**691700**	12518244	26.18	243	**691700**	**10901646**	**25.73**
$S(13,6)$									599	**2473322**	46404167	98.51	243	2474354	**38681737**	**47.44**
$S(14,3)$	147	456	15190	0.06	399	**76**	5042	0.04	401	**76**	4108	0.05	273	**76**	**3851**	**0.03**
$S(14,4)$	147	2748840	35774652	84.42	525	**103610**	3321327	7.00	486	**103610**	2063297	4.24	273	**103610**	**1859627**	**4.20**
$S(14,5)$					651	**2183885**	74079956	160.32	571	**2183885**	40990610	88.06	273	**2183885**	**37962162**	**86.85**
$S(14,6)$									656	**10437102**	200486641	432.79	273	10441664	**183274168**	**424.49**
$S(15,3)$	168	600	21287	0.08	438	**100**	6652	0.05	440	**100**	5483	**0.04**	303	**100**	**5305**	0.05
$S(15,4)$	168	6976512	93094291	213.79	573	**265060**	8449402	17.70	531	**265060**	5368238	10.89	303	**265060**	**4637648**	**10.29**
$S(15,5)$									622	**194209**	159071864	**238.33**	303	**194209**	**137221068**	318.19
$S(15,6)$																

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

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, pruning all symmetric values is NP-hard in general. 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: ordering constraints based on generators of the symmetry group, constraints proposed in (Puget 2005), and a specialized precedence constraint. We have shown that despite theoretical differences in their ability to prune symmetric values, the three methods explore very similar search spaces in practice. However, the specialized precedence constraint appears to offer a runtime advantage. 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, 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 *5th Int. Conf. on Knowledge Rep. and Reasoning, (KR ’96).*

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).*

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

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

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

Petrie, K. E., and Smith, B. M. 2003. Symmetry Breaking in Graceful Graphs. Technical Report APES-56a-2003.

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

Puget, J.-F. 2005. Breaking all value symmetries in surjection problems. In *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 Hentenryck, P. V. 2005. Structural symmetry breaking. In *Proc. of 19th IJCAI.*

Walsh, T. 2006a. General symmety 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.* IOS Press.