Posting Static Symmetry Breaking Constraints Dynamically

George Katsirelos and **Toby Walsh**
NICTA and UNSW, Sydney, Australia
george.katsirelos,toby.walsh@nicta.com.au

Abstract

Symmetry is an important feature of many combinatorial search problems. We propose a new method for dynamically posting static symmetry breaking constraints during search. This method is based on the observations that *any symmetry* of a set of symmetry breaking constraints can be used to break symmetry, and different symmetries pick out different solutions. We choose which symmetry to post as branching decisions force the choice. Unlike other dynamic methods, we are not restricted to breaking symmetry with lexicographical ordering constraints. We prove that our method is correct and only eliminates symmetric solutions. We also identify some common conditions under which it eliminates all symmetric solutions. This approach inherits good properties of both dynamic and static symmetry breaking methods: we can have fast propagation on the posted symmetry breaking constraints without conflicting with the branching heuristic. Experimental results show that the method performs well.

Introduction

Many combinatorial search problems contain symmetry. For instance, the colours in a graph colouring problem are interchangeable. If we have a proper colouring of a graph, we can permute the colours and still have a proper colouring. As a second example, the bins in a bin packing problem are interchangeable. If we have a packing into bins, we can swap any two bins and still have a packing. The presence of symmetry hinders the search effort, as the search procedure might need to explore exponentially many symmetric parts of the search space that contain no solutions. Therefore, we need to factor such symmetry out of the search space to be able to find solutions efficiently.

In constraint programming, one way to deal with symmetry is to add constraints *statically* which eliminate symmetric solutions (see, for instance, (Puget 1993; Shlyakhter 2001; Flener et al. 2002; Aloul, Sakallah, and Markov 2003; Law and Lee 2006; Walsh 2006)). For example, we can add constraints which limit search to the lexicographically least solution in each symmetry class (Crawford et al. 1996). Another way to deal with symmetry (which is not limited to constraint programming) is to modify the search procedure so that it *dynamically* avoids symmetric solutions (see, for instance, (Fahle, Schamberger, and Sellmann 2001; Gent and Smith 2000; Roney-Dougal et al. 2004)). Each method has advantages and disadvantages. Static methods are usually simple to implement and are often highly effective. Even for problems with many symmetries, a small number of static symmetry breaking constraints can often eliminate much or all of the symmetry. However, static methods pick out particular solutions in each symmetry class, and the branching heuristic may conflict with this choice. Dynamic methods, on the other hand, do not conflict with the branching heuristic, but often perform poorly when there are many symmetries. In addition, dynamic methods do not prune the search space. With static methods, propagation between problem and symmetry breaking constraints can eliminate large parts of the search tree.

We propose here a new method for breaking symmetry that combines together good features of both approaches. This method can post any type of static symmetry breaking constraint dynamically during search. The method makes symmetry breaking insensitive to the branching heuristic, whilst still pruning the search space by propagation. Whilst the method has been developed in the context of constraint programming, it could be easily adapted to work in other types of combinatorial search.

Background

A constraint satisfaction problem (CSP) is of a set of variables X_i, each with a domain of values, and a set of constraints C specifying allowed combinations of values for subsets of variables. A solution is an assignment to the variables satisfying the constraints. We write $sol(C)$ for the set of all solutions to the constraints C. A common method to find a solution of a constraint satisfaction problem is backtracking search. Constraint solvers typically prune the backtracking search space by enforcing a local consistency property like domain consistency. A constraint is *domain consistent* iff for each variable, every value in its domain can be extended to an assignment satisfying the constraint. We make a constraint domain consistent by pruning values for variables which cannot be in any satisfying assignment. During the search for a solution, a constraint can become entailed. A constraint is *entailed* when any assignment of values from the respective domains satisfies the constraint, For instance, $X_1 < X_n$ is entailed iff the largest value in the domain of

X_1 is smaller than the smallest value in the domain of X_n. A constraint is and *dis-entailed* when its negation is entailed. For instance, $X < Y$ is dis-entailed if and only if the smallest value in the domain of X is larger than or equal to the largest value in the domain of Y.

Constraint satisfaction problems can contain symmetry. Although *detecting* symmetry is an active area in research, we concern ourselves here with *exploiting* symmetry to improve search. We consider two types of symmetry (see (Cohen et al. 2006) for more discussion). A *variable symmetry* is a permutation of the variables that preserves solutions. Formally, a variable symmetry is a bijection σ on the indices of variables such that if $X_1 = d_1, \ldots, X_n = d_n$ is a solution then $X_{\sigma(1)} = d_1, \ldots, X_{\sigma(n)} = d_n$ is also. A *value symmetry*, on the other hand, is a permutation of the values that preserves solutions. Formally, a value symmetry is a bijection θ on the values such that if $X_1 = d_1, \ldots, X_n = d_n$ is a solution then $X_1 = \theta(d_1), \ldots, X_n = \theta(d_n)$ is also. Symmetries can more generally act on both variables and values. Our methods also work with such symmetries. As the inverse of a symmetry and the identity mapping are symmetries, the set of symmetries of a problem forms a group under the composition operator $\circ$. We will use a simple running example which has a small number of variable and value symmetries. This example demonstrates that we can use symmetry itself to pick out different solutions in each symmetry class. The computational challenge is to adapt this simple idea to deal with large number of symmetries.

Running Example. *The all interval series problem (prob007 in CSPLib.org) asks for a permutation of 0 to $n-1$ so that neighbouring differences form a permutation of 1 to $n-1$. We model this as a CSP with $X_i = j$ iff the ith number is j. One solution for $n = 11$ is:*

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

The differences form a permutation of 1 to 10:

$$4, 3, 2, 1, 5, 10, 9, 8, 7, 6$$

This model has a number of different symmetries. First, there is a variable symmetry σ_{rev} that reverses any solution:

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

Second, there is a value symmetry θ_{inv} that inverts values. If we subtract all values in (a) from 10, we generate a second (but symmetric) solution:

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

Third, we can do both. By reversing and inverting (a), we generate a fourth (but symmetric) solution:

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

The model thus has four symmetries in total: σ_{id} (the identity mapping), σ_{rev}, θ_{inv}, and $\theta_{inv} \circ \sigma_{rev}$. ♣

Symmetry breaking

One method to deal with symmetry is to add constraints to eliminate symmetric solutions (Puget 1993). Two important properties of symmetry breaking constraints are soundness and completeness. A set of symmetry breaking constraint is sound iff it leaves at least one solution in each symmetry class, and complete iff it leaves exactly one solution.

Running Example. *Consider again the all interval series problem. To eliminate the reversal symmetry σ_{rev}, we can post the constraint:*

$$X_1 < X_{11} \quad (1)$$

This eliminates solution (b) as it is the reversal of (a). To eliminate the value symmetry θ_{inv}, we can post:

$$X_1 \leq 5, \qquad X_1 = 5 \Rightarrow X_2 < 5 \quad (2)$$

This eliminates solution (c) as it is the inversion of (a). Finally, to eliminate the third symmetry $\theta_{inv} \circ \sigma_{rev}$ where we both reverse and invert the solution, we can post:

$$\langle X_1, \ldots, X_6 \rangle \quad \leq_{lex} \quad \langle 10 - X_{11}, \ldots, 10 - X_6 \rangle \quad (3)$$

This eliminates solution (a) as it is the reversal and inversion of (d). Note that of the four symmetric solutions given earlier, only (d) with $X_1 = 2$ and $X_{11} = 7$ satisfies all these symmetry breaking constraints. The other three solutions are eliminated. ♣

Our symmetry breaking method is based on the observations that *any symmetry* of the original problem, when acting on a set of symmetry breaking constraints, produces another set of symmetry breaking constraints. Moreover, each symmetry of a set of symmetry breaking constraints picks out a different solution in each equivalence class. We let branching choose which symmetry to post during search. This requires us to consider the action of a symmetry on a symmetry breaking constraint. We have defined the symmetry of an assignment as the result of a symmetry acting on an assignment. We can lift this definition to symmetries acting on symmetry breaking constraints, producing symmetries of symmetry breaking constraints. The action of a variable symmetry on a constraint changes the variables on which the constraint acts. More precisely, a variable symmetry σ applied to the constraint $C(X_j, \ldots, X_k)$ gives $C(X_{\sigma(j)}, \ldots, X_{\sigma(k)})$. The action of a value symmetry is also easy to compute. A value symmetry θ applied to the constraint $C(X_j, \ldots, X_k)$ gives $C(\theta(X_j), \ldots, \theta(X_k))$.

Running Example. *To illustrate how we can break symmetry in the all interval series problem with the symmetry of a set of symmetry breaking constraints, we consider symmetries of (1), (2) and (3). These break the reflection and inversion symmetries of the all interval series problem.*

If we apply σ_{rev} to (1), we get an ordering constraint that again breaks the reversal symmetry:

$$X_{\sigma_{rev}(1)} < X_{\sigma_{rev}(11)}$$

This simplifies to:

$$X_{11} < X_1$$

If we apply σ_{rev} to (2), we get constraints that again break the inversion symmetry:

$$X_{11} \leq 5, \qquad X_{11} = 5 \Rightarrow X_{10} < 5$$

Finally, if we apply σ_{rev} to (3), we get a constraint that again breaks the combined reversal and inversion symmetry:

$$\langle X_{11}, \ldots, X_6 \rangle \quad \leq_{lex} \quad \langle 10 - X_1, \ldots, 10 - X_6 \rangle$$

Note that of the four symmetric solutions given earlier, only (c) satisfies σ_{rev} of (1), (2) and (3).

We can also break symmetry with any other symmetry of the symmetry breaking constraints. For instance, if we apply $\theta_{inv} \circ \sigma_{rev}$ to (1), we get a constraint that again breaks the reversal symmetry:

$$10 - X_{11} < 10 - X_1$$

This simplifies to:

$$X_1 < X_{11}$$

If we apply $\theta_{inv} \circ \sigma_{rev}$ to (2), we get constraints that again breaks the inversion symmetry:

$$10 - X_{11} \leq 5, \qquad 10 - X_{11} = 5 \Rightarrow 10 - X_{10} < 5$$

This simplifies to:

$$X_{11} \geq 5, \qquad X_{11} = 5 \Rightarrow X_{10} > 5$$

Finally, if we apply $\theta_{inv} \circ \sigma_{rev}$ to (3), we get a constraint that again breaks the combined reversal and inversion symmetry:

$$\langle 10 - X_{11}, \ldots, 10 - X_6 \rangle \quad \leq_{lex} \quad \langle X_1, \ldots, X_6 \rangle$$

Note that of the four symmetric solutions given earlier, only (a) satisfies $\theta_{inv} \circ \sigma_{rev}$ of (1), (2) and (3). ♣

The running example illustrates that we can break symmetry with a symmetry of a set of symmetry breaking constraints. We now prove that this holds in general:

Any symmetry of a set of symmetry breaking constraints itself breaks symmetry.

More precisely, if a set of symmetry breaking constraints is sound, then any symmetry of these constraints is also sound. Similarly, if a set of symmetry breaking constraints is complete, then any symmetry of these constraints is also complete. In addition, different symmetries of the symmetry breaking constraints pick out different solutions in each symmetry class.

Theorem 1. *Given a set of symmetries Σ of C, if S is a sound (complete) set of symmetry breaking constraints for Σ then $\sigma(S)$ for any $\sigma \in \Sigma$ is also a sound (complete) set of symmetry breaking constraints for Σ.*

Proof: (Soundness) Consider $s \in sol(C \cup S)$. Then $s \in sol(C)$ and $s \in sol(S)$. We exploit the fact that σ^{-1}, the inverse of σ is itself a symmetry of C. Hence $\sigma^{-1}(s) \in sol(C)$. Since $s \in sol(S)$, it follows that $\sigma^{-1}(s) \in sol(\sigma(S))$. Thus, $\sigma^{-1}(s) \in sol(C \cup \sigma(S))$. Hence, there is at least one solution, $\sigma^{-1}(s)$ in every symmetry class of $C \cup \sigma(S)$. That is, $\sigma(S)$ is a sound set of symmetry breaking constraints for Σ.

(Completeness) Consider $s \in sol(C \cup \sigma(S))$. By a similar argument to soundness, $\sigma^{-1}(s) \in sol(C \cup S)$. Hence, there is at most one solution in every symmetry class of $C \cup \sigma(S)$. That is, $\sigma(S)$ is a complete set of symmetry breaking constraints for Σ. □

Posting constraints dynamically

We use these observations to post symmetry breaking constraints dynamically during search that do not conflict with the branching decisions. We will post the symmetry of the symmetry breaking constraints which is consistent with the branching decisions made so far. Thus, if the branching heuristic is smart or lucky enough to branch immediately to a solution, symmetry breaking will not interfere with this.

Running Example. *Consider again the all interval series problem. Suppose we begin by trying $X_1 = 10$. Since the X_i are all different, $X_{11} \in [0, 9]$. Hence, the symmetry breaking constraint $X_{11} < X_1$ is entailed. This is σ_{rev} of (1). It is also θ_{inv} of (1). We do not yet need to commit to which of these two symmetries of the symmetry breaking constraints we will post. We are sure, however, that we are not posting σ_{id} or $\theta_{inv} \circ \sigma_{rev}$ of the constraints (1) to (3). These two symmetries would require $X_1 > X_{11}$, and this is dis-entailed. We therefore post $X_{11} < X_1$ and continue search. This constraint now becomes part of the model and is never retracted.* ♣

The example demonstrates that we post constraints that are a symmetry of a symmetry breaking constraint once they are entailed. When there are only a few symmetries, we can easily implement this with non-backtrackable variables and reification. Suppose we reify the two ordering constraints:

$$B_1 \Leftrightarrow (X_1 < X_{11}), \qquad B_2 \Leftrightarrow (X_{11} < X_1)$$

We then make the Boolean variables, B_1 and B_2 non-backtrackable so that, once they are instantiated, their value remains on backtracking. We assume that our solver posts the conclusion of an implication when its hypothesis is entailed. Suppose $X_1 < X_{11}$ is entailed. Then B_1 will be set $true$. As B_1 is non-backtrackable, $X_1 < X_{11}$ will be posted. Unfortunately, posting symmetry breaking constraints immediately when they are entailed can sometimes be too eager.

Running Example. *Consider again the all interval series problem. As before, suppose backtracking has set $X_1 = 10$, and we have posted the entailed symmetry breaking constraint $X_{11} < X_1$. Now $X_1 \geq 5$ is also entailed. This is θ_{inv} of the first inequality in (2). If we post this, we commit to breaking symmetry with θ_{inv} of (1) to (3). However, this would rule out breaking symmetry with σ_{rev} of (1) to (3) which are also still consistent with the branching decisions so far.*

Suppose we next try $X_{11} = 5$. The assignments to X_1 and X_{11} are only consistent with θ_{inv} of (2) and of (3). In fact, both of these constraints are now entailed. However, $X_1 = 10$ and $X_{11} = 5$ are not consistent with posting σ_{rev} of (3). This would require that:

$$\langle X_{11}, \ldots, X_6 \rangle \quad \leq_{lex} \quad \langle 10 - X_1, \ldots, 10 - X_6 \rangle$$

This is dis-entailed. Hence, our branching decisions have committed us to break symmetry with θ_{inv} of (1) to (3). We therefore post these constraints. If search continues, we will discover the unique solution consistent with symmetry breaking and the initial branching decisions:

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

♣

Note that in a given node of the search tree, it may be the case that no new constraints are entailed. In this case, the branching heuristic has not committed to a particular symmetry and therefore we do not add any constraints to the model.

Least commitment rule

We can formalize the symmetry breaking described in the running example with a simple rule to decide when to post dynamically a static symmetry breaking constraint. The proposed rule is least committing as it only posts symmetry breaking constraints once the branching heuristic has forced their choice. Suppose S is a set of symmetry breaking constraints for Σ, and we have posted T, a symmetry of a subset of S. A symmetry $\sigma \in \Sigma$ is *consistent* with T iff T is entailed by $\sigma(S)$ and *inconsistent* otherwise. A symmetry $\sigma \in \Sigma$ is *eliminated* by posting some symmetry breaking constraint c iff σ is consistent with T but inconsistent with $T \cup \{c\}$. We consider the following *least commitment* rule for incrementally posting symmetry breaking constraints:

> *Given a set of symmetry breaking constraints, if during backtracking search a symmetry of one of these constraints is entailed, this symmetry is consistent with previously posted symmetry breaking constraints, and all symmetries eliminated by this entailed constraint are inconsistent with the current state then we post the entailed constraint.*

We first show that this rule is sound.

Theorem 2. *Given a set of symmetries Σ of C, if S is a sound set of symmetry breaking constraints for Σ then the least commitment rule using S is a sound symmetry breaking method.*

Proof: The rule only permits constraints of a particular symmetry to be posted. By Theorem 1, this is sound. □

In general, this rule may not be complete even when given a complete set of symmetry breaking constraints. However, it is easy to modify the rule so that it is complete. Whenever we reach a solution, we simply pick a consistent symmetry and post all the symmetry breaking constraints associated with this symmetry. We can also define a common property of many symmetry breaking constraints for which the unmodified rule is complete. A set of symmetry breaking constraints S for the symmetries Σ of C is *proper* iff S is sound and complete for Σ and every non-identity symmetry in Σ maps any solution of $S \cup C$ onto a different solution. With a proper set of symmetry breaking constraints, each solution within a symmetry class is associated with a different symmetry. For instance, constraints (1) to (3) form a proper set of symmetry breaking constraints. Static symmetry breaking constraints for frequently occuring symmetries like value interchangeability (Walsh 2006) and partial variable and value interchangeability (Law et al. 2007) are proper. This is, however, not the case for complete variable interchangeability.

We now prove that with a proper set of symmetry breaking constraints, the least commitment rule is a sound *and* complete symmetry breaking method. That is, it will find exactly one solution in each symmetry class.

Theorem 3. *Given a set of symmetries Σ of C, if S is a proper set of symmetry breaking constraints for Σ then the least commitment rule is both sound and complete.*

Proof: (Soundness) Immediate as a proper set is sound.

(Completeness) Consider the first solution visited. As the set of symmetry breaking constraints is proper, only one symmetry of these constraints will be entailed. All other symmetries are inconsistent with the current state and are eliminated. The least commitment rule therefore post this symmetry of the symmetry breaking constraints. By Theorem 1, as the symmetry breaking constraints are complete, this eliminates all other solutions in the same symmetry class. □

Finally, we observe that with certain symmetry breaking constraints, the least commitment rule is equivalent to posting symmetry breaking constraints as soon as they are entailed. For symmetry breaking constraints like $X_1 < X_{11}$, as soon as the constraint or its negation is entailed, all variable symmetries are either consistent or they are eliminated.

Comparison with SBDS

Perhaps closest in spirit to our method is SBDS. This also posts static symmetry breaking constraints dynamically during search according to the choices made by the branching heuristic (Backofen and Will 1999; Gent and Smith 2000; Backofen and Will 2002). SBDS can work with any type of branching decision but for simplicity we assume that branching decisions are of the form $Var = val$. All current implementations of SBDS make this assumption. If we have a symmetry σ, the partial assignment A and have explored and rejected $Var = val$ then on backtracking, SBDS posts:

$$\sigma(A) \rightarrow \sigma(Var \neq val)$$

This ensures that we never explore the symmetric state to the one that has just been excluded. Our method also posts static symmetry breaking dynamically during search. However, the two methods differ along three important dimensions:

When symmetry breaking constraints are posted: SBDS posts symmetry breaking constraints when backtracking and exploring the second branch of the search tree; here, on the other hand, we can post symmetry breaking constraints down either branch;

What symmetry breaking constraints are posted: SBDS posts symmetries of the current nogood; here, on the other hand, we can post *any* type of symmetry breaking constraint. For instance, as we see in the next section, we can dynamically post complex symmetry breaking constraints;

Whether symmetry breaking ever conflicts with branching: If the branching heuristic goes directly to a solution, both SBDS and our method permit this. SBDS has the additional property that it never conflicts with the branching heuristic at any point in search. Our method may conflict with the branching heuristic later on in

search as constraint propagation on the posted symmetry breaking prunes values that branching might have taken.

In fact, our method can be viewed as a generalization of SBDS to work with symmetry breaking constraints other than symmetric nogoods. We can identify pathological examples where the methods perform differently. For example, on a simple model of the pigeonhole problem given in (Walsh 2007), SBDS takes an exponential amount of time to solve the problem irrespective of the branching heuristic whilst our method using the symmetry breaking constraints described in the next section will take just polynomial time.

Comparison with other methods

Jefferson *et al,* have proposed GAPLex, a hybrid method that also combines together static and dynamic based symmetry breaking (Jefferson et al. 2006). However, unlike our method, GAPLex is limited to dynamically posting lex ordering constraints, and to searching with a fixed variable ordering. As a consequence, GAPLex performs poorly when there are large numbers of symmetries. In addition, GAPLex is unable to profit from effective dynamic variable ordering heuristics.

Puget has also proposed "Dynamic Lex", a hybrid method that dynamically posts static symmetry breaking constraints during search (Puget 2003). Puget's method adds lex ordering symmetry breaking constraints dynamically during search that are compatible with the current partial assignment. There are several differences with the hybrid method proposed here. The first is that Puget's method needs to compute the stabilizers of the current partial assignment. This requires a potentially expensive graph isomorphism problem to be solved at each node of the search tree. By comparison, our method requires simple entailment tests. The second difference is that Puget's method needs to limit the symmetry breaking constraints posted in some way as all symmetries are compatible with the root node of the search tree. A third difference is that Puget's method is limited to posting lex ordering constraints.

Interchangeable variables and values

To illustrate how our method can post any type of symmetry breaking constraint, we consider a common type of symmetry where variables and values partition into interchangeable sets (Sellmann and Hentenryck 2005; Flener et al. 2006). Suppose that the n variables partition into a disjoint sets and variables within each set are interchangeable. Similarly, suppose that the m values partition into b disjoint sets and values within each set are interchangeable. We will order variable indices so that $X_{p(i)}$ to $X_{p(i+1)-1}$ is the ith partition of variables for $1 \leq i \leq a$, and value indices so that $d_{q(j)}$ to $d_{q(j+1)-1}$ is the jth partition of values for $1 \leq j \leq b$.

Flener *et al.* (Flener et al. 2006) proved that we can eliminate the exponential number of symmetries due to such interchangeability with the following constraints:

$$X_{p(i)} \leq \ldots \leq X_{p(i+1)-1}$$

$$\textsc{Gcc}([X_{p(i)}, \ldots, X_{p(i+1)-1}], [d_1, \ldots, d_m], [O_1^i, \ldots, O_m^i])$$

$$(O^1_{q(j)}, \ldots, O^a_{q(j)}) \geq_{\text{lex}} \ldots \geq_{\text{lex}} (O^1_{q(j+1)-1}, \ldots, O^a_{q(j+1)-1})$$

Where $i \in [1, a]$ and $j \in [1, b]$, and Gcc counts the number of occurrences of the values in each equivalence class of variables. That is, $O^i_j = |\{k | X_k = d_j, p(i) \leq k < p(i+1)\}|$. The *signature* of d_k is $(O^1_k, \ldots, O^a_k)$, the number of occurrences of d_k in each variable partition. The signature is invariant to the permutation of variables within each equivalence class. By ordering variables within each equivalence class, we prevent permutation of interchangeable variables. Similarly, by ordering the signatures, we prevent permutation of interchangeable values.

We will dynamically post a symmetry of these symmetry breaking constraints during search. We consider symmetries that act along two degrees of freedom: the order of interchangeable variables within a variable partition, and the order of the signatures of interchangeable values within a value partition. Let σ be some permutation of the indices of interchangeable variables. Then we can break the symmetry of variable interchangeability with the following symmetry of the variable ordering constraints:

$$X_{\sigma(p(i))} \leq \ldots \leq X_{\sigma(p(i+1)-1)}$$

We shall choose σ dynamically during search according to the choices of the branching heuristic. Based on the least commitment rule, we post $X_j \leq X_k$ whenever X_j and X_k are interchangeable and $X_j < X_k$ is entailed. Similarly let θ be some permutation of the indices of interchangeable values. Then we can break the symmetry of value interchangeability with this symmetry of the signature ordering constraints:

$$(O^1_{\theta(q(j))}, \ldots, O^a_{\theta(q(j))}) \geq_{\text{lex}} \ldots \geq_{\text{lex}} (O^1_{\theta(q(j+1)-1)}, \ldots)$$

We shall choose θ dynamically during search according to the choices of the branching heuristic. Again, we will post such lex ordering constraints based on the least commitment rule.

Special case: only interchangeable variables. Suppose variables partition into interchangeable sets but the values do not. Then this method simplifies to the symmetry breaking rule:

If $X_i < X_j$ is entailed, and X_i and X_j are interchangeable then post $X_i \leq X_j$.

That is, we break symmetry by ordering interchangeable variables whenever branching strictly orders them.

Special case: only interchangeable values. Suppose the values partition into interchangeable sets but the variables do not. Then this method simplifies to a dynamic form of value precedence (Law and Lee 2004) in which we order the first occurrence of interchangeable values. It corresponds to the symmetry breaking rule:

If d_j first occurs before d_k, and the two values are interchangeable then post constraints to ensure that d_j always first occurs before d_k.

Experiments

We dynamically and statically posted the symmetry breaking constraints of Flener *et al.* (Flener et al. 2006) in Gecode

Instance	Static symmetry breaking						Our dynamic method			SBDS		
	Lex			Antilex			Lex/Antilex			Lex/Antilex		
	o	t	b	o	t	b	o	t	b	o	t	b
1	9	**0.02**	**13**	9	316.81	5973 K	9	0.13	105	9	305.25	5973 K
2	17	26.02	99 K	17	0.01 *	0 *	17	**0.08**	**43**	17	0 *	0 *
3	14	11.2	46 K	14	0 *	0 *	14	**6.94**	**17 K**	14	0 *	0 *
4	11	388.77	1838 K	11	0.01 *	0 *	11	**17.82**	**43 K**	11	0 *	0 *
5	-	-	-	12	69.63 *	2555 K *	12	**4.37**	**8197**	12	69.49 *	2555 K *
6	15	**37.41**	**113 K**	15	0 *	0 *	15	70.13	148 K	15	0 *	0 *
7	20	**72.46**	**275 K**	20	0 *	0 *	20	163.3	392 K	20	0 *	0 *
8	10	41.91	223 K	10	0 *	0 *	10	**0.06**	**17**	10	0 *	0 *
9	13	**1.32**	**7012**	13	0 *	0 *	13	2.76	7039	13	0 *	0 *
10	-	-	-	14	0 *	0 *	14	**1.02**	**2117**	14	0 *	0 *
11	15	36.84	56 K	15	0 *	0 *	15	**0.05**	**9**	15	0 *	0 *
12	10	**27.59**	**160 K**	10	0.01 *	0 *	10	62.75	159 K	10	0 *	0 *
13	13	**287.15**	**1293 K**	13	0 *	0 *	13	0.05 *	0 *	13	0 *	0 *
14	7	**0.01**	**11**	7	5.86	108 K	7	0.13	89	7	5.5	108 K
15	7	**52.79**	**427 K**	7	0 *	0 *	7	122.79	428 K	7	0 *	0 *

Table 1: Static versus dynamic symmetry breaking constraints in graph coloring using smallest domain variable ordering and lexicographic or inverse lexicographic value ordering. The table gives the value of the **o**ptimal solution found, as well as the **t**ime and the **b**ranches needed to find it and prove optimality. "-" indicates that no solution was found within the timeout. "*" indicates that optimality was not proven and the result is for just finding the optimal solution.

Instance	Static symmetry breaking						Our dynamic method			SBDS		
	Lex			Antilex			Lex/Antilex			Lex/Antilex		
	o	t	b	o	t	b	o	t	b	o	t	b
1	**2894**	**2.1**	**2128**	1765	553.72 *	828 K *	**2894**	2.54	2184	1765	532.58 *	828 K *
2	**2245**	**1.87**	**1836**	2194	428.51 *	849 K *	**2245**	3.38	3585	2194	409.25 *	849 K *
3	**2639**	**19.75**	**15 K**	1685	175.07 *	239 K *	**2639**	24.2	16 K	1685	159.91 *	239 K *
4	**2962**	**2.69**	**2008**	2610	102.83 *	149 K *	**2962**	3.34	2136	2610	101.99 *	149 K *
5	**3634**	**3.67**	**3797**	3286	107.2 *	233 K *	**3634**	4.69	3930	3286	102.32 *	233 K *
6	**3358**	**2.22**	**2094**	3322	82.7 *	126 K *	**3358**	2.71	2196	3322	75.04 *	126 K *
7	**3262**	**1.39**	**1102**	3186	30.53 *	66 K *	**3262**	2.47	2315	3224	562.71 *	1078 K *
8	**3288**	**3.61**	**2606**	1658	492.53 *	734 K *	**3288**	4.33	2808	1658	450.71 *	734 K *
9	**3434**	**16.35**	**14 K**	2335	102.6 *	174 K *	**3434**	20.22	**14 K**	2335	102.35 *	174 K *
10	**2847**	**4.44**	**4888**	2649	151.09 *	329 K *	**2847**	5.75	5030	2649	152.73 *	329 K *

Table 2: Static versus dynamic symmetry breaking in the concert hall scheduling problem using smallest domain variable ordering and lexicographic or inverse lexicographic value ordering within each value partition. Same abbreviations are used as in Table 1.

2.2.0 and evaluated them on the same two benchmark domains used in a previous study of symmetry breaking for interchangeable variables and values (Law et al. 2007). Experiments were run on an 2-way Intel Xeon with 6MB of cache and 4 cores in each processor running at 2GHz. All instances were terminated after 10 minutes. We used smallest domain as a variable ordering heuristic in each experiment. As a value ordering heuristic, we used a lexicographical and anti-lexicographical order of values.

Our experiments are designed to test two hypotheses. The first hypothesis is that our method of dynamically posting static symmetry breaking constraints is less sensitive to the branching heuristic than purely static methods. Our choice of value ordering heuristics demonstrate the impact on search of inverting the value ordering. Due to the interchangeability of values, other value ordering heuristics will exhibit similar behaviour. The second hypothesis is that our method of dynamically posting static symmetry breaking constraints explores a smaller search tree than dynamic methods like SBDS due to propagation of the posted symmetry breaking constraints.

We limit our comparison of dynamic methods to comparison against SBDS. Whilst there is a specialized dynamic symmetry breaking method for interchangeable variables and values, experiments in (Heller et al. 2008) show that this is several orders of magnitude slower than static methods. In addition, dominance detection methods like SBDD are shown to be three orders of magnitude slower than static methods in (Heller et al. 2008). Finally, we used SBDS to break just generators of the symmetry group as breaking the full symmetry group quickly ran out of memory.

The first set of experiments uses random graph coloring problems generated in the same way as the previous experimental study in (Law et al. 2007). There is a variable for each vertex and not-equals constraints between variables corresponding to connected vertices. All values in this model are interchangeable. In addition, we introduce variable symmetry by partitioning variables into interchange-

able sets of size at most 8. We randomly connect the vertices within each partition with either a complete graph or an empty graph, and choose each option with equal probability. Similarly, between any two partitions there is equal probability that the partitions are completely connected or independent. Results for graphs with 40 vertices are shown in Table 1.

The second set of experiments uses a more structured benchmark which is again taken from a previous experimental study (Law et al. 2007). In the concert hall scheduling problem, we have n applications to use one of m identical concert halls. Each application has a start and end time as well as an offer for the hall. We accept applications so that their intervals do not overlap and the profit (the sum of the offers of accepted applications) is maximized. We randomly generate instances so that applications are split into partitions of size at most 8 and within each partition all applications have the same start and end time and offer. Our model assigns $X_i = j$ if the i^{th} application is accepted and placed in hall j, and $X_i = m + 1$ if it is rejected. Variables corresponding to applications in the same partition are interchangeable. Values divide into two partitions: the values 1 to m are interchangeable, while the value $m + 1$ is in a separate partition. Results for instances with 40 applications and 10 halls are shown in Table 2.

The results support both our hypotheses. Our dynamic method is less sensitive to the branching heuristic than the static method. Using our method, the difference in times between the two value ordering heuristics was less than the timing error. Note this also holds for SBDS. In contrast, the static method can exhibit very poor performance when used in conjunction with the anti-lexicographical value ordering, because the branching heuristic guides the search towards the lexicographically greatest solution, but the static symmetry breaking constraints eliminate all but the lexicographically least solution from each equivalence class. Our second hypothesis, that our method explores a smaller search tree than SBDS is also confirmed. SBDS was unable to prove optimality in all but one instance. In addition, on many of the harder graph coloring instances, our method significantly outperforms the static method with either ordering. It can usually quickly find the optimal solution and then use the set of symmetry breaking constraints induced by this solution to prove optimality much faster. On the concert hall scheduling problem, our method is slightly slower than the best static method, mostly due to the time that it takes to select a symmetry of the symmetry breaking constraints, but it is much better than SBDS which takes much longer to find lower quality solutions.

Conclusions

It is important to factor symmetry out of combinatorial search. We have described a hybrid method for posting constraints during search which eliminate symmetric solutions. The method is based on the observations that any symmetry of a set of symmetry breaking constraints itself breaks symmetry, and that each symmetry picks out a different solution in each symmetry class. We post symmetry breaking constraints during search as branching decisions force the choice of a particular symmetry of the symmetry breaking constraints. We proved that the method is correct in general. That is, it will only eliminate symmetric solutions. We also identified common conditions under which it eliminates all symmetric solutions. We illustrated the method with a common type of symmetry where variables and values are interchangeable. Our approach inherits good properties of both dynamic and static symmetry breaking methods: we have fast and efficient propagation on the posted constraints, yet we do not conflict with the branching heuristic. Whilst the method has been proposed in the context of backtracking search in constraint programming, many of the ideas would appear to apply to combinatorial search in general.

Acknowledgements

This research is funded by the Department of Broadband, Communications and the Digial Economy, and the ARC through Backing Australias Ability and the ICT Centre of Excellence program.

References

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

Backofen, R., and Will, S. 1999. Excluding symmetries in constraint-based search. In Jaffar, J., ed., *Proceedings of the 5th International Conference on Principles and Practice of Constraint Programming*, number 1713 in Lecture Notes in Computer Science, 73–87. Springer-Verlag.

Backofen, R., and Will, S. 2002. Excluding symmetries in constraint-based search. *Constraints* 7(3-4):333–349.

Cohen, D.; Jeavons, P.; Jefferson, C.; Petrie, K.; and Smith, B. 2006. Symmetry definitions for constraint satisfaction problems. *Constraints* 11(2–3):115–137.

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

Fahle, T.; Schamberger, S.; and Sellmann, M. 2001. Symmetry breaking. In Walsh, T., ed., *Proceedings of 7th International Conference on Principles and Practice of Constraint Programming (CP2001)*, 93–107. Springer.

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 International Conference on Principles and Practices of Constraint Programming (CP-2002)*. Springer.

Flener, P.; Pearson, J.; Sellmann, M.; and Hentenryck, P. V. 2006. Static and dynamic structural symmetry breaking. In *Proceedings of 12th International Conference on Principles and Practice of Constraint Programming (CP2006)*. Springer.

Gent, I., and Smith, B. 2000. Symmetry breaking in constraint programming. In Horn, W., ed., *Proceedings of ECAI-2000*, 599–603. IOS Press.

Heller, D.; Panda, A.; Sellmann, M.; and Yip, J. 2008. Model restarts for structural symmetry breaking. In *14th International Conference on the Principles and Practice of Constraint Programming*, 539–544.

Jefferson, C.; Kelsey, T.; Linton, S.; and Petrie, K. 2006. Gaplex: Generalised static symmetry breaking. In *Proceedings of 6th International Workshop on Symmetry in Constraint Satisfaction Problems (SymCon-06), held alongside CP-06*.

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

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 International Conference on Principles and Practices of Constraint Programming (CP-2007)*. Springer-Verlag.

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

Puget, J.-F. 2003. Symmetry breaking using stabilizers. In Rossi, F., ed., *Proceedings of 9th International Conference on Principles and Practice of Constraint Programming (CP2003)*. Springer.

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

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

Shlyakhter, I. 2001. Generating effective symmetry-breaking predicates for search problems. In *Proceedings of LICS workshop on Theory and Applications of Satisfiability Testing (SAT 2001)*.

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

Walsh, T. 2007. Breaking value symmetry. In *13th International Conference on Principles and Practices of Constraint Programming (CP-2007)*. Springer-Verlag.