

Dynamic Posting of Static Symmetry Breaking Constraints*

George Katsirelos

NICTA

Sydney, Australia

george.katsirelos@nicta.com.au

Nina Narodytska

NICTA and UNSW

Sydney, Australia

ninan@cse.unsw.edu.au

Toby Walsh

NICTA and UNSW

Sydney, Australia

toby.walsh@nicta.com.au

Abstract

We have proposed a method for dynamically posting symmetry breaking constraints to eliminate value interchangeability [1]. We now extend this method to work with *any* type of symmetry breaking constraint for *any* type of symmetry. We prove that this method is correct in general. That is, it only eliminates symmetric solutions. We also identify some simple conditions when it eliminates all symmetric solutions. We illustrate the method with a common type of symmetry where both variables and values partition into interchangeable sets, and a polynomial number of symmetry breaking constraints breaks an exponential number of symmetries. This hybrid approach inherits good properties of both dynamic and static symmetry breaking methods: we can have fast and efficient propagation on the posted symmetry breaking constraints, yet we do not conflict with the branching heuristic.

1 Introduction

Symmetry is an important aspect of constraint programming. To deal with symmetry we can *statically* add constraints which eliminate some or all of the symmetric solutions (e.g. [2; 3]), or we can modify the search procedure so that it *dynamically* avoids symmetric solutions (e.g. [4; 5; 6]). Each method has its 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 method, on the other hand, typically do not conflict with the branching heuristic, but can perform poorly when there are many symmetries. In addition, dynamic methods do not prune other variables. With static methods, propagation between problem constraints and symmetry breaking constraints can prune search greatly. In Section 9, we describe related work in more detail.

*NICTA is funded by the Australian Government’s Department of Broadband, Communications, and the Digital Economy and the Australian Research Council.

Methods have been developed that attempt to achieve more pruning while maintaining the advantages of dynamic methods (see, for instance, [7]). In [1], we have proposed such a method for breaking the symmetry introduced by interchangeable values. This method posts static symmetry breaking constraints dynamically during search. The posted constraints are chosen to be consistent with the initial choices of the branching heuristic. We now extend this hybrid method in a number of important directions both practical and theoretical. First, we broaden the method beyond the simple symmetry of interchangeable values studied in [1]. We show that the method can be used to post dynamically *any* type of symmetry breaking constraints for *any* type of symmetry. In particular, it is not restricted to posting lex leader symmetry breaking constraints. Second, we show that the “entailment” rule proposed in [1] is too eager since it may post symmetry breaking constraints before branching decisions have forced their choice. Third, we give a new “least commitment” rule which does not have such problems. We prove that this rule is correct and only eliminates symmetric solutions. Fourth, we show how to apply this method to a common type of symmetry where both variables and values partition into interchangeable sets. Our experiments demonstrate that, as intended, this hybrid method makes symmetry breaking largely insensitive to the branching heuristic.

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. A solution is an assignment to the variables satisfying the constraints. We write $sol(C)$ for the solution to a set of constraints C . A constraint is *domain consistent* iff for each variable, every value in its domain can be extended to an assignment that satisfies the constraint. A constraint is *entailed* when any assignment of values left in the domain of the variables is a satisfying assignment. A constraint is *dis-entailed* when its negation is entailed.

Constraint satisfaction problems can contain symmetry. We will consider two special types of symmetry. A *variable symmetry* is a permutation of the variables that preserves solutions. Formally, it is a bijection σ on the indices of variables such that if $X_1 = d_1, \dots, X_n = d_n$ is a solution then $X_{\sigma(1)} = d_1, \dots, X_{\sigma(n)} = d_n$ is also. A *value*

symmetry, on the other hand, is a permutation of the values that preserves solutions. Formally, it is a bijection θ on the values such that if $X_1 = d_1, \dots, X_n = d_n$ is a solution then $X_1 = \theta(d_1), \dots, X_n = \theta(d_n)$ is also. Finally, a *variable/value symmetry* acts on both variables and values. Our methods also work with such general types of symmetry.

Running Example: *The all interval series problem from musical composition (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 can model this as a constraint satisfaction problem with $X_i = j$ iff the i th number in the series is j . One solution for $n = 11$ is:*

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

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

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

Second, there is a value symmetry θ_{inv} that inverts values. If we subtract all values in (1) from 10, we generate a second (but symmetric) solution:

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

Third, we can do both. By reversing and inverting (1), we generate a third (but symmetric) solution:

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

3 Symmetry breaking constraints

One method to deal with symmetry is to add constraints which eliminate some but not all of the symmetric solutions [2]. Two important properties of a set 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 in each symmetry class.

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

$$X_1 < X_{11} \quad (5)$$

For example, this eliminates solution (2) as it is the reversal of (1). To eliminate the value symmetry θ_{inv} , we post:

$$X_1 \leq 5, \quad X_1 = 5 \Rightarrow X_2 < 5 \quad (6)$$

For example, this eliminates solution (3) as it is the inversion of (1). Finally, to eliminate the third symmetry $\theta_{inv} \circ \sigma_{rev}$ where we both reverse and invert the solution, we post:

$$\langle X_1, \dots, X_6 \rangle \leq_{lex} \langle 10 - X_{11}, \dots, 10 - X_6 \rangle \quad (7)$$

For example, this eliminates solution (1) as it is the reversal and inversion of (4). Note that of the four symmetric solutions given earlier, only (4) satisfies all these symmetry breaking constraints. The other three solutions are eliminated.

We will dynamically post a symmetry of a set of symmetry breaking constraints during search. This requires us to consider the action of a symmetry on a symmetry breaking constraint. We have defined symmetries as acting on assignments, mapping solutions to solutions. However, we can lift symmetries to act on constraints. The action of a variable symmetry on a constraint simply changes the variables on which the constraint acts. More precisely, σ applied to $C(X_j, \dots, X_k)$ gives $C(X_{\sigma(j)}, \dots, X_{\sigma(k)})$. The action of a value symmetry is also easy to compute. θ applied to $C(X_j, \dots, X_k)$ gives $C(\theta(X_j), \dots, \theta(X_k))$. In many cases, as we shall see, we can compute this action symbolically. Alternatively, we can use *element* constraints:

$$C(Y_j, \dots, Y_k), \text{element}(X_j, [\theta(1), \dots, \theta(m)], Y_j), \dots$$

Running Example: *To illustrate how we can break symmetry with the symmetry of a set of symmetry breaking constraints, we consider symmetries of (5), (6) and (7). If we apply σ_{rev} to (5), we get a symmetric 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 (6), we get a symmetric ordering constraint that again breaks the inversion symmetry:

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

Finally, if we apply σ_{rev} to (7), we get a constraint that again breaks the combined reversal and inversion symmetry:

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

Note that of the four symmetric solutions given earlier, only (3) satisfies the reversal symmetry of (5), (6) and (7). 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 (5), 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 (6), we get a constraint that again breaks the inversion symmetry:

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

This simplifies to: $X_{11} \geq 5$, $X_{11} = 5 \Rightarrow X_{10} > 5$. Finally, if we apply $\theta_{inv} \circ \sigma_{rev}$ to (7), we get a constraint that again breaks the combined reversal and inversion symmetry:

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

Note that of the four symmetric solutions given earlier, only (1) satisfies the combined reversal and inversion symmetry of (5), (6) and (7).

The running example illustrates how 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, any symmetry of these constraints is also complete.

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 $s \in \text{sol}(C \cup S)$. Then $s \in \text{sol}(C)$ and $s \in \text{sol}(S)$. We exploit the fact that σ^{-1} , the inverse of σ is itself a symmetry of C . Hence $\sigma^{-1}(s) \in \text{sol}(C)$. Since $s \in \text{sol}(S)$, it follows that $\sigma^{-1}(s) \in \text{sol}(\sigma(S))$. Thus, $\sigma^{-1}(s) \in \text{sol}(C \cup \sigma(S))$. 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 $s \in \text{sol}(C \cup \sigma(S))$. By a similar argument to soundness, $\sigma(s) \in \text{sol}(C \cup S)$. 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 .

One direct application of this result, following [8], would be to use restarts and a random symmetry of a set of symmetry breaking constraints.

4 Entailment rule

We have shown that we can post a symmetry of a set of symmetry breaking constraints, and that this preserves the soundness and completeness of the symmetry breaking constraints. We consider now how to post such a set of symmetry breaking constraints incrementally during search. 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. In [1], we proposed the following entailment rule for incrementally posting symmetry breaking constraints:

Definition 1 (Entailment rule) *Given a set of symmetry breaking constraints S , if during backtracking search a symmetry $\sigma(s)$ of a constraint $s \in S$ is entailed and σ is consistent with the previously posted symmetry breaking constraints T , then the entailment rule posts the entailed constraint $\sigma(s)$ so it holds during search from then onwards as if it was posted at the root of the search tree.*

We first show that this rule is sound and complete. A symmetry breaking method is sound iff it will find at least one solution in each symmetry class, and complete iff it will find at most one solution in each symmetry class.

Theorem 2 *If S is a sound and complete set of symmetry breaking constraints then the entailment rule using S is also a sound and complete symmetry breaking method.*

Proof: *Soundness.* By Theorem 1, the symmetry of a sound set of symmetry breaking constraints is itself sound. As the entailment rule only permits entailed constraints of a particular symmetry to be posted, it is sound by monotonicity.

Completeness. Let sol be a solution found using the entailment rule. This solution must be consistent with at least

one symmetry $\sigma(S)$ of S . All constraints $s \in \sigma(S)$ are fully instantiated and satisfied, thus they are entailed and the rule will post them. At this point, the rule has posted a complete set of symmetry breaking constraints, thus it will not visit two solutions from the same equivalence class. \diamond

The entailment rule is too eager, as it may commit to a symmetry of the symmetry breaking constraints before the branching heuristic has forced that decision. Thus, like static methods, it may conflict with the branching heuristic. In addition, the rule is not completely specified.

Running Example: *Consider the first part of (6), $X_1 \leq 5$ and the symmetry of this under θ_{inv} , $X_1 \geq 5$. If we assign $X_1 = 5$ then both $X_1 \leq 5$ and $X_1 \geq 5$ are entailed and may be posted by the entailment rule. As soon as one is posted, the other is inconsistent and will not be posted. However, the entailment rule does not specify which should be posted.*

5 Least commitment rule

We now propose a more restricted rule which waits until branching decisions force the choice of symmetry. 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 *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:

Definition 2 (Least commitment rule) *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 the least commitment rule posts the entailed constraint so it holds during search from then onwards as if it was posted at the root of the search tree.*

Note that the least commitment rule is a special case of the entailment rule. It prevents the entailment rule from posting an entailed symmetry breaking constraint if this eliminates a symmetry that is still consistent. We first show that this new rule is sound and complete.

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

Proof: *Soundness.* Since the entailment rule is sound and the least commitment rule is more restricted, the least commitment rule is also sound.

Completeness. Let sol be a solution visited by the least commitment rule. Let Σ_{sol} be the set of symmetries that are consistent with sol . For each symmetry $\sigma \in \Sigma_{\text{sol}}$, all constraints in $\sigma(S)$ are entailed. The least commitment rule only posts those constraints in $\sigma(S)$ that break symmetries in $\Sigma - \Sigma_{\text{sol}}$. However, all the symmetries in Σ_{sol} leave sol unchanged, otherwise there would exist a constraint in $\sigma(S)$ that is not entailed. Therefore, the least commitment rule posts

constraints that break the symmetries that generate other solutions from the equivalence class of sol , so it will not visit those other solutions. \diamond

We also propose a further restriction of the least commitment rule that only posts constraints when a solution is found.

Definition 3 (Solution rule) *Given a set of symmetry breaking constraints S , if during backtracking search a solution sol is found, a symmetry $\sigma(s)$ of a constraint $s \in S$ is entailed, σ is consistent with the previously posted symmetry breaking constraints T , and all symmetries σ' eliminated by this entailed constraint are inconsistent with the current state then the solution rule posts the entailed constraint $\sigma(s)$ so it holds during search from then onwards as if it was posted at the root of the search tree.*

Theorem 4 *Given a set of symmetries Σ of C , if S is a sound and complete set of symmetry breaking constraints for Σ then the solution rule using S is a sound and complete symmetry breaking method.*

Proof: Analogous to the proof of theorem 3. \diamond

Even though these rules are sound and complete, they take some time before they post a complete set of symmetry breaking constraints. As a result, they may visit symmetric unsatisfiable branches that would not be visited if we had statically posted the same symmetry breaking constraints. We identify a simple property of a symmetry group that guarantees that the least commitment and solution rules post a complete set of symmetry breaking constraints as soon as a solution is found. A symmetry group Σ of C is *strict* iff every non-identity symmetry in Σ maps a solution of C onto a different one. With a strict set of symmetry breaking constraints, each solution within an equivalence class is associated with a different symmetry.

Running Example: *Consider the three non-identity symmetries in Σ . σ_{rev} swaps X_1 and X_{11} . As $X_1 < X_{11}$, σ_{rev} maps any solution of C onto a different solution. θ_{inv} maps X_i onto $10 - X_i$. There are two cases to consider. If $X_1 < 5$, then $X_1 \neq \theta_{inv}(X_1)$. If $X_1 = 5$, then $X_2 < 5$ and hence $X_2 \neq \theta_{inv}(X_2)$. Thus, in both cases, θ_{inv} maps any solution of C onto a different solution. Finally, $\theta_{inv} \circ \sigma_{rev}$ maps X_i onto $10 - X_{12-i}$. Suppose this maps a solution of C onto itself. Now the difference between X_{11} and X_{10} is $|(10 - X_1) - (10 - X_2)| = |X_1 - X_2|$, the difference between X_1 and X_2 . But no two differences are identical. This is a contradiction. Hence, the action of $\theta_{inv} \circ \sigma_{rev}$ must give a different solution. To conclude, each of the three non-identity symmetries maps a solution of C onto a different solution.*

Not all symmetry groups are strict. In fact, a symmetry group that is strict for one problem might not be for another.

Running Example: *Consider a relaxation of the all interval series problem in which differences can be repeated but at most twice. The symmetries of this problem are those of the original all interval series problem. However they are not strict. Consider the following solution:*

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

The action of $\theta_{inv} \circ \sigma_{rev}$ on this solution leaves the assignment unchanged.

We now prove that if the symmetries of a problem C are strict, these rules post a complete set of symmetry breaking constraints as soon as the first solution is found.

Theorem 5 *Given a set of symmetries Σ of C , if Σ is strict with respect to C , then the least entailment rule and the solution rule post a complete set of symmetry breaking constraints as soon as the first solution is found.*

Proof: Since the least entailment rule is more eager than the solution rule, we only need to show this for the solution rule.

Let sol be the first solution found. As the symmetries are strict with respect to C , there exists exactly one $\sigma \in \Sigma$ such that $\sigma(S)$ is consistent with s . Therefore all constraints of $\sigma(S)$ are entailed. In addition, every other symmetry σ' maps sol to a different solution, thus is inconsistent with sol . Therefore, the solution rule posts all of $\sigma(S)$, which is a complete set of symmetry breaking constraints. \diamond

Finally, we observe that with certain symmetry breaking constraints, the least commitment rule is equivalent to the entailment rule. This was the case with the symmetry breaking constraints used to deal with value interchangeability in [1]. A set of symmetry breaking constraints S for Σ is *regular* iff S is sound and complete for Σ , and for each $c \in S$ and $\sigma, \sigma' \in \Sigma$, either $\sigma(c)$ is entailed or dis-entailed by $\sigma'(S)$.

Running Example: *Let S be the union of the constraints given in (5), (6) and (7). Although S is a sound and complete set of symmetry breaking constraints, it is not regular. For instance, take $X_1 \leq 5$. Then $\sigma_{rev}(X_1 \leq 5)$ is $X_{11} \leq 5$. But this is neither entailed nor dis-entailed by $\sigma_{id}(S)$. On the other hand, let S contain a single global constraint that is the conjunction of the constraints given in (5), (6) and (7). Since S is strict, it follows quickly that it is also regular.*

With a regular set of symmetry breaking constraints, the least commitment and entailment rules are identical.

Theorem 6 *Given a set of symmetries Σ , if S is a regular set of symmetry breaking constraints then the least commitment and entailment rules post constraints at the same time.*

Proof: Suppose we have posted the symmetry breaking constraints T and the entailment rule is about to post $\sigma(c)$ where $c \in S$ and $\sigma \in \Sigma$. Then $\sigma(c)$ is entailed, and σ is consistent with T . Consider any $\sigma' \in \Sigma$. There are two cases. In the first, $\sigma(c)$ is entailed by $\sigma'(S)$. Hence, σ' is consistent with $T \cup \{\sigma(c)\}$. In the second, $\sigma(c)$ is dis-entailed by $\sigma'(S)$. Hence, σ' is eliminated by $\sigma(c)$. As $\sigma(c)$ is currently entailed and $\sigma(c)$ is dis-entailed by $\sigma'(S)$, σ' is inconsistent with the current state. Thus, all symmetries eliminated by the entailed constraint are inconsistent with the current state. Thus, the least commitment rule will post $\sigma(c)$. \diamond

6 Implementation

A simple way to implement the least commitment rule is with non-backtrackable variables and reification. For every symmetry σ and every constraint $s \in \sigma(S)$, we create a Boolean variable b_s . We encode the condition that s is entailed and the symmetries that it breaks are inconsistent into a constraint C_s and post the following constraints: $C_s \rightarrow b_s$, $b_s \rightarrow s$. We make the Boolean variable b_s be non-backtrackable. Once it is instantiated, we do not undo its value on backtracking.

Running Example: The symmetries of (5) can be dynamically posted using the following two reified ordering constraints: $B_1 \Leftrightarrow (X_1 < X_{11})$, $B_2 \Leftrightarrow (X_{11} < X_1)$ where B_1 and B_2 are non-backtrackable. Suppose $X_1 < X_{11}$ is entailed. Then B_1 will be assigned to true. As B_1 is non-backtrackable, $X_1 < X_{11}$ will be posted on all future branches as required by the least commitment rule.

As we shall see in the experimental section, such an implementation may be a little inefficient due to redundant constraints. This suggests it will be worth developing specialized algorithms for propagating the dynamically posted symmetry breaking constraints that ignores redundant constraints. Note that the same technique cannot be used to implement the entailment rule (except for the case of regular symmetry breaking constraints.) This is because of the fact that when multiple constraints are entailed that break different symmetries, the rule has to make a choice among those constraints. This cannot be simply encoded using non-backtrackable variables.

7 Interchangeable variables and values

To illustrate this new symmetry breaking method, we consider a common type of symmetry that has been tackled by both purely static and purely dynamic methods [9; 10; 11]. 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 n disjoint sets and values within each set are interchangeable. We order variable indices so that $X_{p(i)}$ to $X_{p(i+1)-1}$ is the i th 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 j th partition of values for $1 \leq j \leq b$.

Flener *et al.* [10] proved that we can eliminate the exponential number of symmetries due to variable and value interchangeability by posting a polynomial number of static symmetry breaking constraints of the form:

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

$$\text{GCC}([X_{p(i)}, \dots, X_{p(i+1)-1}], [d_1, \dots, d_m], [O_1^i, \dots, O_m^i])$$

$$(O_{q(j)}^1, \dots, O_{q(j)}^a) \geq_{\text{lex}} \dots \geq_{\text{lex}} (O_{q(j+1)-1}^1, \dots, O_{q(j+1)-1}^a)$$

GCC is the global cardinality constraint. This counts the number of occurrences of the different values. That is, $O_j^i = |\{k | X_k = d_j, p(i) \leq k < p(i+1)\}|$. The *signature* of d_k is (O_k^1, \dots, O_k^a) , 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 interchangeable variables being permuted. Similarly, by ordering the signatures of values within each equivalence class, we prevent interchangeable values being permuted.

We post a *symmetry* of these symmetry breaking constraints dynamically during search. We consider symmetries that act along two degrees of freedom: the order of variables with a variable partition, and the order of the signatures of interchangeable values with a value partition. To compute O_j^i , the number of occurrences of the value j in the i th partition of variables, we post the following constraints:

$$\text{GCC}([X_{p(i)}, \dots, X_{p(i+1)-1}], [d_1, \dots, d_m], [O_1^i, \dots, O_m^i])$$

Note that these constraints are invariant to permutation of interchangeable variables and values. To break the symmetry between interchangeable variables within each partition of variables, we can order the variables within each partition:

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

Let σ be some permutation of the indices of interchangeable variables. Then, we can also break symmetry with this symmetry of the ordering constraints:

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

We shall choose σ dynamically during search according to the choices of the branching heuristic. To do this, we reify all possible ordering constraints within each equivalence class of variables as follows:

$$\forall i, p(i) \leq j, k \leq p(i+1) - 1, j \neq k. \quad B_{jk} \Leftarrow (X_j > X_k)$$

$$\forall i, p(i) \leq j, k \leq p(i+1) - 1, j \neq k. \quad B_{jk} \Rightarrow (X_j \geq X_k)$$

If the reified variables B_{jk} are not-backtrackable, then we will post the ordering constraint $X_j \leq X_k$ as soon as $X_j < X_k$ is entailed. This is as soon as $X_j \leq X_k$ is entailed and $X_k \leq X_j$ (the other possible symmetry of this ordering constraint) is dis-entailed. In other words, we post $X_j \leq X_k$ according to the *least commitment* rule. We assume we maintain domain consistency on the ordering constraints. This way, we ensure transitivity of the ordering relation defined by B_{jk} .

To break the symmetry between interchangeable values within each partition of values, we can order the signatures:

$$(O_{q(j)}^1, \dots, O_{q(j)}^a) \geq_{\text{lex}} \dots \geq_{\text{lex}} (O_{q(j+1)-1}^1, \dots, O_{q(j+1)-1}^a)$$

Let θ be some permutation of the indices of interchangeable values. We can also break symmetry with this symmetry of the signature ordering constraints:

$$(O_{\theta(q(j))}^1, \dots, O_{\theta(q(j))}^a) \geq_{\text{lex}} \dots \geq_{\text{lex}} (O_{\theta(q(j+1)-1)}^1, \dots, O_{\theta(q(j+1)-1)}^a)$$

We shall choose θ dynamically during search according to the choices of the branching heuristic. We do this by reification similar to the ordering of interchangeable variables:

$$\forall j, q(j) \leq k, l \leq q(j+1) - 1, k \neq l.$$

$$C_{kl} \Leftarrow ((O_k^1, \dots, O_k^a) >_{\text{lex}} (O_l^1, \dots, O_l^a)) \wedge$$

$$C_{kl} \Rightarrow ((O_k^1, \dots, O_k^a) \geq_{\text{lex}} (O_l^1, \dots, O_l^a))$$

The reified variables C_{kl} are not-backtrackable. In this way, we will post lex ordering constraints on signatures according to the least commitment rule. We again need to ensure that the ordering relation defined by C_{kl} is transitive. This is not guaranteed as decomposing a chain of lex ordering constraints hinders propagation. We therefore post the transitivity constraints: $C_{kl} \wedge C_{lm} \Rightarrow C_{km}$ where k, l and m range over the indices of values from the same partition.

Special case: interchangeable variables. Suppose variables partition into interchangeable sets, but values are not interchangeable. Then this method simplifies to:

If $X_i < X_j$ is entailed, and X_i and X_j are interchangeable then we post the symmetry breaking constraint $X_i \leq X_j$.

That is, we break symmetry by ordering interchangeable variables when the branching heuristic strictly orders them.

Special case: interchangeable values. Suppose values partition into interchangeable sets, but variables are not interchangeable. Then this method simplifies to value precedence [12; 13] in which we order the first occurrence of values. It corresponds to the rule:

If d_j first occurs before d_k , and d_j and d_k are interchangeable then we post symmetry breaking constraints to ensure d_j always first occurs before d_k .

8 Experiments

We implemented the symmetry breaking methods described in this paper in Gecode 2.0.1 and evaluated them on graph coloring and concert hall scheduling problems. We compared with static symmetry breaking methods [10]. No performance results have been reported for the dynamic methods proposed in [9]. However, as these methods have an $O(nm^{3.5} + n^2m^2)$ time cost at each node, we conjecture that they are unlikely to be competitive. 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 a timeout of 10 minutes. Our hypothesis was that our dynamic symmetry breaking methods would be less sensitive to the branching heuristic compared to static methods.

In our first set of experiments, we used random graph coloring problems. We model graph coloring as a constraint optimization problem with 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 interchangeable 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 20 instances on graphs with 40 vertices and are shown in table 1.

We compare the static method against the dynamic method using either a lexicographic value ordering or an inverse lexicographic value ordering. The results support our hypothesis as, unlike the static method, the dynamic method is largely unaffected by the choice of value ordering heuristic. Indeed, the dynamic method is in some cases better than the static method with either value ordering heuristic in terms of branches visited, both to find the optimal solution and to prove optimality (e.g., instances 40-7 and 40-11) and is never significantly worse. On the other hand, the dynamic method performs worse in terms of runtime. This suggests it would be worth developing a specialized method to post symmetry breaking constraints dynamically. Our simple method using non-backtrackable variables posts $O(m^2)$ lex ordering constraints even though, after we pick a specific ordering, only $O(m)$ are needed. Therefore, even though our dynamic method often explores a much smaller search tree, it can end up being slower (e.g. instance 40-8). One possibility would be to implement a specialized propagator that only propagates posted constraints that are not redundant.

The second set of experiments is on the concert hall scheduling problem. We have a set of 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 40 random instances with 40 applications for 10 or 14 halls are shown in table 2. Instances that were unsolvable by all algorithms are omitted.

In this set of experiments, the values partition into two sets. An inverse lexicographic ordering would therefore choose values from the same partition in a different order and try the partitions in a different order. This is beyond the scope of our method. Therefore we modified the ordering heuristic to try the values *within each partition* in lexicographic or inverse lexicographic ordering, but try the partitions in the same order always. The results are similar to those for the graph coloring problem. Unlike the graph coloring instances, however, the lexicographic value ordering is not always the best choice for the static method. Regardless of which ordering performs best, the dynamic method with either heuristic tends to explore approximately the same search space as the best performing heuristic for the static method (see, e.g., instances 40-10-3 and 40-10-7). One further difference is that the partitions are smaller, thus the effect on the runtime of posting $O(m^2)$ lex ordering constraints is less pronounced.

9 Related work

Puget proved that symmetric solutions can be eliminated by the addition of static constraints [2]. Crawford *et al.* presented the first general method for constructing static constraints for breaking variable symmetries [3]. Their “lex-leader” method constructs a symmetry breaking constraint for each symmetry which ensures that any solution found is lexicographically less than any of its symmetries. Puget and Walsh independently extended the lex-leader method to value symmetries [14; 15]. The full set of lex-leader constraints can often be simplified. For example, for problems where variables are symmetric and must take all different values, Puget has shown that the lex-leader constraints simplify to a linear number of ordering constraints [16].

Puget has proposed adding lex-leader symmetry breaking constraints dynamically during search that are compatible with the current partial assignment [17]. There are several differences with the method proposed here. First, Puget’s method needs to compute the stabilizers of the current partial assignment. This requires solving a graph isomorphism problem at each node of the search tree. Second, in general Puget’s method needs to limit the symmetry breaking constraints posted as all symmetries are compatible with the root node of the search tree. Third, Puget’s method is limited to posting lex-leader constraints. Perhaps closest to our own

Problem	Static symmetry breaking				Dynamic symmetry breaking			
	Lex		Inverse Lex		Lex		Inverse Lex	
	t (f/p)	b (f/p)	t (f/p)	b (f/p)	t (f/p)	b (f/p)	t (f/p)	b (f/p)
40-1	0.07 / 0.1	66 / 216	- / -	- / -	2.57 / 30.61	0 / 911	2.12 / 23.32	0 / 911
40-2	0.03 / 21.86	0 / 79811	30.04 / 51.96	116333 / 210421	2.61 / -	0 / -	2.58 / -	0 / -
40-3	0.05 / 0.2	19 / 468	242.92 / 243.22	1216243 / 1217334	4.78 / 81.65	0 / 937	3.82 / 62.66	0 / 937
40-4	0.06 / -	62 / -	- / -	- / -	2.72 / -	0 / -	3.19 / -	0 / -
40-5	394.67 / -	1966155 / -	- / -	- / -	2.16 / 10.19	0 / 224	2.36 / 10.15	0 / 224
40-6	0.03 / 0.12	0 / 319	- / -	- / -	2.46 / 11.3	0 / 265	2.4 / 11.32	0 / 265
40-7	0.05 / 22.84	13 / 80531	- / -	- / -	2.33 / 5.36	0 / 26	2.28 / 5.33	0 / 26
40-8	0.06 / 0.41	28 / 849	- / -	- / -	2.55 / 11.4	0 / 183	2.47 / 11.23	0 / 183
40-9	0.03 / -	0 / -	- / -	- / -	3.66 / -	0 / -	4.24 / -	0 / -
40-10	0.02 / 0.16	0 / 430	15.61 / 15.61	43485 / 43488	2.85 / 21.59	0 / 445	2.37 / 18.24	0 / 445
40-11	2.21 / 73.02	5190 / 249224	- / -	- / -	2.9 / 23.11	0 / 455	2.77 / 22.23	0 / 455
40-12	21.6 / 39.38	77515 / 141903	- / -	- / -	3.59 / -	0 / -	4.12 / -	0 / -
40-13	0.03 / -	- / -	- / -	- / -	3.69 / -	0 / -	2.65 / -	0 / -
40-14	0.48 / -	672 / -	- / -	- / -	3.79 / -	0 / -	3.88 / -	0 / -
40-15	0.03 / 0.04	0 / 19	- / -	- / -	3.86 / 7.77	0 / 26	3.06 / 6.13	0 / 26
40-16	- / -	- / -	- / -	- / -	93.2 / -	1489 / -	100.43 / -	1489 / -
40-17	0.04 / -	8 / -	- / -	- / -	3.65 / -	0 / -	4.05 / -	0 / -
40-18	0.03 / 519.7	0 / 1585957	- / -	- / -	2.87 / -	0 / -	3.25 / -	0 / -
40-19	0.05 / 258.6	66 / 968576	- / -	- / -	3.0 / 6.66	0 / 23	2.9 / 6.43	0 / 23
40-20	- / -	- / -	- / -	- / -	2.09 / -	0 / -	2.46 / -	0 / -

Table 1: Static versus dynamic symmetry breaking constraints in graph coloring using lexicographic or inverse lexicographic value ordering. The table gives the **time** and the **branches** needed to find the best solution and to **prove** optimality. “-” indicates that no solution was found (resp. optimality was not proven) within the timeout. Best results are in bold.

Problem <i>n-m-#</i>	Static symmetry breaking				Dynamic symmetry breaking			
	Lex		Inverse Lex		Lex		Inverse Lex	
	t (f/p)	b (f/p)	t (f/p)	b (f/p)	t (f/p)	b (f/p)	t (f/p)	b (f/p)
40-10-1	41.9 / 51.44	132829 / 159570	377.45 / 434.82	1313393 / 1526120	204.36 / 257.76	135139 / 161462	217.32 / 273.96	135139 / 161462
40-10-2	73.9 / 103.9	205756 / 285423	340.54 / 450.56	1285095 / 1674502	267.97 / 380.03	211618 / 291285	267.18 / 377.54	211618 / 291285
40-10-3	253.35 / -	747552 / -	0.71 / -	2563 / -	3.1 / -	2554 / -	3.36 / -	2554 / -
40-10-4	66.59 / 96.55	207807 / 296182	547.71 / -	1831381 / -	426.92 / 583.48	292409 / 383528	436.47 / -	292409 / -
40-10-5	14.08 / 100.07	37432 / 278816	64.48 / 343.32	226343 / 1140594	61.18 / 364.14	51648 / 303952	61.29 / 372.77	51648 / 303952
40-10-6	368.45 / -	1482959 / -	345.03 / -	1838833 / -	286.61 / -	323846 / -	295.91 / -	323846 / -
40-10-7	0.02 / 129.21	15 / 321120	0.12 / 516.76	286 / 1694496	0.41 / 435.65	224 / 325680	0.41 / 437.38	224 / 325680
40-10-8	1.42 / 1.86	4453 / 5718	6.23 / 10.51	22756 / 39644	9.31 / 11.75	5941 / 7206	9.67 / 12.22	5941 / 7206
40-10-9	1.72 / -	7391 / -	5.58 / -	27637 / -	18.08 / -	14574 / -	19.22 / -	14574 / -
40-10-10	12.6 / 27.94	40978 / 93592	51.41 / 96.17	200268 / 373684	209.04 / 365.96	169574 / 288429	208.98 / 364.58	169574 / 288429
40-10-11	78.06 / -	355034 / -	451.55 / -	2441677 / -	442.31 / -	417835 / -	445.86 / -	417835 / -
40-10-12	22.88 / -	123468 / -	12.74 / -	58905 / -	120.21 / -	125111 / -	124.89 / -	125111 / -
40-10-13	184.06 / -	602326 / -	373.92 / -	1511030 / -	326.04 / -	284341 / -	339.07 / -	284341 / -
40-10-14	510.88 / -	2122202 / -	480.71 / -	2137893 / -	0.45 / -	183 / -	0.46 / -	183 / -
40-10-15	409.34 / 451.16	1848031 / 2021059	561.33 / -	2684197 / -	536.94 / -	533773 / -	549.46 / -	533773 / -
40-10-16	392.77 / -	1541315 / -	522.52 / -	2098782 / -	596.23 / -	417565 / -	244.63 / -	170394 / -
40-10-17	21.02 / 390.65	106630 / 1760967	76.27 / -	384129 / -	134.83 / -	110440 / -	147.17 / -	110440 / -
40-10-18	141.13 / 436.24	598697 / 1496853	398.42 / -	1844402 / -	316.79 / -	231729 / -	318.02 / -	231729 / -
40-10-19	10.19 / -	48358 / -	60.17 / -	366986 / -	60.9 / -	51444 / -	67.4 / -	51444 / -
40-10-20	388.83 / 485.01	1457685 / 1820545	6.46 / -	32707 / -	9.62 / -	9166 / -	10.09 / -	9166 / -
40-14-1	360.72 / 376.69	833969 / 873829	411.3 / -	1128661 / -	367.8 / -	142363 / -	372.87 / -	142363 / -
40-14-2	79.86 / 223.03	182978 / 514455	429.93 / -	1361547 / -	417.14 / -	190185 / -	489.23 / -	190185 / -
40-14-3	5.37 / -	14045 / -	76.5 / -	245435 / -	42.68 / -	19023 / -	41.75 / -	19023 / -
40-14-4	517.8 / -	1218491 / -	528.83 / -	1761888 / -	316.59 / -	141753 / -	326.23 / -	141753 / -
40-14-5	71.31 / 419.43	174845 / 1039083	306.61 / -	907303 / -	447.93 / -	243115 / -	490.73 / -	243115 / -
40-14-6	0.04 / -	14 / -	0.14 / -	279 / -	0.82 / -	206 / -	0.96 / -	206 / -
40-14-7	0.07 / 399.57	108 / 897058	2.28 / -	7843 / -	1.71 / -	701 / -	1.92 / -	701 / -
40-14-8	36.5 / 65.85	123069 / 214222	254.97 / 584.35	918030 / 2140530	387.07 / -	136105 / -	415.84 / -	136105 / -
40-14-9	29.87 / -	110691 / -	73.68 / -	307555 / -	339.62 / -	140964 / -	419.13 / -	140964 / -
40-14-12	0.04 / 6.82	31 / 26228	3.54 / 45.28	11765 / 148773	0.91 / 68.89	188 / 25673	1.49 / 125.63	188 / 25673
40-14-13	515.23 / -	1611855 / -	2.22 / -	8906 / -	3.28 / -	1723 / -	5.33 / -	1723 / -
40-14-14	0.04 / -	33 / -	0.12 / -	272 / -	1.02 / -	219 / -	1.15 / -	219 / -
40-14-17	17.71 / -	75776 / -	40.81 / -	186800 / -	209.54 / -	99017 / -	239.91 / -	99017 / -
40-14-20	1.01 / -	3044 / -	118.06 / -	532730 / -	7.1 / -	3881 / -	7.49 / -	3881 / -

Table 2: Static versus dynamic symmetry breaking in the concert hall scheduling problem using lexicographic or inverse lexicographic value ordering within each value partition. The table gives the **time** and the **branches** needed to find the best solution and to **prove** optimality. “-” indicates that no solution was found (resp. optimality was not proven) within the timeout.

work are the dynamic lex leader constraints of [7]. On the one hand, these conflict even less with the branching heuristic as they lex leader constraints use the same variable ordering as in the search tree. On the other hand, whilst there is more propagation than SBDS, there is still less than static methods or our own method. Further, our method works with any type of symmetry breaking constraints (not just lex leader constraints), any symmetry (not just variable and value symmetries), and any type of value ordering heuristic (Puget's method assumes values are tried in order). A third method, GAPLex also combines static and dynamic symmetry breaking methods [18]. Again, there are several significant differences with the method proposed here. For instance, GAPLex uses a fixed ordering on values, within the lex leader constraints. As a second example, GAPLex only achieves as much pruning as SBDS or SBDD, whereas our method eventually achieves as much pruning as the static method.

A number of dynamic methods have been proposed to deal with symmetry. For instance, SBDS posts symmetry breaking constraints dynamically during search [4]. SBDS can be seen as instance of the more general method proposed here. A limitation of SBDS is that it adds a symmetry breaking constraint for each unbroken symmetry. As there can be an exponential number of symmetries, this can be prohibitive. One of our main insights is that we can post *other* types of symmetry breaking constraint dynamically during search. A small number of symmetry breaking constraints may be adequate for special symmetries (e.g. those due to interchangeable variables and values). Another dynamic method for breaking symmetry is SBDD [5]. This checks if a node of the search tree is symmetric to some previously explored node. Finally, Roney-Dougal *et al.* gave a dynamic method to construct a GE-tree, a search tree without value symmetry [6]. A weakness of both these dynamic methods is that they do not propagate their symmetry breaking constraints. Propagation between the problem constraints and the static symmetry breaking constraints can reduce search exponentially [19].

10 Conclusions

We have developed a general method for dynamically posting symmetry breaking constraints during search. The method can post *any* type of symmetry breaking constraints for *any* type of variable, value or variable/value symmetry. The basic idea is very simple. Given a set of symmetry breaking constraints, if a symmetry of one of these constraints is entailed during search, this is consistent with previously posted symmetry breaking constraints, and all symmetries eliminated by posting this constraint are inconsistent with the current state, then we post this symmetry breaking constraint so it holds from then onwards. We proved that this method is correct in general and eliminates all symmetric solutions. We illustrated the method with a common type of symmetry where variables and values partition into interchangeable sets, and a polynomial number of constraints can break an exponential number of symmetries. This hybrid approach inherits good properties of both dynamic and static symmetry breaking methods: we have fast and efficient propagation of the posted constraints, yet we do not conflict with the branching heuristic.

References

- [1] Katsirelos, G., Walsh, T.: Dynamic symmetry breaking constraints. In: ECAI 2008 Workshop on Modeling and Solving Problems with Constraints. (2008)
- [2] Puget, J.F.: On the satisfiability of symmetrical constrained satisfaction problems. In Proc. of ISMIS'93. (1993) 350–361
- [3] Crawford, J., Luks, G., Ginsberg, M., Roy, A.: Symmetry breaking predicates for search problems. In: Proc. of the 5th Int. Conf. on Knowledge Representation and Reasoning, (KR '96). (1996) 148–159
- [4] Gent, I., Smith, B.: Symmetry breaking in constraint programming. In: Proc. of ECAI-2000, (2000) 599–603
- [5] Fahle, T., Schamberger, S., Sellmann, M.: Symmetry breaking. In: Proc. of 7th Int. Conf. on Principles and Practice of Constraint Programming (CP2001), (2001)
- [6] Roney-Dougal, C., Gent, I., Kelsey, T., Linton, S.: Tractable symmetry breaking using restricted search trees. In: Proceedings of ECAI-2004, (2004)
- [7] Puget, J.F.: Dynamic Lex Constraints. In: Proc. of 12th Int. Conf. on Principles and Practice of Constraint Programming (CP2006), (2006)
- [8] Heller, D., Pada, A., Sellmann, M., Yip, J.: Model Restarts for Structural Symmetry Breaking. In: 14th Int. Conf. on Principles and Practices of Constraint Programming (CP-2008), (2008)
- [9] Sellmann, M., Hentenryck, P.V.: Structural symmetry breaking. In: Proc. of 19th IJCAI, (2005)
- [10] Flener, P., Pearson, J., Sellmann, M., Hentenryck, P.V.: Static and dynamic structural symmetry breaking. In: 12th Int. Conf. on Principles and Practice of Constraint Programming (CP2006), (2006)
- [11] Law, Y.C., Lee, J., Walsh, T., Yip, J.: Breaking symmetry of interchangeable variables and values. In: 13th Int. Conf. on Principles and Practices of Constraint Programming (CP-2007), (2007)
- [12] Law, Y., Lee, J.: Global constraints for integer and set value precedence. In: 10th Int. Conf. on Principles and Practice of Constraint Programming (CP2004), (2004)
- [13] Walsh, T.: Symmetry breaking using value precedence. In: Proc. of the 17th ECAI, (2006)
- [14] Puget, J.F.: An efficient way of breaking value symmetries. In: Proc. of the 21st National Conf. on AI, (2006)
- [15] Walsh, T.: General symmetry breaking constraints. In: 12th Int. Conf. on Principles and Practices of Constraint Programming (CP-2006), (2006)
- [16] Puget, J.F.: Breaking symmetries in all different problems. In: Proc. of 19th IJCAI, (2005) 272–277
- [17] Puget, J.F.: Symmetry breaking for matrix models using stabilizers. In: Proc. of 9th Int. Conf. on Principles and Practice of Constraint Programming (CP2003), (2003)
- [18] Jefferson, C., Kelsey, T., Linton, S., Petrie, K.: GAPlex: Generalized static symmetry breaking. In Benhamou, F., Jussien, N., OSullivan, B., eds.: Trends in Constraint Programming. ISTE and Wiley (2007)
- [19] Walsh, T.: Breaking value symmetry. In: 13th Int. Conf. on Principles and Practices of Constraint Programming (CP-2007), (2007)