Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Breaking More Composition Symmetries Using Search Heuristics

Jimmy H. M. Lee and Zichen Zhu*
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
{jlee,zzhu}@cse.cuhk.edu.hk

Abstract

The pruning power of partial symmetry breaking depends on the given subset of symmetries to break as well as the interactions among symmetry breaking constraints. In the context of Partial Symmetry Breaking During Search (ParSBDS), the search order determines the set of symmetry breaking constraints to add and thus also makes an impact on node and solution pruning. In this paper, we give the first formal characterization of the pruning behavior of ParSBDS and its improved variants. Introducing the notion of Dominance-Completeness (DC-ness), we show that ParSBDS and variants eliminate the symmetry group of the given subset of symmetries if the resultant search tree is DC, and give an example scenario. Unfortunately, building a DC tree is not always possible. We propose two search heuristics with the aim of having more nodes dominated and thus also pruned during search. Extensive experimentation demonstrates how the proposed heuristics and their combination can drastically reduce the solution set size, search space and runtime when compared against the state-of-the-art static and dynamic symmetry breaking methods.

Introduction

Symmetries are common in many constraint problems. They can be broken statically (Crawford et al. 1996; Flener et al. 2002) or dynamically (Fahle, Schamberger, and Sellmann 2001; Roney-Dougal et al. 2004; Sellmann and Van Hentenryck 2005). Static methods alter the original problem by adding new constraints to remove symmetric solutions. In contrast, dynamic methods modify the search procedure to exclude exploration of symmetric regions. Practical symmetry breaking methods often trade completeness for efficiency by only breaking a subset of symmetries (Flener et al. 2002; Grayland, Miguel, and Roney-Dougal 2009; Mears et al. 2014). The pruning power of partial symmetry breaking depends on the given subset of symmetries (Jefferson and Petrie 2011) to break as well as the extra composition symmetries that are broken by the interactions among symmetry breaking constraints (Lee and Li 2012).

In the context of Partial Symmetry Breaking During Search (ParSBDS) (McDonald and Smith 2006) that adds conditional symmetry breaking constraints dynamically, the search order determines the set of symmetry breaking constraints to add, and thus affects the extra composition symmetries to be broken and makes an impact on node and solution pruning (McDonald and Smith 2006). We propose a search order under which ParSBDS can break more composition symmetries. In this paper, we give the first formal characterization of the pruning behavior of ParSBDS and its state-of-the-art variants (Lee and Zhu 2014a; 2014b). Relying on a generalization of (symmetry) dominance (Puget 2002; Fahle, Schamberger, and Sellmann 2001), we prove that ParSBDS and variants prune all and only nodes that are dominated. We introduce the notion of Dominance-Completeness (DC-ness), show that ParSBDS and variants eliminate the symmetry group of the given subset of symmetries if the resultant search tree is DC, and give a situation where this happens. Unfortunately, building a DC tree is not always possible. We approximate it using two search heuristics. We propose the "Most Frequent Decision First" and "Most Symmetries Active" search heuristics with the aim of having more nodes dominated (thus also pruned) during search. Unlike popular search heuristics that are either variable based (Bessiere and Régin 1996; Boussemart et al. 2004), value based (Dechter and Pearl 1988; Frost and Dechter 1995) or a simple combination of both, our proposed heuristics select a variable-value pair (decision) to assign. Extensive experimentation demonstrates how our proposed heuristics and their combination can drastically reduce the solution set size, search space and runtime when compared against the state-of-the-art static and dynamic symmetry breaking methods.

Background

A *constraint satisfaction problem* (CSP) P is a tuple (X, D, C) where X is a finite set of variables, D is a finite set of domains such that each $x \in X$ has a domain $D(x)$ and C is a set of constraints, each is a subset of the Cartesian product $D(x_{i_1}) \times \cdots \times D(x_{i_k})$ of the domains of the involved variables (*scope*). We abuse notation to say that $D \subseteq D'$ iff $D(x) \subseteq D'(x)$ for all $x \in X$. A constraint is *generalized arc consistent (GAC)* iff when a variable in the scope of a constraint is assigned any value in its domain, there exist compatible values (called *supports*) in the domains of all the other variables in the scope of the constraint. An *assignment*

*This research has been supported by the grant CUHK413713 from the Research Grants Council of Hong Kong SAR.

assigns a value v to a variable x. A *full assignment* is a set of assignments, one for each variable in X. A *solution* to P is a full assignment satisfying every constraint in C.

We consider search trees with binary branching, in which every non-leaf node has exactly two children. The root node is a CSP P. Suppose a non-leaf node P_1 has x and $v \in D(x)$ as the *branching variable* and *value*. The left and right children of P_1 are $cons(P_1 \cup \{x = v\})$ and $cons(P_1 \cup \{x \neq v\})$ respectively where *cons*() enforces some form of consistency (which could be none) to a CSP. We call $x = v$ the *branching decision* of P_1. Each node P_1 is *associated with a partial assignment* A_1 which is the set of branching decisions collected from the root P to P_1. A *search node* is either the root or a node resulting from branching in the search tree. A *fail node* is a search node in which either (a) a variable has empty domain or (b) a constraint has all variables assigned but is violated.

A *symmetry* is a solution-preserving bijection on assignments. A set of variables X (values V) is *interchangeable* iff any bijection mapping from $X \rightarrow X$ ($V \rightarrow V$) is a variable (value) symmetry. Consider a symmetry group Σ. A *symmetry class* (Flener et al. 2002) is an equivalence class of full assignments, in which every pair of assignments can map into each other with some symmetry $g \in \Sigma$.

A set of symmetries G *generates* a group Σ if every product of any sequence of elements of G is in Σ with the composition operator $\circ$ and every element of Σ can be written as a product of elements in G. We call G a set of *generators* for Σ, which in turn is the *symmetry group* of G.

We assume symmetries are already detected once a CSP is given. We focus on how to break these symmetries efficiently. Given a symmetry group Σ to a CSP, Symmetry Breaking During Search (SBDS) (Gent and Smith 2000) adds conditional constraints for each symmetry upon backtracking. Consider a node P in the search tree with partial assignment A and branching decision $x = v$. After backtracking from the node $cons(P \cup \{x = v\})$, for each symmetry $g \in \Sigma$, SBDS adds the constraint $\neg(A \wedge (x = v))^g$ to the node $cons(P \cup \{x \neq v\})$ meaning that once $A \wedge (x = v)$ has been searched, its symmetric partial assignments $(A \wedge (x = v))^g$ for any $g \in \Sigma$ in this subtree should not be searched at all. A symmetry g is *active* at a node with partial assignment A iff A^g is true.

Partial SBDS (ParSBDS) (McDonald and Smith 2006) is SBDS but deals with only a subset of all symmetries. Recursive SBDS (ReSBDS) (Lee and Zhu 2014a) extends ParSBDS by breaking not only the given symmetries but also some symmetry compositions. Consider the input symmetries G. Upon each backtracking, ReSBDS adds symmetry breaking constraints added by ParSBDS, and also maintains a backtrackable set T of assignments, which is initially empty at the root node. Whenever ReSBDS adds a symmetry breaking constraint, $\neg(A \wedge (x = v))^g$ (where $g \in G$), all assignments in $(A \wedge (x = v))^g$ are recorded into T. Once an assignment $x_i = v_i$ in T is violated at node P with partial assignment B, ReSBDS adds $\neg(B \wedge (x_i = v_i))^h$ for any given symmetry h to the current node. Some symmetry compositions $g \circ h$ are thus broken this way. This violated assignment $x_i = v_i$ is deleted from T. All assignments in $(B \wedge (x_i = v_i))^h$ are recorded into T. ReSBDS recursively initiates constraint propagation and checks violations to add extra constraints until no new constraints are added. A light version, (LReSBDS) (Lee and Zhu 2014b), of ReSBDS is also proposed. Upon each backtracking, LReSBDS adds symmetry breaking constraints added by ParSBDS, and also adds extra symmetry breaking constraints $\neg(B \wedge (x_i = v_i))^h$ for any given symmetry h when $x_i = v_i$ is pruned by a symmetry breaking constraint at a node P with partial assignment B.

Symmetry Breaking During Search

We give a formal analysis to characterize exactly the prunings effected by ParSBDS, ReSBDS and LReSBDS. We generalize the definition of (symmetry) dominance (Puget 2002) in the context of ParSBDS and its variants. A theorem is proved to show that *all and only* nodes which are dominated are pruned by these symmetry breaking methods.

We give some basic definitions. A CSP $P = (X, D, C)$ *entails* an assignment $(x = v)$ if $D(x) = \{v\}$. Consider a subset of symmetries G whose symmetry group is Σ. Suppose τ is a search tree, and P_0 and P_1 are two search nodes. P_1 is *descendant symmetric* to P_0 iff $A_0^g \subseteq E_1$ where $g \in \Sigma$, A_0 is the partial assignment of P_0 and E_1 is the set of assignments entailed by P_1. Furthermore, if in addition $g \in G \subseteq \Sigma$, P_1 is *G-descendant symmetric* to P_0, and P_1 is a (G-)descendant symmetric node of P_0.

P_1 being (G-)descendant symmetric to P_0 means that P_1 is symmetric to P_0 or a descendant node of P_0. Upon backtracking from node P_0, ParSBDS adds constraints to prune all of P_0's G-descendant symmetric nodes. ReSBDS and LReSBDS subsume ParSBDS and add also extra constraints to prune G-descendant symmetric nodes of some pruned nodes. Domain filtering prunes values by an AC3-like (Mackworth 1977) constraint propagation algorithm. During the propagation of c with current domain D, if a value v is pruned from $D(x)$, we say this pruning is *effected* by constraint c and happens *under* domain D.

Consider a search tree τ and a search node $P_0 = cons(X, D_0, C)$ with partial assignment A_0. If v is pruned from $D_0(x)$ during the consistency enforcement of P_0 under domain $D' \subseteq D_0$, $P_1 = (X, D_1, C)$ is a *deleted* node of P_0 where $D_1(x') = D'(x')$ for all $x' \in X - \{x\}$ and $D_1(x) = \{v\}$. The partial assignment of P_1 is $A_0 \cup \{x = v\}$. All deleted nodes at a search node are partitioned into two sequences Γ and Π as explained in the following.

Note that deleted nodes are not search nodes. Depending on the constraint propagation algorithm, once a deleted node is generated at a search node, it is appended to the sequences Γ and Π immediately. Consider the partial search tree in Figure 1. Nodes P_1 and P_3 have deleted nodes. Consider a dynamic symmetry breaking method. Deleted nodes in Π are used to further prune their G-descendant symmetric nodes while deleted nodes in Γ are not.

ParSBDS adds symmetry breaking constraints only upon backtracking and does not utilize deleted nodes to further prune symmetric parts. Π is always empty when using ParSBDS. All deleted nodes are put into Γ. ReSBDS adds extra symmetry breaking constraints if an assignment $x = v$ that

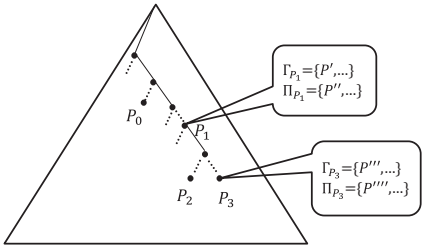

Figure 1: A Partial Search Tree.

has been recorded into the backtrackable set T is pruned. All deleted nodes due to the pruning of such an assignment are thus put into Π when using ReSBDS. All other deleted nodes are put into Γ. LReSBDS adds extra symmetry breaking constraints when a pruning is effected by a symmetry breaking constraint. All deleted nodes due to such prunings are put into Π when using LReSBDS. All other deleted nodes are put into Γ.

In the following, a node of a search tree τ is either a search node or a deleted node. We thus extend the definition of "(G-)descendant symmetric" to cater for also deleted nodes of a search tree.

Deleted nodes are generated due to consistency enforcement of constraints in search nodes. If no consistency is enforced and constraints are used only to check their violations, no values are pruned by constraint filtering algorithms.

Lemma 1. *If no consistency is enforced during search, there are no deleted nodes in the search tree.*

We define simple relations between search nodes and deleted nodes of a search tree, assuming depth first search. Consider a search tree τ and two search nodes P_0 and P_1 in τ. Node P_0 is *to the left of* P_1 iff there exists a search node P in τ such that P_0 is in P's left subtree and P_1 is in P's right subtree. P_0 is *searched before* P_1 iff P_0 is to the left of P_1 or P_0 is an ancestor node of P_1.

In Figure 1, P_0 is to the left of P_1, P_2 and P_3. P_0 and P_1 are searched before P_2 and P_3.

Consider a search tree τ and two nodes P_0 and P_1 of τ. P_0 *precedes* P_1 iff

1. P_0 and P_1 are search nodes. P_0 is to the left of P_1; or
2. P_0 is a search node and P_1 is a deleted node of a search node P. P_0 is to the left of P; or
3. P_0 is in Π of a search node P and P_1 is a search node. P is searched before P_1; or
4. P_0 is in Π of a search node P and P_1 is a deleted node of node P'. P is searched before P'; or
5. P_0 and P_1 are in Π of a search node. P_0 is before P_1 in Π.

Precedence defines a partial order on all nodes of a search tree. In Figure 1, P_0 precedes P_1, all of P_1's deleted nodes, P_2, P_3 and all of P_3's deleted nodes. All nodes in Π_{P_1} precede P_2, P_3 and all of P_3's deleted nodes. Any node in Π_{P_1} or Π_{P_3} precedes all nodes after it in Π_{P_1} or Π_{P_3} respectively.

Consider a subset of symmetries G, a search tree τ and two nodes P_0 and P_1 of τ. P_0 *dominates* P_1 iff P_0 precedes P_1 and P_1 is G-descendant symmetric to P_0.

Note that the notions of precedence and dominance are always associated with a specific symmetry breaking method. We have the following lemma.

Lemma 2. *Consider a subset of symmetries G. If a node P_0 dominates node P_1, all of P_0's ancestors nodes which precede P_1 dominate P_1.*

Proof. Assume A_i and E_i denote the partial assignment and the set of entailed assignments of node P_i. Suppose P_2 is an ancestor node of P_0. Then $A_2 \subseteq A_0$. Since P_1 is G-descendant symmetric to P_0, we have $A_0^g \subseteq E_1$ where $g \in G$. Thus, $A_2^g \subseteq A_0^g \subseteq E_1$. This means P_1 is G-descendant symmetric to P_2. If P_2 further precedes P_1, P_2 dominates P_1. □

A node is *pruned* in a search tree iff it is either a deleted node or a fail node.

Theorem 1. *Consider a subset of symmetries G. ParSBDS/ReSBDS/LReSBDS prunes all and only dominated nodes in the resulting search tree.*

Proof. Upon backtracking from a node P_0 to a node P_1, ParSBDS adds constraints to prune all nodes that are dominated by P_0 in the subtree of P_1. Lemma 2 shows that all nodes dominated by a descendant node of P_0 in the subtree of P_1 would also be dominated by P_0. All constraints added at each ancestor node of P_1 are also posted at P_1. Thus all nodes in the subtree of P_1 which are dominated by any search nodes that are to the left of P_1 are pruned by the constraints posted at P_1. Similarly, ReSBDS/LReSBDS adds extra constraints at each search node P to prune all nodes that are dominated by a deleted node in Π of node P or of some search node that is searched before P. □

In other words, dominated nodes are exactly the ones pruned by the associated symmetry breaking method.

Dominated nodes are deleted nodes if GAC is enforced at each search node on all constraints. If no consistency is enforced, dominated nodes are fail nodes. If all constraints are enforced GAC, ReSBDS utilizes dominated nodes as well as other deleted nodes to do extra prunings in subsequent search. LReSBDS utilizes a subset of dominated nodes to do extra prunings.

Lemma 3. *Suppose all constraints are enforced GAC. A subset of symmetries G are broken by ReSBDS/LReSBDS. All dominated nodes are in the sequence Π of a search node in the resulting search tree when using ReSBDS. All deleted nodes in Π of each search node are dominated nodes when using LReSBDS.*

Proof. ReSBDS adds constraints to prune dominated nodes and all assignments in these constraints are recorded into T. Once a dominated node is pruned by ReSBDS, a recorded assignment in T must be pruned. This dominated node is

thus put into Π. For LReSBDS, all deleted nodes in Π are results of prunings effected by symmetry breaking constraints, i.e. pruned by LReSBDS. Since LReSBDS prunes only dominated nodes, these deleted nodes are dominated. □

If all constraints are made GAC, ReSBDS is stronger than ParSBDS in pruning power (Lee and Zhu 2014a). If no consistency is enforced, we have the following theorem.

Theorem 2. *If no consistency is enforced, ParSBDS, ReSBDS and LReSBDS have the same pruning power.*

Proof. According to Lemma 1, there are no deleted nodes. The sequence Π of each search node is empty. No extra constraints can be added by ReSBDS and LReSBDS. □

Breaking More Composition Symmetries

In the following, we show how search orders can affect the number of dominated nodes. After that, we give a theorem stating when the entire symmetry group is eliminated in partial symmetry breaking. A possibility is when the search tree is *Dominance-Complete* (DC). Unfortunately, it is not always possible to build such a search tree. We propose two effective search heuristics to approximate DC-ness.

We use as example the 2×2 unconstrained matrix problem with domain size 2. The problem model has no constraints but a 2×2 matrix of variables $\{x_{11}, x_{12}, x_{21}, x_{22}\}$, one for each square with domain $\{1, 2\}$. The generator symmetries are $G = \{R, C\}$, where R and C are interchangeable rows and columns respectively. Figures 2 and 3 give two solution symmetry classes. Symmetric solutions are numbered and connected by lines with double arrows marked by the corresponding generator symmetries.

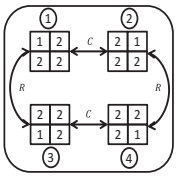

Figure 2: Symmetry Class α

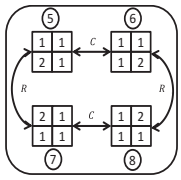

Figure 3: Symmetry Class β

We show two partial search trees under different search orders with ParSBDS and no consistency enforcement in Figures 4 and 5 in which the gray triangles are pruned. Each solution in symmetry class α (Figure 2) is marked by its solution number. Solutions 2 and 3 are pruned in Figure 4 since P_2 and P_3 are dominated by P_0 and P_1 ($A_0^R \subseteq E_3$, $A_0^C \subseteq E_2$, $A_1^R \subseteq E_2$ and $A_1^C \subseteq E_3$ where A_i and E_i denote the partial assignment and the set of entailed assignments of node P_i). Solution 4 in Figure 4 is not pruned since P_0 cannot dominate solution 4 which is only descendant symmetric, but not G-descendant symmetric, to P_0. Solutions 2, 3 and 4 are pruned in Figure 5 as P_3 is dominated by nodes P_1 and P_2 ($A_1^R \subseteq E_3$, $A_2^C \subseteq E_3$). In Figure 4, solution 4 is not dominated since it precedes solutions 2 and 3 to which solution 4 is only symmetric according to the given generators individually. This shows that search orders affect the number of dominated nodes.

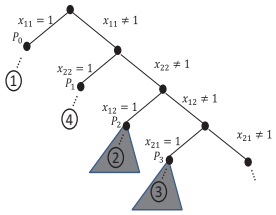

Figure 4: First Search Tree

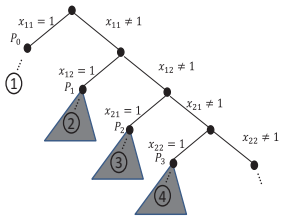

Figure 5: Second Search Tree

To analyze how symmetry breaking methods can break composition symmetries, we give the following definitions. Consider a symmetry group Σ, and a search tree τ and two nodes P_0 and P_1 of τ. P_0 *rules over* P_1 iff P_0 precedes P_1 and P_1 is descendant symmetric to P_0. A node P is a *master node* iff it is not ruled over by any node in τ. A symmetry breaking method *eliminates* a symmetry g in a search tree τ iff for any two search nodes which are not fail nodes, no one rules over the other according to g. A symmetry breaking method *eliminates* a symmetry group Σ in τ iff all symmetries in Σ except the identity one are eliminated in τ.

If the entire symmetry group Σ is posted, P_0 dominates P_1 if P_0 rules over P_1. Theorem 1 proves all dominated nodes are pruned by SBDS. Thus the entire symmetry group is eliminated since all non-master nodes are dominated and thus pruned. This explains why SBDS is complete. For partial symmetry breaking, we give the following theorem following directly from the definition of "eliminate".

Theorem 3. *Consider a set of symmetries G to break whose symmetry group is Σ. Σ is eliminated by ParSBDS/ReSBDS/LReSBDS in a search tree τ if every node P of τ is either a master node or a dominated node.*

Master nodes would never be dominated and pruned by symmetry breaking methods. We thus would like to dominate *all* non-master nodes so that they can be pruned.

Dominance Completeness

Solution 4 is not dominated in Figure 4 since it is only ruled over but not dominated by solution 1 and all of solution 1's ancestor nodes are to the left of solution 4. How can we make a ruled over node P, which is not dominated by a master node, dominated?

The following theorem gives a way to use dominated nodes to dominate more nodes during search.

Theorem 4. *Consider a subset of symmetries G, and a search tree τ and two nodes P_0 and P_1 of τ where P_0 with partial assignment A_0 dominates P_1 according to symmetry $g \in G$. If the partial assignment of P_1 is A_0^g, all nodes that are ruled over by P_0 according to symmetry $g \circ h$ for all $h \in G$ and preceded by P_1 are dominated by P_1.*

Proof. Suppose P_2 is ruled over by P_0 according to symmetry $g \circ h$ where $h \in G$, i.e. $A_0^{g\circ h} \subseteq E_2$ where E_2 is the set of entailed assignments of P_2. As P_1 has partial assignment

A_0^g, P_2 is G-descendant symmetric to P_1 according to h. If P_1 also precedes P_2, P_1 dominates P_2. □

Consider a subset of symmetries G and a search tree τ. Suppose for any dominated node P_1 which is dominated by a node P_0 with partial assignment A_0 according to symmetry $g \in G$, P_1 has partial assignment A_0^g and precedes all its G-descendant symmetric nodes which are not dominated by any node that precedes P_1 or is searched before the search node where P_1 is pruned. τ is *Dominance-Complete* (DC).

Theorem 4 ensures that all nodes that are ruled over but not dominated by a master node would be dominated by a dominated node in a DC search tree. The entire symmetry group Σ is thus eliminated by Theorem 3.

Theorem 5. *Consider a set of symmetries G to break whose symmetry group is Σ. Σ is eliminated by ParSBDS/ReSBDS/LReSBDS in a DC search tree.*

In a DC tree, assuming depth first search, the nodes to visit next are the ones already dominated by the searched part. After they are visited, all their G-descendant symmetric nodes which are not visited yet are dominated. The following theorem gives a special case when a DC tree can be built.

Theorem 6. *Consider a fixed variable (value) ordering γ and adjacent variables (values) interchangeability G. Applying ReSBDS on G and searching with the γ variable (value) ordering builds a DC search tree.*

Proof. Suppose a constraint $\neg(A \wedge (x = v))^g$ for $g \in G$ is posted by ReSBDS. This constraint has not been satisfied iff $A^g = A$ and $x^g \neq x$ ($v^g \neq v$). Now $\neg(x = v)^g$ is immediately enforced to be true and a deleted node P' with partial assignment $A \wedge (x = v)^g$ is generated and put into the Π sequence by ReSBDS. As $A^g = A$, the partial assignment of P' is equivalent to $(A \wedge (x = v))^g$. Moreover, since the search order is γ, P' precedes all its G-descendent symmetric nodes which have not been visited or dominated yet. Thus, the resultant search tree is DC. □

We do not have an efficient algorithm to construct search trees that are DC, when possible, in general yet. However, we have examined all search trees of the 2×3 unconstrained matrix problem with domain size 2, given only adjacent row and column interchangeability as symmetries. None satisfies DC and none eliminates the entire symmetry group. In the following, we propose two efficient search heuristics aiming at dominating as many nodes as possible.

Most Frequent Decision First

ParSBDS/ReSBDS/LReSBDS adds symmetry breaking constraints to prune dominated nodes. Each of these constraints is the negation of a set of assignments. All nodes in subsequent search which entail such a set of assignments are dominated nodes. DC can be approximated in the following way. Once a symmetry breaking constraint $\neg(A \wedge (x = v))^g$ is added, simply choose one of the assignments in this constraint as the next branching decision until search has to backtrack (in which case this constraint is violated). Now all nodes which entail $A^g \wedge (x = v)^g$ are searched first, and all their G-descendant symmetric nodes which are not visited yet would be dominated. If there are more than one symmetry breaking constraints posted by ParSBDS/ReSBDS/LReSBDS in the constraint store, we choose *the decision which occurs most frequently among all symmetry breaking constraints in the constraint store at the current search node* as the next branching decision. Such a decision tends to allow more dominated nodes to be encountered in subsequent search. We call this the *Most Frequent Decision First* (MFDF) heuristic.

Keeping Most Symmetries Active

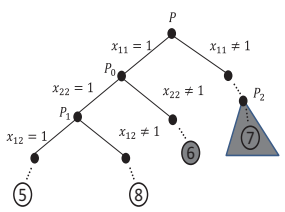

Figure 6: First Search Tree

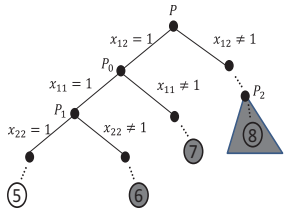

Figure 7: Second Search Tree

The MFDF heuristic does not help if there are no symmetry breaking constraints in the constraint store yet. Consider another two partial search trees under different search orders with ParSBDS and no consistency enforcement in Figures 6 and 7 for solutions in symmetry class β (Figure 3). The gray triangles are pruned and gray solution nodes outside of gray triangles are fail nodes. At nodes P, P_0 and P_1 of each search tree, MFDF cannot choose any useful decision to branch since no symmetry breaking constraints are in the constraint store yet. In Figure 6, the search order $x_{11} = 1 \wedge x_{22} = 1$ excludes solutions 6 and 7 from the subtree of P_1. There is no way to let these two solutions precede and thus dominate solution 8 in subsequent search anymore. In Figure 7, solution 8 is pruned as P_2 is dominated by the common ancestor P_0 of solutions 6 and 7.

We thus try to build a search order such that G-descendant symmetric nodes (solutions 6 and 7) of a node P (solution 5) are visited earlier than other nodes (solution 8) that are ruled over but not dominated yet. To achieve this, we choose *the decision which keeps most symmetries active* as the next branching decision. In this way, the given symmetries are broken at deeper levels and the deepest common ancestor of a node P and its G-descendant symmetric nodes are very close to the leaves. This increases the chance that, once a master node is visited, its G-descendant symmetric nodes is visited soon. We call this the *Most Symmetries Active* (MSA) heuristic. The number of active symmetries of a CSP are reduced rapidly along the branching. Thus MSA is useful usually at upper search levels.

We combine MSA and MFDF (MFDF+MSA) in the following way: if there are symmetry breaking constraints in the constraint store of the current search node, use MFDF; otherwise, use MSA. In the experimental part, we compare

MFDF, MSA and MFDF+MSA against the smallest domain first (with ties broken by the input variable order) and minimum value (SDF+MV) heuristic. Our three heuristics break ties by SDF+MV.

Experimental Results

This section gives five experiments. We choose the most efficient dynamic partial symmetry breaking method LReSBDS (Lee and Zhu 2014b) to break the given generator symmetries under SDF+MV, MSA, MFDF and MFDF+MSA respectively. We compare against Doublelex (Flener et al. 2002) which is an efficient partial static method to break matrix symmetries, Precedence (Law and Lee 2004) which is an efficient static method to break value interchangeability and VAR constraints (Puget 2005b; 2005a) which is an efficient static method to break variable symmetries with alldifferent constraints. We also compare against the complete and dynamic method SBDS (Gent and Smith 2000). *Unless otherwise specified*, the search order is defaulted to SDF+MV. For better efficiency, the increasing nogoods global constraint and its filtering algorithm (Lee and Zhu 2014b) are used in LReSBDS and SBDS. We ran more instances than those reported here for space reasons, and report only the representative cases.

All experiments are conducted using Gecode Solver 4.2.0 on Intel C2D E8400 3.0Ghz (7GB). In our tables, $\#s$, $\#f$ and t denote number of solutions, failures and runtimes in seconds respectively. The best results are highlighted in **bold**.

Unconstrained Matrix Problem.

To show the improvement of our heuristics without interaction with problem constraints, we solve the unconstrained matrix problem which is modeled by an $n \times m$ matrix of variables with domain $\{1, \ldots, d\}$. This model has matrix symmetries, value interchangeability and their compositions. We first deploy DoubleLex and Precedence. LReSBDS is given adjacent rows (columns) interchangeability, cartesian-product of adjacent rows interchangeability and adjacent columns interchangeability, and adjacent value interchangeability to break.

MFDF reduces the number of solutions in all benchmarks (Table 1) and MSA is on par with SDF+MV. The combination MFDF+MSA performs the best. The solution size is reduced to 39% on average when compared to SDF+MV. This shows more composition symmetries are broken. The runtime is improved 2.30 times on average taking advantage of the search tree reduction. Compared with the static method, MFDF+MSA runs 3.23 times faster on average.

Error Correcting Code-Lee Distance.

The Error Correcting Code-Lee Distance problem is prob036 in CSPLib (Gent and Walsh 1999). We follow Lee and Li (2012) in the modeling and turning the optimization problem into a satisfaction problem with parameters (n, c, b) to illustrate the effect on solution set size. This model has matrix symmetries and no value symmetries. Thus, Precedence is not used here. We post the same subset of matrix symmetries to break as in the unconstrained matrix problem for LReSBDS.

More composition symmetries are broken using MFDF (Table 2). MSA, however, seems to interact badly with the problem constraints, and make bad early decisions. The combined heuristic MFDF+MSA reduces the solution size to 60% on average. However, its runtime is not the fastest since MSA does not cooperate well with the problem constraints. MFDF runs the fastest and is 1.51 times faster on average than SDF+MV. Compared with the static method DoubleLex, MFDF runs 3.02 times faster on average taking advantage of breaking more symmetries.

NNQueen.

The NNQueen problem is to color an $n \times n$ chessboard with n colors, such that no lines (row, column or diagonal) contain the same color twice (Kelsey, Linton, and Roney-Dougal 2004). We model it with n^2 variables, one per square of the chess board. It has 8 geometric symmetries, value interchangeability and their compositions. We first deploy VAR and Precedence. LReSBDS is given the 8 geometric symmetries and adjacent value interchangeability.

MFDF does not contribute (Table 3) since all symmetry breaking constraints added by LReSBDS are unconditional in the resultant search tree and thus dominated nodes are pruned immediately. Only exploiting MSA in LReSBDS, half of the solutions are eliminated. Moreover, MSA runs 1.53 and 1.38 times faster than SDF+MV and VAR+Precedence on average respectively.

Diagonal Latin Square.

The Diagonal Latin Square problem is to assign n numbers to the cells of an $n \times n$ board with no numbers occurring more than once in each row, column and the two diagonals. This problem is modeled as an $n \times n$ matrix of variables, each with domain $\{1, \ldots, n\}$. It has the same set of symmetries with NNQueen and all three symmetry breaking methods work the same as that in NNQueen. Table 4 shows the results. With similar reason to the case of NNQueen, MFDF only changes the search tree slightly. Exploiting MSA, half of the solutions are eliminated. MSA runs 2.03 and 1.98 times faster than SDF+MV and VAR+Precedence on average respectively.

N-Queens.

We model the N-Queens problem the standard way using one variable per column. All 8 geometric symmetries are given to SBDS. We give LReSBDS only the two generators rx (reflection on the vertical axis) and $d1$ (reflection on the diagonal). Since all constraints added by LReSBDS according to these two symmetries are unconditional, MFDF is not useful in here. We thus only run LReSBDS under the default heuristic and MSA. When using MSA, we choose to keep symmetry rx active when N is odd and keep symmetry $d1$ active when N is even.

When compared with the complete SBDS which eliminates all symmetric solutions, MSA only leaves 0.48% redundant symmetric solutions on average (Table 5). This

Table 1: Unconstrained Matrix Problem

	DoubleLex+Precedence		LReSBDS		$\text{LReSBDS}_{\text{MFDF}}$		$\text{LReSBDS}_{\text{MSA}}$		$\text{LReSBDS}_{\text{MFDF+MSA}}$	
n, m, d	$\#s$	t	$\#s$	t	$\#s$	t	$\#s$	t	$\#s$	t
3 6 4	29,808,607	59.97	19,080,577	41.73	12,006,046	28.92	19,320,572	50.25	**8,703,246**	**22.98**
4 4 4	7,493,397	14.57	4,867,420	9.95	3,084,246	6.34	3,832,171	8.66	**2,082,126**	**4.69**
4 5 4	621,064,372	1,267.37	359,626,461	835.16	202,192,112	459.11	289,610,732	673.60	**122,929,688**	**295.44**
3 5 5	18,760,547	37.81	13,766,350	27.90	7,828,201	16.87	12,008,380	26.74	**5,300,942**	**13.04**
3 6 5	676,207,419	1,402.60	471,039,224	1,036.72	253,358,428	551.98	470,773,863	1,177.35	**174,712,527**	**423.80**
4 4 5	78,724,080	148.87	58,598,557	116.63	34,925,945	70.18	37,913,437	81.74	**20,535,746**	**47.31**

Table 2: Error Correcting Code-Lee Distance

	DoubleLex			LReSBDS			$\text{LReSBDS}_{\text{MFDF}}$			$\text{LReSBDS}_{\text{MSA}}$			$\text{LReSBDS}_{\text{MFDF+MSA}}$		
n, c, b	$\#s$	$\#f$	t	$\#s$	$\#f$	t	$\#s$	$\#f$	t	$\#s$	$\#f$	t	$\#s$	$\#f$	t
6,4,4	4.70M	4.03M	173.99	2.33M	2.11M	95.04	1.54M	**1.29M**	**58.91**	2.39M	3.08M	116.84	**1.51M**	1.65M	68.09
5,6,5	1.44M	6.26M	210.89	0.64M	3.00M	99.93	0.41M	**1.95M**	**66.45**	0.63M	7.61M	249.40	**0.38M**	3.82M	130.23
5,6,6	0.30M	16.85M	664.51	0.14M	8.38M	343.17	0.09M	**7.19M**	**291.81**	0.11M	38.09M	1,650.21	**0.08M**	20.65M	929.60
8,4,4	35.63M	44.44M	1,966.97	16.96M	22.68M	1,042.14	11.03M	**13.79M**	**648.12**	18.18M	33.77M	1,384.89	**10.84M**	16.95M	729.61
6,4,5	29.35M	73.05M	3,045.02	12.83M	32.46M	1,392.44	7.60M	**18.81M**	**835.21**	13.02M	36.04M	1,564.07	**7.46M**	20.76M	915.48

demonstrates MSA can eliminate almost the entire symmetry group given only the two generators. Now MSA runs 2.29 times faster than SBDS on average taking advantage of less symmetries posted and the efficient heuristic. The search tree size is reduced substantially when N is odd and only slightly when N is even.

Conclusion

Our contributions are two efficient search heuristics grounded on formal analysis. First, we generalize the notion of dominance and characterize exactly the prunings effected by ParSBDS and variants. Second, we show how search orders can affect the number of dominated nodes given a subset of symmetries and give a theorem to state when the entire symmetry group is eliminated. Third, we introduce DC-ness and prove that a DC search tree ensures symmetry group elimination. Fourth, we propose two decision-based search heuristics as well as their combination to approximate DC effectively as a result of theoretical studies. Fifth, we demonstrate the *consistent* advantages of our proposal with extensive experimentation.

Our method focuses on binary branching, and can be generalized easily to n-ary branching (or labeling). A limitation of our method is that it will not work with non decision-based branching, such as bisection branching ($<$ on the left and $\geq$ on the right).

SBDD (Fahle, Schamberger, and Sellmann 2001; Gent et al. 2003) is another widely used dynamic symmetry breaking method, which works by checking whether the current search node is dominated by a node that is visited earlier. It will be interesting to study if our approach can also be extended to partial symmetry breaking version of SBDD. In addition, Sellmann and Van Hentenryck (2005) propose DSSB, a polynomial-time dominance-detection algorithm, to break simultaneous piecewise variable and value symmetry. It is also interesting to investigate the relation between DSSB and DC tree.

References

Bessiere, C., and Régin, J.-C. 1996. Mac and combined heuristics: Two reasons to forsake FC (and CBJ?) on hard problems. In *CP'96*, 61–75.

Boussemart, F.; Hemery, F.; Lecoutre, C.; and Sais, L. 2004. Boosting systematic search by weighting constraints. In *ECAI'04*, 146–150.

Crawford, J.; Ginsberg, M.; Luks, E.; and Roy, A. 1996. Symmetry breaking predicates for search problems. In *KR'96*, 148–159.

Dechter, R., and Pearl, J. 1988. Network-based heuristics for constraint-satisfaction problems. *Artificial Intelligence* 34:1–38.

Fahle, T.; Schamberger, S.; and Sellmann, M. 2001. Symmetry breaking. In *CP'01*, 93–107.

Flener, P.; Frisch, A.; Hnich, B.; Kiziltan, Z.; Miguel, I.; Pearson, J.; and Walsh, T. 2002. Breaking row and column symmetries in matrix models. In *CP'02*, 187–192.

Frost, D., and Dechter, R. 1995. Look-ahead value ordering for constraint satisfaction problems. In *IJCAI'95*, 572–578.

Gent, I., and Smith, B. 2000. Symmetry breaking in constraint programming. In *ECAI'00*, 599–603.

Gent, I., and Walsh, T. 1999. CSPLib: a benchmark library for constraints. In *CP'99*, 480–481.

Gent, I. P.; Harvey, W.; Kelsey, T.; and Linton, S. 2003. Generic SBDD using computational group theory. In *CP'03*, 333–347.

Grayland, A.; Miguel, I.; and Roney-Dougal, C. M. 2009. Snake Lex: an alternative to Double Lex. In *CP'09*, 391–399.

Jefferson, C., and Petrie, K. 2011. Automatic generation of constraints for partial symmetry breaking. *CP'11* 729–743.

Kelsey, T.; Linton, S.; and Roney-Dougal, C. 2004. New developments in symmetry breaking in search using computational group theory. In *Artificial Intelligence and Symbolic Computation*, 199–210.

Table 3: NNQueen

	VAR+Precedence			LReSBDS			$\text{LReSBDS}_{\text{MFDF}}$			$\text{LReSBDS}_{\text{MSA}}$			$\text{LReSBDS}_{\text{MFDF+MSA}}$		
n	$\#s$	$\#f$	t	$\#s$	$\#f$	t	$\#s$	$\#f$	t	$\#s$	$\#f$	t	$\#s$	$\#f$	t
7	4	854	0.02	4	860	0.02	4	860	0.02	**2**	**459**	**0.01**	**2**	**459**	**0.01**
8	**0**	149,573	1.99	**0**	149,580	2.14	**0**	149,580	2.18	**0**	**109,304**	**1.50**	**0**	**109,304**	**1.50**
9	**0**	140,316,433	2,076.10	**0**	140,316,441	2,301.54	**0**	140,316,441	2,303.44	**0**	**94,994,393**	**1,503.19**	**0**	**94,994,393**	1,526.57

Table 4: Diagonal Latin Square

	VAR+Precedence			LReSBDS			$\text{LReSBDS}_{\text{MFDF}}$			$\text{LReSBDS}_{\text{MSA}}$			$\text{LReSBDS}_{\text{MFDF+MSA}}$		
n	$\#s$	$\#f$	t	$\#s$	$\#f$	t	$\#s$	$\#f$	t	$\#s$	$\#f$	t	$\#s$	$\#f$	t
6	128	814	**0.01**	128	819	**0.01**	128	945	**0.01**	**64**	**569**	**0.01**	**64**	**569**	**0.01**
7	171,200	168,098	2.48	171,200	168,104	2.53	171,200	175,738	2.88	**85,600**	**72,050**	**1.25**	**85,600**	**72,050**	**1.25**

Table 5: N-Queens

	SBDS			LReSBDS			$\text{LReSBDS}_{\text{MSA}}$		
N	$\#s$	$\#f$	t	$\#s$	$\#f$	t	$\#s$	$\#f$	t
15	**285,053**	2,861,030	19.44	324,173	3,092,401	12.82	287,095	**1,621,554**	**7.11**
16	**1,846,955**	17,553,738	121.48	2,071,568	18,884,559	81.09	1,854,141	**16,676,355**	**72.42**
17	**11,977,939**	114,336,279	816.30	13,388,788	123,139,684	542.16	12,046,226	**66,466,624**	**305.81**
18	**83,263,591**	789,951,820	5,830.79	91,967,520	847,352,319	3,721.53	83,526,571	**733,909,932**	**3,318.02**
19	**621,012,754**	5,726,127,348	42,252.00	685,143,337	6,139,933,399	27,665.70	623,676,737	**3,383,027,220**	**16,295.00**

Law, Y. C., and Lee, J. 2004. Global constraints for integer and set value precedence. In *CP'04*, 362–376.

Lee, J., and Li, J. 2012. Increasing symmetry breaking by preserving target symmetries. In *CP'12*, 422–438.

Lee, J., and Zhu, Z. 2014a. Boosting SBDS for partial symmetry breaking in constraint programming. In *AAAI'14*, 2695–2702.

Lee, J., and Zhu, Z. 2014b. An increasing-nogoods global constraint for symmetry breaking during search. In *CP'14*, 465–480.

Mackworth, A. K. 1977. Consistency in networks of relations. *Artificial intelligence* 8(1):99–118.

McDonald, I., and Smith, B. 2006. Partial symmetry breaking. In *CP'06*, 207–213.

Mears, C.; de la Banda, M. G.; Demoen, B.; and Wallace, M. 2014. Lightweight dynamic symmetry breaking. *Constraints* 19(3):195–242.

Puget, J.-F. 2002. Symmetry breaking revisited. In *CP'02*, 446–461. Springer.

Puget, J.-F. 2005a. Breaking symmetries in all different problems. In *IJCAI'05*, 272–277.

Puget, J.-F. 2005b. Elimination des symétries dans les problemes injectifs. In *Premières Journées Francophones de Programmation par Contraintes*.

Roney-Dougal, C. M.; Gent, I. P.; Kelsey, T.; and Linton, S. 2004. Tractable symmetry breaking using restricted search trees. In *ECAI'04*, 211–215.

Sellmann, M., and Van Hentenryck, P. 2005. Structural symmetry breaking. In *IJCAI'05*, 298–303.