

Structural Symmetry Breaking

Meinolf Sellmann and Pascal Van Hentenryck

Brown University, Department of Computer Science

P.O. Box 1910, Providence, RI 02912, U.S.A.

sello,pvh@cs.brown.edu

Abstract

Symmetry breaking has been shown to be an important method to speed up the search in constraint satisfaction problems that contain symmetry. When breaking symmetry by dominance detection, a computationally efficient symmetry breaking scheme can be achieved if we can solve the dominance detection problem in polynomial time. We study the complexity of dominance detection when value and variable symmetry appear simultaneously in constraint satisfaction problems (CSPs) with single-valued variables and set-CSPs. We devise an efficient dominance detection algorithm for CSPs with single-valued variables that yields symmetry-free search trees and that is based on the abstraction to the actual, intuitive structure of a symmetric CSP.

1 Introduction

During the past five years, symmetry breaking has been a topic of increasing interest in the constraint programming community. It was shown that symmetry breaking can play a decisive role in the solution of numerous constraint problems, and sophisticated symmetry breaking methods have been developed, such as the addition of symmetry breaking constraints (see for example [Crawford *et al.*, 1996]), symmetry breaking by adapting the search strategy [Brown *et al.*, 1988], symmetry breaking during search (SBDS) [Gent and Smith, 2000], or symmetry breaking by dominance detection (SBDD) [Fahle *et al.*, 2001; Focacci and Milano, 2001]. Especially the latter has attracted a lot of interest: improvements were suggested in [Barnier and Brisset, 2002; Puget, 2002] for example. SBDD works by checking whether the current choice point under investigation represents a symmetric variant of a part of the search space that has been investigated completely before.

The core of an SBDD symmetry breaking code is dominance detection which was automated in [Gent *et al.*, 2003] by using the generic computational group theory tool, yielding a method named GAP-SBDD. While using a generic tool is appealing from the perspective of the ease-of-use for the constraint programmer, the generality comes with no guarantee of efficiency. Therefore, in [Van Hentenryck *et al.*, 2003], the special case of dominance detection in the presence of

some classes of pure value symmetry was investigated and proven to be computationally tractable. The approach taken was generalized in [Roney-Dougal *et al.*, 2004], where the notion of symmetry-free search trees, so-called GE-trees, was introduced. In combination with GAP, GE-trees were shown to break arbitrary value symmetries in polynomial time.

In this paper, we continue this successful line of research by studying combinations of value and variable symmetry in constraint satisfaction problems (CSPs) and CSPs with set variables (set-CSPs). We offer three main contributions: The first regards a combination of variable and value symmetry for CSPs with single-valued variables. For the first time, in Section 3, we show that the corresponding dominance detection problem is computationally tractable. In Section 4, we build up on this result and show an efficient way of using the dominance detection algorithm for filtering rather than just pruning. The third main contribution regards set variables: in Section 5 we show that the general dominance detection problem becomes NP-hard for set-CSPs that contain symmetric set variables and value symmetry. Finally, as a minor side-note, in Section 6 we show that higher forms of symmetry where sets of set variables are symmetric to other sets of set variables in combination with value symmetry can be reduced to the partial set variable symmetry case.

2 Preliminaries

Let us start out by introducing some notation that we will use throughout the paper. In the remainder of this paper, let us assume that $n, m, p \in \mathbb{N}$.

Definition 1

- A Constraint Satisfaction Problem (CSP) is a tuple (Z, V, D, C) where $Z = \{X_1, \dots, X_n\}$ is a finite set of variables, $V = \{v_1, \dots, v_m\}$ is a set of values, $D = \{D_1, \dots, D_n\}$ is a set of finite domains where each $D_i \in D$ is the set of possible instantiations to variable X_i , and $C = \{c_1, \dots, c_p\}$ is a finite set of constraints where each $c_i \in C$ is defined on a subset of the variables in Z and specifying their valid combinations. We say that the CSP has single-valued variables iff for all $D_i \in D$ it holds that $D_i \subseteq V$. If for all $D_i \in D$ it holds that $D_i \subseteq 2^V$ (where 2^V denotes the set of all subsets of V), we say that the CSP is a set-CSP.

- Given a CSP with single-valued variables, an assignment A is a set of pairs $(X, v) \in Z \times V$ such that $(X, v), (X, w) \in A$ implies $v = w$. Given a set-CSP, a set-assignment A is a set of pairs $(X, S) \in Z \times 2^V$ such that $(X, S_1), (X, S_2) \in A$ implies $S_1 = S_2$.

Definition 2

- Given a set S and a set of sets $P = \{P_1, \dots, P_r\}$ such that $\bigcup_i P_i = S$ and the P_i are pairwise non-overlapping, we say that P is a partition of S , and we write $S = \sum_i P_i$.
- Given a set S and a partition $S = \sum_i P_i$, a bijection $\pi : S \mapsto S$ such that $\pi(P_i) = P_i$ (where $\pi(P_i) = \{\pi(s) \mid s \in P_i\}$) is called a partial permutation over $S = \sum_i P_i$.

Definition 3

- Given a CSP (Z, V, D, C) , and partitions $Z = \sum_{k \leq r} P_k$, $V = \sum_{l \leq s} Q_l$, we say that the CSP has partial variable and value symmetry iff all variables within each P_k and all values within each Q_l are considered as symmetric.
- Given two assignments A and B on a partially symmetric CSP with single-valued variables, we say that A dominates B iff there exist partial permutations π over $Z = \sum_{k \leq r} P_k$ and α over $V = \sum_{l \leq s} Q_l$ such that for all $(X, v) \in A$ it holds that $(\pi(X), \alpha(v)) \in B$.
- Given a partially symmetric set-CSP and set-assignments A and B , we say that A dominates B iff there exist partial permutations π over $Z = \sum_{k \leq r} P_k$ and α over $V = \sum_{l \leq s} Q_l$ such that $A(\pi, \alpha) \subseteq B$, where $A(\pi, \alpha) := \{(\pi(X), \alpha(S)) \mid (X, S) \in A\}$.
- Given two arbitrary (set-)assignments A and B for a partially symmetric (set-)CSP, we call the problem of determining if A dominates B the Dominance Detection Problem.

3 Symmetric Single-Valued Variables over Symmetric Values

The first general symmetry model that we study is powerful enough to manage symmetric single-valued variables over symmetric values. Throughout this section, we consider the partially symmetric CSP (Z, V, D, C) with single-valued variables. We assume also that we are given a partition of the variables $\sum_{k \leq r} P_k = Z$ and a partition of the values $\sum_{l \leq s} Q_l = V$ such that all variables within each P_k and all values within each Q_l are considered symmetric. Note that such partitions could be derived by a static analysis of a constraint program. The main objective in this section will be to show that there exists an effective symmetry breaking algorithm that runs in polynomial time for this scenario.

The key idea consists in the introduction of *structural abstractions*: to model a CSP, we need to uniquely label each value and each variable with a name — which is, of course, not natural when certain variables and certain values are actually indistinguishable. We can rectify this by viewing each

variable and each value as a member of a symmetry class. In the beginning, these classes correspond directly to the sets P_k and Q_l . When assignments are committed, though, some of those initial symmetries are broken. Then, in order to check which CSP objects are still indistinguishable, we need to introduce subclasses of the original symmetry classes. We will see that we can detect the remaining symmetries by labeling each of those subclasses with a certain *signature* that is defined by the set of initial symmetries and the given assignments. We will see also that it is really these signatures that capture our intuitive wish to abstract from the CSP model at hand to the actual structure of the problem.

3.1 Signatures

Consider the following example: We have variables X_1, \dots, X_8 over domains $D(X_1) = \dots = D(X_8) = \{v_1, \dots, v_6\}$. Now assume that the first four and the last four variables are indistinguishable, i.e. $P_1 = \{X_1, \dots, X_4\}$ and $P_2 = \{X_5, \dots, X_8\}$. Furthermore, assume that $Q_1 = \{v_1, \dots, v_3\}$, $Q_2 = \{v_4, \dots, v_6\}$, and that we are given the following two assignments: $A_1 = \{(X_1, v_1), (X_2, v_1), (X_3, v_2), (X_6, v_5), (X_7, v_1), (X_8, v_2)\}$ and $A_2 = \{(X_1, v_6), (X_2, v_1), (X_3, v_2), (X_4, v_2), (X_5, v_1), (X_6, v_6), (X_7, v_2), (X_8, v_2)\}$. See Figure 1(a) for an illustration. When looking at the first assignment, we see that: 1. There is one value (v_1) in Q_1 that is taken by two variables in P_1 and one variable in P_2 . 2. There is one value (v_2) in Q_1 that is taken by one variable in P_1 and one variable in P_2 . 3. There is one value (v_5) in Q_2 that is taken by one variable in P_2 . On the other hand, in the second assignment: I. There is one value (v_2) in Q_1 that is taken by two variables in P_1 and two variables in P_2 . II. There is one value (v_1) in Q_1 that is taken by one variable in P_1 and one variable in P_2 . III. There is one value (v_6) in Q_2 that is taken by one variable in P_1 and one variable in P_2 . Lining up 1-I ($v_1 \mapsto v_2, \{X_1, X_2\} \mapsto \{X_3, X_4\}, \{X_7\} \mapsto \{X_7, X_8\}$), 2-II ($v_2 \mapsto v_1, \{X_3\} \mapsto \{X_2\}, \{X_8\} \mapsto \{X_5\}$), and 3-III ($v_5 \mapsto v_6, \{X_6\} \mapsto \{X_6\}$), we see that A_2 is structurally an assignment extended from A_1 , or, in other words, that A_1 dominates A_2 (see also Figure 1(b)).

What we have done in this small example is to abstract from the given model and the (arbitrary) names of variables and values to the actual *structure* of the problem. That is, instead of talking about specific variables and values, we considered members of classes. Specifically, for each assignment we implicitly assigned each value a *signature* that captures by how many members of each variable-symmetry class it was taken. For instance, in A_1 v_1 has the signature $(2 \times P_1, 1 \times P_2)$, or, in shorter writing, the signature of v_1 is $sig_{A_1}(v_1) = (2, 1)$. In A_2 , on the other hand, the signature of v_2 is $sig_{A_2}(v_2) = (2, 2)$. Consequently, v_2 in A_2 can be viewed as more specialized than v_1 in A_1 , or one may also say that v_1 in A_1 *dominates* v_2 in A_2 . In this terminology, v_1 in A_2 has signature $sig_{A_2}(v_1) = (1, 1)$ and therefore dominates v_1 in A_1 . Note that $sig_{A_2}(v_6)$ is also $(1, 1)$, but that v_6 in A_2 does not dominate v_1 in A_1 since $v_6 \in Q_2$ whereas $v_1 \in Q_1$. In general:

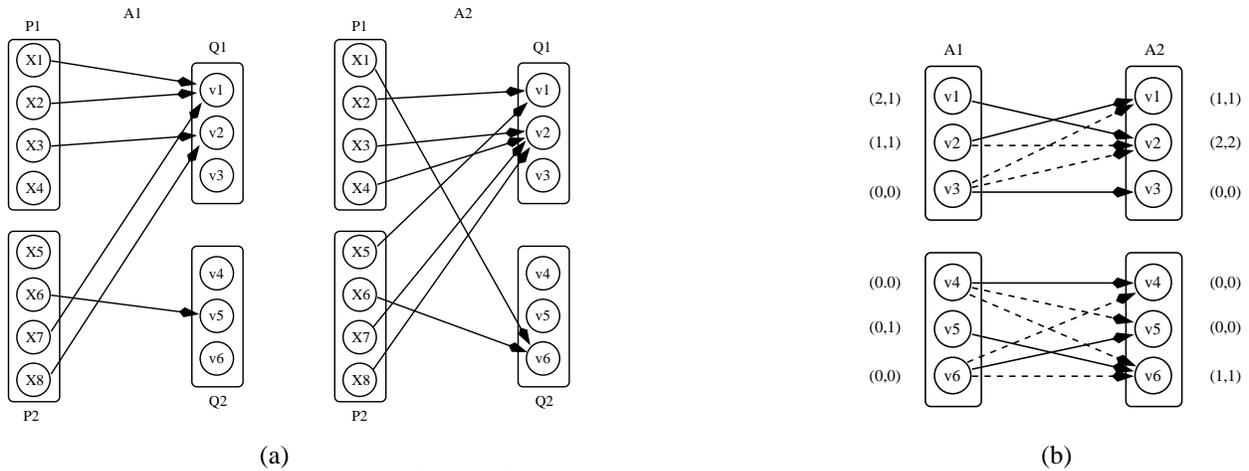


Figure 1: Part (a) illustrates assignments A_1 and A_2 . Part (b) gives the signatures for each value, links pairs of values where the one in assignment A_1 dominates to the one in A_2 , and a perfect matching that proves that A_1 dominates A_2 is designated by solid lines.

Definition 4

- We say that a value v in an assignment A dominates a value w in assignment B iff v and w belong to the same value-symmetry class and $sig_A(v) \leq sig_B(w)$.¹
- We say that a value v in an assignment A is structurally equivalent to a value w in assignment B iff v and w belong to the same value-symmetry class and $sig_A(v) = sig_B(w)$.

3.2 Dominance Detection Using Signatures

The following lemma shows how signature-abstractions can help to detect dominance relations among assignments:

Lemma 1

An assignment A dominates another assignment B in a partially symmetric CSP iff there exists a partial permutation α over $\sum_{l \leq s} Q_l$ such that v in A dominates $\alpha(v)$ in B for all $v \in V$.

Proof: First, let us assume that A dominates B . Then, there exist partial permutations π over $\sum_{k \leq r} P_k$ and α over $\sum_{l \leq s} Q_l$ such that for all $(X, v) \in A$ it holds that $(\pi(X), \alpha(v)) \in B$. Since both X and $\pi(X)$ belong to the same symmetry class, we have that for all values $v \in V$ it is $sig_A(v) \leq sig_B(\alpha(v))$, which is the same as to say that v in A dominates $\alpha(v)$ in B . Now assume there exists a partial permutation α over $\sum_{l \leq s} Q_l$ such that $sig_A(v) \leq sig_B(\alpha(v))$ for all $v \in V$. Then, since each variable is assigned to at most one value, there exists a partial permutation π over $\sum_{k \leq r} P_k$ such that for all $(X, v) \in A$ it holds that $(\pi(X), \alpha(v)) \in B$. Thus, we have that A dominates B . ■

Consequently, we have that A dominates B iff there exists a perfect matching in a bipartite graph where the edges are defined by the signature-relation of values (see Figure 1(b)):

¹Where the \leq -relation on vectors is defined as the usual component-wise comparison that yields to the so-called dominance ordering - which is different from a lexicographic ordering.

Definition 5

Given two assignments A and B , let us denote with V' a set of duplicates of the values in V by attaching a prime sign after the values' names (i.e. $V' := \{v' \mid v \in V\}$). Then, we define the dominance detection graph $DDG(A, B) := (V \cup V', E)$ where $E := \{(v, w') \mid v \text{ in } A \text{ dominates } w \text{ in } B\}$ denotes the set of arcs.

Theorem 1

Given two assignments A and B over a CSP with partially symmetric single-valued variables and partially symmetric values, the dominance detection problem between A and B has complexity $O(M + m^2 + mn)$, where $M = O(m^{2.5})$ is the time needed to determine whether there exists a perfect matching in $DDG(A, B)$, m is the number of values, and n is the number of variables.

In the interest of space here and at many other occasions we must omit a formal proof. However, we would like to note that, with Lemma 1, it is clear that the dominance detection problem can be solved basically by determining whether there exists a perfect bipartite matching in $DDG(A, B)$. The additional complexity denoted in the theorem is due to the necessity to construct $DDG(A, B)$ first. It is obvious that this can be achieved in time $O(nm^2)$, which already proves that symmetry breaking in this scenario is tractable. However, the runtime can be improved to the complexity that is claimed here by using sparse representations of signatures. Then, in the algorithm analysis we can exploit that the total number of non-zero signature components is bounded by $O(|A| + |B|) = O(n)$.

Interestingly, it can also be shown that every bipartite graph can also be viewed as a dominance detection graph of a CSP and assignments A and B that can be determined in time linear in the size of the given graph. And therefore, a perfect bipartite matching exists iff A dominates B , which makes the dominance detection problem at least as hard as bipartite matching. In other words, we can show that dominance detection takes time T where $T \in \Omega(M) \cap O(M + m^2 + mn)$.

4 Symmetry-Based Filtering

With Theorem 1, we can break all symmetries when given a CSP with partially symmetric variables and values in polynomial time when using a symmetry breaking by dominance detection (SBDD) approach [Fahle *et al.*, 2001; Focacci and Milano, 2001]. What is annoying in this setting is that we still have to check every choice point to see if it is not dominated by one that was previously expanded, i.e. we still have to touch the garbage in order to see that it is garbage. We will now develop an algorithm that does not suffer from this disadvantage.

We achieve this goal by using dominance detection also for *filtering* rather than just pruning². Obviously, a brute-force approach could simply try assignments out and use the dominance detection algorithm above to perform filtering as well. This procedure would lead to a very poor runtime, though. In the following, we will show that filtering based on symmetry can be performed much more efficiently.

Within SBDD, there exists a natural distinction between two types of filtering that apply: The first consists in making sure that none of the newly created children are symmetric to a node that was fully expanded before the node that is currently branching off. When applying unary branching constraints (which we assume are used here), this can be achieved by shrinking domain variables accordingly. The other, fundamentally different type of “filtering” consists in the creation of children that are also not symmetrical to each other. Both types need to be addressed to achieve a symmetry-free search tree (which corresponds to the GE-trees in [Roney-Dougal *et al.*, 2004]). We distinguish the two types of filtering by naming them differently: *symmetric-ancestor based filtering* and *symmetric-sibling based filtering*.

Symmetric-Ancestor based Filtering

The goal of symmetric-ancestor based filtering is to shrink the domains such that instantiating a variable with one of its domain values will not result in the creation of a search node that is symmetric to one that was previously expanded.

Definition 6

Given a depth-first-search tree T ,³ we say that a choice point B (associated with its homonymous assignment B that captures previously committed unary branching decisions) is ancestor-symmetry resistant iff for all previously fully expanded nodes $A \in T$ (A is called an ancestor of B) and for all variables X and values $v \in D(X)$ it holds that A does not dominate $B \cup \{(X, v)\}$.

Assume that we are currently investigating choice point B and that A is some ancestor node that does not dominate B . Observe that instantiating one more variable $X \in P_k$ for some k by setting $X \mapsto w \in Q_l$ for some l will change only the signature of w from $sig_B(w)$ to $sig_B(w) + e_k$, where e_k denotes the unit vector with a 1 in the k th component. We set $B' := B \cup \{(X, w)\}$. Then, $D_1 := DDG(A, B)$

²With ‘filtering’, we refer to the idea of domain reduction in CP; ‘pruning’ refers to the detection of a sufficient reason to backtrack.

³In the interest of space, we must omit a formal definition of a search tree here. The meaning should become clear from the context.

and $D_2 := DDG(A, B')$ only differ in that the latter bipartite graph may contain some additional edges that must all be incident to w' in the right partition. Obviously, if D_2 contains an m -matching, this matching must contain exactly one of those additional edges. Consequently, if A dominates B' , then D_1 must contain an $m - 1$ -matching. Only if this is the case, work needs to be done to make B ancestor-symmetry resistant with respect to A .

So let us assume that D_1 contains an $m - 1$ -matching. Provided with that matching, using some straight-forward matching theory we can identify efficiently those and only those additional edges that would allow us to transform the existing matching into a perfect one. Furthermore, it can also be shown that those critical edges are independent of the particular $m - 1$ -matching that we computed (For an introduction to matching theory we refer to [Ahuja *et al.*, 1993]). Among those critical edges that, if added, would allow us to construct an m -matching, the only ones that we need to consider are those which run between nodes v and w' with $v, w \in Q_l$ for some $1 \leq l \leq s$ and for which there exists $1 \leq k \leq r$ such that $sig_A(v) \leq sig_B(w) + e_k$. If and only if we find such a pair of nodes, a single extra assignment added to B will result in a successful dominance detection. Precisely, every assignment of w to a previously unassigned variable $X \in P_k$ will result in a dominated choice point. Thus, if we remove w from the domain of X for all unassigned $X \in P_k$, we keep the unique parts of the search space and we never produce choice points that are symmetric to one that was expanded previously to B .

With Theorem 1, the runtime needed for the initial value-matching algorithm is bounded by $O(m^{2.5} + mn)$. We can also prove that a filtering algorithm can be formulated that, once the matching is computed, runs in time $O(m^2 + mn)$.⁴ Therefore, since within SBDD at most $n(m - 1)$ ancestor nodes need to be considered, we can prove the following:

Theorem 2

Given a CSP with partially symmetric single-valued variables and partially symmetric values, we can achieve ancestor-symmetry resistance for a given search node in time $O(nm^{3.5} + n^2m^2)$.

Symmetric-Sibling based Filtering

To achieve full symmetry prevention, we also need to guarantee that newly created siblings are not symmetric to each other. Therefore, after we choose the next variable to be assigned, but before we branch on it, we need to perform one more “filtering” step (it is actually more of an implicit pruning step) where we choose a single representative value out of each equivalence class of values which, when assigned to the chosen variable, would result in the creation of symmetric choice points. Due to the fact that, whenever a sibling dominates another, they both must already be structurally equivalent (see Definition 4), we can avoid producing symmetric siblings simply by choosing exactly one representative value

⁴Again, the main technical idea consists in the use of sparse signatures. The proof of the complexity is not particularly insightful but lengthy, which is why we leave it out here.

among those that are structurally equivalent. The complexity of this filtering step is dominated by that of symmetric-ancestor based filtering.

Putting ancestor and sibling-based filtering together, we have completed our development of an effective symmetry breaking algorithm for CSPs with partial single-valued variable and value symmetry that runs in polynomial time. Note that the practical performance of the algorithms sketched can be enhanced in practice: for example, it is fully sufficient to check against previously expanded nodes for which an $m - 1 - h$ -maximum matching was found only after variable instantiations to h different values have been committed. And as usual, by considering incremental updates of matchings, memory can be traded for cpu-time.

5 Limits of Efficient Dominance Detection

After having developed a polynomial symmetry breaking algorithm for CSPs with partially symmetric single-valued variables in the presence of partial value symmetry, we now show that dominance detection for partially symmetric set variables in the presence of partial value symmetry is NP-hard. More precisely, we reduce the corresponding dominance detection problem to subgraph-isomorphism.

In order to achieve the desired reduction we construct a set-assignment from a graph in the following way:

Definition 7

Given an undirected graph $G = (V, E)$ with $c := |V|$, we define a set of symmetric values $N := \{n_1, \dots, n_c\}$, and a set of symmetric variables $P := \{p_{ij} \mid \{i, j\} \in E\}$. Then, the set-assignment $A(G)$ is defined as $A(G) := \{(p_{ij}, \{n_i, n_j\}) \mid \{i, j\} \in E\}$.

Theorem 3

Given two undirected graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$, G_1 is sub-isomorphic to G_2 iff $A(G_1)$ dominates $A(G_2)$ when all variables and values are considered to be symmetric.

Proof: We start by showing that $A(G_1)$ dominates $A(G_2)$ if G_1 is sub-isomorphic to G_2 . Let $\sigma : V \mapsto V$ bijective such that $\{i, j\} \in E_1$ implies $\{\sigma(i), \sigma(j)\} \in E_2$. Then, for all $(p_{ij}, \{n_i, n_j\}) \in A(G_1)$ it holds that $(p_{\sigma(i), \sigma(j)}, \{n_{\sigma(i)}, n_{\sigma(j)}\}) \in A(G_2)$. Therefore, $A(G_1)$ dominates $A(G_2)$.

Now let us assume that $A(G_1)$ dominates $A(G_2)$. Then, there exist functions $\pi : E_1 \mapsto E_2$ and $\alpha : V \mapsto V$ such that $(p_{ij}, \{n_i, n_j\}) \in A(G_1)$ implies $(p_{\pi(\{i, j\})}, \{n_{\alpha(i)}, n_{\alpha(j)}\}) \in A(G_2)$. By construction of $A(G_2)$, this is equivalent to $\{n_{\alpha(i)}, n_{\alpha(j)}\} \in E$ for all $\{i, j\} \in E$. Thus, α is a sub-isomorphism between G_1 and G_2 . ■

With Theorem 3, it is easy to prove the following

Corollary 1

The dominance detection problem over partially symmetric set variables and partially symmetric values is NP-hard.

Proof: We reduce the problem to subgraph-isomorphism. In order to apply Theorem 3, we need to ensure that both graphs operate over the same set of nodes. In case that the set of nodes of the given graphs differ, it is easy to see that G_1 cannot be sub-isomorphic to G_2 if G_1 contains more nodes than G_2 . In case that G_1 actually contains fewer nodes than G_2 , it is easy to see that we can add isolated nodes to G_1 without affecting subgraph-isomorphism. Then, we have that both graphs contain the same number of nodes, and by relabelling the nodes in both graphs, we may assume that both graphs operate on the same set of nodes. ■

Note that, despite this negative result, in some important special cases the symmetry detection problem for CSPs with partially symmetric set variables and values is still tractable. For instance, when the set variables cannot overlap, the algorithm developed in Section 3 can be adapted easily (by simply exchanging the roles of values and variables) to break all symmetries efficiently.

Corollary 2

The dominance detection problem over partially symmetric non-overlapping set variables and partially symmetric values is tractable.

Note that the dominance detection problem as we consider it here regards arbitrary assignments. This implies that, when the detection problem is tractable, we can break symmetries efficiently. However, the situation changes when we achieve an intractability result like the previous one: Within methods like SBDD the assignments that need to be compared can only differ in a rather specific fashion. We can also show that these more specific dominance detection problems are NP-hard as well, therefore proving that SBDD in its general form is incapable of breaking symmetries in partially symmetric set-CSPs efficiently. However, we would like to stress that this result does not imply that symmetry breaking is NP-hard in general since we do not consider other methods here like remodeling or the adaption of the branching scheme.

6 Higher Forms of Symmetry

Note that our intractability proof above shows that dominance detection over symmetric set variables and symmetric values is already NP-hard when there exists only one set of symmetric set variables and one set of symmetric values. Clearly, the problem stays NP-hard when we allow partial symmetry. Partial symmetry is very helpful when even more complicated forms of symmetry need to be handled. Consider for example the Social Golfer Problem (SGP):

32 golfers want to play in 8 groups of 4 each week, such that any two golfers play in the same group at most once. How many weeks can they do this for?⁵

The problem can be generalized by parameterizing it to g groups of s players each, playing for w weeks (instances are usually written in g - s - w -format). A common model for the problem introduces a set variable for each group, so that intra-group symmetries are broken automatically. Groups within

⁵Problem 10 in CSPLib - <http://www.csplib.org/>.

	Group 1		Group 2		Group 3	
Week 1	1	2	3	4	5	6
Week 2	1	3	2	5	*	*

Table 1: A partial instantiation of SGP 3-2-2.

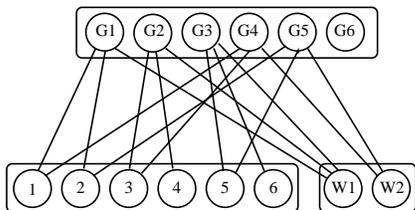


Figure 2: Symmetry-breaking model for SGP 3-2-2 from Table 1. We view the problem as containing six groups G_1, \dots, G_6 (3 per week). Only upon instantiation of the associated set variable, a group is assigned to a week.

the same week are symmetric, and furthermore, whole weeks are also symmetric, i.e. whole sets of set variables are symmetric to other sets of set variables. We can model this by introducing new week-values, one of which each group must be assigned to. Consider the assignment to an SGP instance 3-2-2 given in Table 1: By introducing two week-values W_1 and W_2 to one of which each group-variable is assigned upon instantiation, we can model the SGP as a dominance detection problem over symmetric set variables and partially symmetric values (see Figure 2). By means of this little trick, higher forms of symmetry can be reduced to symmetric set variables over partially symmetric values, which makes the corresponding dominance detection problem a most relevant research subject. We believe that developing fast algorithms that (approximately) solve this NP-hard problem should be a focus of symmetry breaking research at this point.

7 Conclusion

When breaking symmetry by dominance detection, a computationally efficient symmetry breaking scheme can be achieved if we can solve the dominance detection problem in polynomial time. Therefore, we studied the complexity of dominance detection between arbitrary assignments for combined variable and value symmetry. For CSPs with partially symmetric single-valued variables and partially symmetric values we developed an efficient dominance detection algorithm that shaves the search-tree to a point where no two search-nodes can be symmetrical to each other. In contrast to these positive tractability results, for set-CSPs with symmetric variables and symmetric values, we found that the dominance detection problem is NP-hard.

Our algorithms are based on the structural abstraction of a given CSP, a method that we refer to as *structural symmetry breaking (SSB)*. When only the structure of variables and values that partition into whole equivalence classes of pairwise symmetric elements is considered, the work presented here effectively closes the outstanding complexity questions regarding SBDD. Further research will have to regard CSPs

whose symmetry structures are more refined. Consider for example the case where the variables form a ring structure and all rotations on the ring yield to an equivalent CSP. Under what conditions can we break value and variable symmetry when the permutations on the CSP elements form structures that are different from partitions?

In our view, it is perceivable that structural symmetries could be derived automatically by a static analysis of a given constraint program if the modeling language used provided semantics regarding symmetries. For instance, an all-different constraint could provide the information that all variables and values are treated as symmetric by this constraint. Then, built-in efficient dominance detection algorithms like the one that we developed can be used efficiently to break the symmetries that were derived automatically, thus making symmetry breaking an effective and efficient component of constraint programming that is totally seamless for the user.

Acknowledgments

Many thanks to Pierre Flener and Justin Pearson for helping tremendously in polishing up the presentation of this work!

References

- [Ahuja *et al.*, 1993] R. Ahuja, T. Magnati, J. Orlin. *Network Flows*. Prentice Hall, 1993.
- [Barnier and Brisset, 2002] N. Barnier and P. Brisset. Solving the Kirkman’s schoolgirl problem in a few seconds. *Proc. CP’02*, 477–491, 2002.
- [Brown *et al.*, 1988] C. Brown, L. Finkelstein, P. Purdom Jr. Back-track searching in the presence of symmetry. *Proc. AAEC-6*, 99–110, 1988.
- [Crawford *et al.*, 1996] J. Crawford, M. Ginsberg, E. Luks, A. Roy. Symmetry-breaking predicates for search problems. *Proc. KR’96*, 149–159, 1996.
- [Fahle *et al.*, 2001] T. Fahle, S. Schamberger, M. Sellmann. Symmetry Breaking. *Proc. CP’01*, 93–107, 2001.
- [Flener *et al.*, 2002] P. Flener, A. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, J. Pearson, T. Walsh. Breaking row and column symmetries in matrix models. *Proc. CP’02*, 462–476, 2002.
- [Focacci and Milano, 2001] F. Focacci and M. Milano. Global cut framework for removing symmetries. *Proc. CP’01*, 77–92, 2001.
- [Gent *et al.*, 2003] I. Gent, W. Harvey, T. Kelsey, S. Linton. Generic SBDD using computational group theory. *Proc. CP’03*, 333–347, 2003.
- [Gent and Smith, 2000] I. Gent, B. Smith. Symmetry breaking in constraint programming. *Proc. ECAI’00*, 599–603, 2000.
- [Puget, 2002] J.-F. Puget. Symmetry breaking revisited. *Proc. CP’02*, 446–461, 2002.
- [Roney-Dougal *et al.*, 2004] C. Roney-Dougal, I. Gent, T. Kelsey, S. Linton. Tractable symmetry breaking using restricted search trees. *Proceedings of ECAI’04*, 2004.
- [Van Hentenryck *et al.*, 2003] P. Van Hentenryck, P. Flener, J. Pearson, M. Agren. Tractable symmetry breaking for CSPs with interchangeable values. *Proc. IJCAI’03*, 2003.