Structural Symmetry Breaking for Constraint Satisfaction Problems

Pierre Flener[1], Justin Pearson[1], Meinolf Sellmann[2], Pascal Van Hentenryck[2], and Magnus Ågren[1] *

[1] Department of Information Technology
Uppsala University, Box 337, SE – 751 05 Uppsala, Sweden
{pierref,justin,agren}@it.uu.se

[2] Department of Computer Science
Brown University, Box 1910, Providence, RI 02912, USA
{sello,pvh}@cs.brown.edu

Abstract. In recent years, symmetry breaking for constraint satisfaction problems (CSPs) has attracted considerable attention. Various general schemes have been proposed to eliminate symmetries. In general, these schemes may take exponential space or time to eliminate all the symmetries. We identify several classes of CSPs that encompass many practical problems and for which symmetry breaking for various forms of value or variable interchangeability is tractable using dedicated search procedures. We also show the limits of efficient symmetry breaking for such dominance-detection schemes by proving intractability results for some classes of CSPs.

1 Introduction

Many constraint satisfaction problems (CSPs) naturally exhibit symmetries. Symmetry breaking may drastically improve performance [3, 17, 20, 25]. An important contribution in this area has been the development of various general schemes for symmetry breaking *during* search in CSPs (e.g., SBDS [2, 15] and SBDD [9, 13, 20], the latter being described briefly in Section 3). Unfortunately, in general, these schemes may require exponential resources to break all the symmetries. Indeed, some schemes may require exponential space to store all the nogoods generated through symmetries, while others may take exponential time to discover whether a partial assignment is symmetric to one of the existing nogoods. As a consequence, practical applications often place limits on how many nogoods can be stored and/or which symmetries to break. Other than eliminating symmetries by re-modelling the problem (see, e.g., [24]), another important approach is to break symmetries by adding constraints *before* search starts (e.g., [8, 19]). Unfortunately, in general, a super-exponential number of constraints may be needed to break all the symmetries. For instance, the lex-leader scheme of [8] adds one constraint per symmetry, but the number of symmetries is often

* The authors' names are ordered according to the Swedish alphabet.

super-exponential (an $m \times n$ matrix with fully interchangeable rows and columns has $m! \cdot n!$ symmetries). As a consequence, practical applications often add only some of these symmetry-breaking constraints (see, e.g., [11, 23]).

We approach symmetry breaking from a different, orthogonal standpoint. *Our goal is to identify classes of CSPs that are practically relevant and for which symmetry breaking is tractable, i.e., polynomial in time and space, using dedicated search procedures.* We identify several such classes whose CSPs feature various forms of value or variable interchangeability and encompass many practical problems. For some of them, symmetry breaking can even be performed with a *constant* overhead with respect to both time and space at every node explored. We also introduce the new notions of existential and abstract nogoods, which are used to derive the results for some of the CSP classes. We believe that these notions are helpful to derive many other classes of tractable symmetries. As such, this paper should be viewed only as a first step in this fascinating area. Finally, we also show the limits of efficient symmetry breaking for dominance-detection schemes like SBDD by proving intractability results for certain classes of CSPs.

It is useful to contrast our approach with the research avenue pioneered by Freuder [14] on value interchangeability. He also introduced various forms of value interchangeability. However, his goal was to *discover* symmetries inside CSPs and to remove them through a preprocessing reformulation. Unfortunately, discovering symmetries in CSPs is not tractable for many interesting classes of CSPs. *This paper, in contrast, assumes that the symmetries in a CSP are known.* It focuses on how to exploit this knowledge *during search* to break symmetries efficiently. In [28], we address the companion issue of how to automatically detect symmetries in CSP models.

Example 1. Consider the scene allocation problem featured in [26]. It aims at producing a movie (or a series) at minimal cost by deciding when to shoot which scenes. Each scene involves a number of actors and at most five scenes a day can be shot. All the actors of a scene must be present on the day the scene is shot. The actors have fees representing the amount to be paid per day they spend in the studio. An optimal solution can be modelled as an assignment of scenes to days that minimizes the production costs. The exact days assigned to the scenes have no importance and are fully interchangeable. What is important is how the scenes are clustered. In fact, the original problem formulation only has a *number*, say n, of days. It is the often necessary *naming* of these days while modelling the problem, say as $1 \dots n$, that induces these symmetries. Our approach does not aim at discovering this fact; it rather focuses on how to exploit it to break the symmetries it induces.

This theoretical paper, which unites and extends[3] our work published in [27, 22], is structured as follows. First, in Section 2, we define CSPs and assignments

[3] Sections 5.3 and 7.2 and the epilogue to Corollary 2 are new, while Section 5.2 was generalized. The originally omitted proofs of Propositions 1 and 2, Lemmas 3 and 5, and Theorems 3 and 4 are now provided, while the proofs of Theorem 1 and Corollary 1 were expanded into greater detail.

in a non-standard way that gives rise to elegant formulations and proofs of our results. Then, in Sections 3 to 7, we formally establish those results, for various forms of value and/or variable interchangeability. Finally, Section 8 summarizes the results and concludes this paper.

2 Preliminaries

Our definition of constraint satisfaction problems (CSPs), although it captures their informal meaning, is non-standard but simplifies the proofs and other definitions considerably. The basic idea is that the set of constraints of a CSP is abstracted by a Boolean function that returns **true** if all these constraints are satisfied. We are not interested in the constraint structure. Solutions are then also represented as functions, namely from the variables to the possible values.

Definition 1 (CSP, Assignment, Solution). *A* constraint satisfaction problem *(*CSP*) is a triplet $\langle V, D, C\rangle$, where V denotes the set of variables, D denotes the set of possible values for these variables and is called their* domain*, and $C : (V \to D) \to Bool$ is a constraint that specifies which assignments of values to the variables are solutions. An* assignment *for a CSP $\mathcal{P} = \langle V, D, C\rangle$ is a function $\alpha : V \to D$. If the domain D is the power-set of some other set, called the* universe*, we say that the CSP is a* set-CSP *and has* set variables*, while we call α a* set assignment*; otherwise, we say that the CSP has* scalar variables*. A* solution *to a CSP $\mathcal{P} = \langle V, D, C\rangle$ is an assignment σ for $\mathcal{P}$ such that $C(\sigma) =$ **true**. The set of all the solutions to a CSP $\mathcal{P}$ is denoted by $Sol(\mathcal{P})$.*

Algorithms to solve CSPs manipulate partial assignments. It is often important to reason about which variables are already assigned (the *scope* of the partial assignment) and the set of values they are assigned to (the *image* of the partial assignment).

Definition 2 (Partial Assignment, Scope, Image). *A* partial assignment *for a CSP $\mathcal{P} = \langle V, D, C\rangle$ is a function $\alpha : W \to D$, where $W \subseteq V$. The* scope *of α, denoted by $scope(\alpha)$, is W. The* image *of α, denoted by $image(\alpha)$, is the set $\{\alpha(v) \mid v \in scope(\alpha)\}$. For each value $d \in image(\alpha)$, we use $\alpha^{-1}(d)$ to denote the set $\{v \mid v \in scope(\alpha)\ \ \&\ \ \alpha(v) = d\}$. We denote the empty partial assignment by ϵ.*

Note that every assignment and solution to a CSP $\mathcal{P} = \langle V, D, C\rangle$ is a partial assignment for $\mathcal{P}$, with scope V. We often denote a partial assignment α by a conjunction of equations, and then see it as a constraint:

$$v_{i_1} = \alpha(v_{i_1})\ \ \&\ \ \cdots\ \ \&\ \ v_{i_k} = \alpha(v_{i_k})$$

where $scope(\alpha) = \{v_{i_1}, \ldots, v_{i_k}\}$.

Example 2. The partial assignment $v_1 = 1\ \ \&\ \ v_2 = 2\ \ \&\ \ v_3 = 3$ represents the function whose scope is $\{v_1, v_2, v_3\}$ and that assigns the value i to v_i.

Definition 3 (Extension of a Partial Assignment). *A partial assignment θ for a CSP $\mathcal{P}$ is an* extension *of a partial assignment α for $\mathcal{P}$ if $scope(\alpha) \subseteq scope(\theta)$ and $\forall v \in scope(\alpha) : \theta(v) = \alpha(v)$.*

Definition 4 (Completion of a Partial Assignment). *A* completion *of a partial assignment α for a CSP $\mathcal{P} = \langle V, D, C \rangle$ is an extension θ of α with $scope(\theta) = V$. The set of all the completions of α for $\mathcal{P}$ is denoted by $Comp(\alpha, \mathcal{P})$.*

Note that the set of all the completions of a solution σ is the singleton $\{\sigma\}$.

Definition 5 (Nogood). *A* nogood *for a CSP $\mathcal{P}$ is a partial assignment α for $\mathcal{P}$ that cannot be extended into a solution, that is $Comp(\alpha, \mathcal{P}) \cap Sol(\mathcal{P}) = \emptyset$.*

The idea behind the noun 'nogood' is that no partial assignment should ever extend any previously identified nogood.

Definition 6 (Violating a Nogood). *A partial assignment θ for a CSP $\mathcal{P}$* violates *a nogood α for $\mathcal{P}$ if θ is an extension of α.*

The verb 'violates' is justified here, as we also view a partial assignment, and hence a nogood, as a constraint, with the form of a conjunction of equations. Strictly speaking, this notion of nogood violation is redundant with the notion of nogood extension, but we keep it for its more intuitive appeal.

Any extension of a nogood is itself a nogood:

Proposition 1. *If a partial assignment θ for a CSP $\mathcal{P}$ violates a nogood for $\mathcal{P}$, then θ is itself a nogood for $\mathcal{P}$.*

Proof. Assume that a partial assignment θ for $\mathcal{P}$ violates a nogood α for $\mathcal{P}$ and assume that θ is not a nogood for $\mathcal{P}$. Then there exists a completion γ of θ such that $\gamma \in Sol(\mathcal{P})$. Since γ is a completion of θ and θ is an extension of α, we have, by transitivity of $\subseteq$ and $=$, that γ is a completion of α. But then α cannot be a nogood, since $\gamma \in Sol(\mathcal{P})$. This is a contradiction, so θ must be a nogood. □

Furthermore, a partial assignment that can only be extended into a nogood is itself a nogood:

Proposition 2. *Let $\mathcal{P} = \langle V, D, C \rangle$ be a CSP where $D = \{d_1, \ldots, d_m\}$. Let α be a partial assignment for $\mathcal{P}$ with $scope(\alpha) = \{v_{i_1}, \ldots, v_{i_k}\}$. If every $\alpha \ \& \ v_{i_{k+1}} = d_i$ $(1 \leq i \leq m)$ is a nogood for $\mathcal{P}$, then α is itself a nogood for $\mathcal{P}$.*

Proof. Assume that α is not a nogood for $\mathcal{P}$. Then there exists an extension γ of α such that $\gamma \in Sol(\mathcal{P})$. But γ must include $\alpha \ \& \ v_{i_{k+1}} = d_i$ for some $1 \leq i \leq m$. But then γ is also an extension of $\alpha \ \& \ v_{i_{k+1}} = d_i$ for that i. But $\alpha \ \& \ v_{i_{k+1}} = d_i$ is a nogood for every $1 \leq i \leq m$, so there cannot exist such a γ. Hence α is a nogood for $\mathcal{P}$. □

In other words, nogoods can be lifted from the children to their parent in a search tree: when all the child nodes have been explored, their nogoods can be forgotten and only the parent nogood needs to be kept.

With respect to the symmetry considered in this paper, in [6], two definitions of symmetry are presented: solution symmetries, which are essentially bijections on the set of variable-value pairs that make up assignments and preserve solutions; and constraint symmetries, which are bijections on the structure of the constraints in the problem. It is shown that the group of constraint symmetries of a CSP is a subgroup (most often strict) of the group of solution symmetries. In this paper, the symmetries are defined as subgroups of the set of solution symmetries without reference to the constraint symmetries. Specific definitions of the particular symmetry considered are given in the respective parts of the paper.

3 Structural Symmetry Breaking for Variable and Value Symmetry

We start our investigation by showing that there exists an efficient symmetry-breaking algorithm for constraint satisfaction problems where both the set of values and the set of variables can be partitioned into subsets such that, within each subset, all variables or values, respectively, are interchangeable. We call these problems piecewise value- and variable-interchangeable CSPs:

Definition 7 (Piecewise Bijection). *Let $S = \cup_i P_i$ such that the sets P_i are disjoint, i.e., $P_i \cap P_j \neq \emptyset$ implies $i = j$. Then, we write $S = \sum_i P_i$ and call $\sum_i P_i$ a* partition *of S. A bijection $b : S \to S$ is a* piecewise bijection *over $\sum_i P_i$ if and only if $b(P_i) = P_i$, where $b(P_i) = \{b(e) \mid e \in P_i\}$.*

Definition 8 (Piecewise Interchangeable CSP). *A CSP $\mathcal{P} = \langle \sum_k V_k, \sum_l D_l, C \rangle$ is a* piecewise interchangeable CSP *if and only if, for each solution $\sigma \in Sol(\mathcal{P})$, each piecewise bijection a over $\sum_k V_k$, and each piecewise bijection b over $\sum_l D_l$, we have $b \circ \sigma \circ a \in Sol(\mathcal{P})$. If the only piecewise bijection over $\sum_k V_k$ (or $\sum_l D_l$) is the identity, then the CSP is* piecewise value-interchangeable (or variable-interchangeable).

We will show how to break all symmetry in piecewise interchangeable CSPs by means of Symmetry Breaking by Dominance Detection (SBDD) [9, 13]. SBDD is a technique to break symmetries during search. The idea is as follows: At any given choice point during search, we check whether the subtree rooted at the current node maps, under application of symmetry, into another subtree that has been fully explored earlier. If so, then the current node need not be investigated further and can be pruned. Different ways to control and limit the number of previously expanded subtrees that must be checked against have been developed in [9, 13]. With these results, the core procedure of any SBDD code that determines its efficiency is the dominance detection algorithm that checks

whether a given partial (set) assignment is dominated by another one. Formally, we define:

Definition 9 (Dominating an Assignment). *Let $\mathcal{P} = \langle \sum_k V_k, \sum_l D_l, C \rangle$ be a piecewise interchangeable CSP. Assignment α* dominates *assignment β if and only if there exist piecewise bijections a over $\sum_k V_k$ and b over $\sum_l D_l$ such that for every $v \in scope(\alpha)$ we have $\beta(a(v)) = b(\alpha(v))$.*

Given two assignments α and β for a piecewise interchangeable CSP, we call the problem of determining whether α dominates β the *dominance detection problem*. Consequently, if we can solve the dominance detection problem efficiently, then we can also break symmetries efficiently.

The key idea to tackle the dominance detection problem for piecewise interchangeable CSPs consists in the introduction of structural abstractions: to model a CSP, we need to uniquely label each value and each variable with a name. This is, of course, problematic when certain variables and certain values are actually interchangeable. 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 V_k and D_l. When assignments are committed, though, some of those initial symmetries are broken. Then, in order to check which CSP objects are still interchangeable, we need to introduce subclasses of the original symmetry classes. We will see that we can detect the remaining symmetries by labelling each of those subclasses with an appropriate 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.

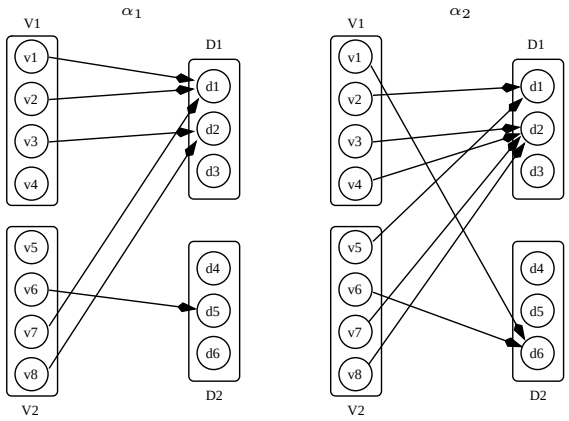
3.1 Signatures

First consider the following example.

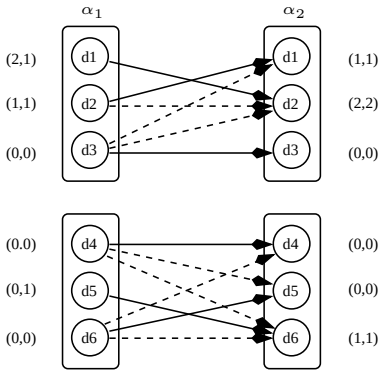
Example 3. Take variables $V = \{v_1, \ldots, v_8\}$ over the domain $D = \{d_1, \ldots, d_6\}$. Assume that the first four and the last four variables are interchangeable: $V_1 = \{v_1, \ldots, v_4\}$ and $V_2 = \{v_5, \ldots, v_8\}$. Assume that the first three and the last three values are interchangeable: $D_1 = \{d_1, \ldots, d_3\}$ and $D_2 = \{d_4, \ldots, d_6\}$. Consider the following two partial assignments (see Figure 1(a)): $\alpha_1 = (v_1 = d_1 \ \& \ v_2 = d_1 \ \& \ v_3 = d_2 \ \& \ v_6 = d_5 \ \& \ v_7 = d_1 \ \& \ v_8 = d_2)$ and $\alpha_2 = (v_1 = d_6 \ \& \ v_2 = d_1 \ \& \ v_3 = d_2 \ \& \ v_4 = d_2 \ \& \ v_5 = d_1 \ \& \ v_6 = d_6 \ \& \ v_7 = d_2 \ \& \ v_8 = d_2)$. When looking at α_1, we see that:

1. There is one value (namely d_1) in D_1 that is taken by two variables in V_1 and one variable in V_2.
2. There is one value (namely d_2) in D_1 that is taken by one variable in V_1 and one variable in V_2.
3. There is one value (namely d_5) in D_2 that is taken by one variable in V_2.

On the other hand, in α_2, we see that:



(a)



(b)

Fig. 1. Part (a) illustrates assignments α_1 and α_2. Part (b) gives the signatures for each value, links pairs of values where the one in assignment α_1 dominates the one in α_2, and designates by solid lines a perfect matching that proves that α_1 dominates α_2.

I. There is one value (namely d_2) in D_1 that is taken by two variables in V_1 and two variables in V_2.
II. There is one value (namely d_1) in D_1 that is taken by one variable in V_1 and one variable in V_2.
III. There is one value (namely d_6) in D_2 that is taken by one variable in V_1 and one variable in V_2.

Lining up 1-I ($d_1 \mapsto d_2$, $\{v_1, v_2\} \mapsto \{v_3, v_4\}$, $\{v_7\} \mapsto \{v_7, v_8\}$), 2-II ($d_2 \mapsto d_1$, $\{v_3\} \mapsto \{v_2\}$, $\{v_8\} \mapsto \{v_5\}$), and 3-III ($d_5 \mapsto d_6$, $\{v_6\} \mapsto \{v_6\}$), we see that α_2 is structurally a partial assignment extended from α_1, or, in other words, that α_1 dominates α_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 the variables and values to the actual structure of

the problem. That is, instead of talking about specific variables and values, we have considered members of classes. Specifically, for each partial 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 α_1, the value d_1 has the signature $(2 \times V_1, 1 \times V_2)$, or, in shorter writing, the signature of d_1 under α_1 is $sig_{\alpha_1}(d_1) = (2, 1)$. Under α_2, on the other hand, the signature of d_2 is $sig_{\alpha_2}(d_2) = (2, 2)$. Consequently, d_2 in α_2 can be viewed as more specialized than d_1 in α_1, or one may also say that d_1 in α_1 dominates d_2 in α_2. In this terminology, d_1 in α_2 has signature $sig_{\alpha_2}(d_1) = (1, 1)$ and therefore dominates d_1 in α_1. Note that $sig_{\alpha_2}(d_6)$ is also $(1, 1)$, but that d_6 in α_2 does not dominate d_1 in α_1 since $d_6 \in D_2$ whereas $d_1 \in D_1$. In general:

Definition 10 (Dominating a Value). *A value d in a partial assignment α* dominates *a value e in a partial assignment β if and only if d and e belong to the same value-symmetry class and $sig_\alpha(d) \leq sig_\beta(e)$.*[4]

A value d in a partial assignment α is structurally equivalent *to a value e in a partial assignment β if and only if d and e belong to the same value-symmetry class and $sig_\alpha(d) = sig_\beta(e)$.*

In the following sub-section, we will show how these notions of dominance and structural equivalence can be exploited to devise a polynomial-time algorithm that solves the dominance detection problem on piecewise interchangeable CSPs.

3.2 Dominance Detection Using Signatures

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

Lemma 1. *A partial assignment α dominates another partial assignment β in a piecewise interchangeable CSP if and only if there exists a piecewise bijection b over $D = \sum_l D_l$ such that d in α dominates $b(d)$ in β for every $d \in D$.*

Proof. First, assume that α dominates β. Then, there exist piecewise bijections a over $\sum_k V_k$ and b over $\sum_l D_l$ such that for every $v \in scope(\alpha)$ we have $\beta(a(v)) = b(\alpha(v))$. Since both v and $a(v)$ belong to the same symmetry class, we have $sig_\alpha(d) \leq sig_\beta(b(d))$ for all values $d \in D$, which is the same as to say that d in α dominates $b(d)$ in β.

Second, assume that there exists a piecewise bijection b over $\sum_l D_l$ such that $sig_\alpha(d) \leq sig_\beta(b(d))$ for every $d \in D$. Then, since each variable is assigned to at most one value, there exists a piecewise bijection a over $\sum_k V_k$ such that $\beta(a(v)) = b(\alpha(v))$ for every $v \in scope(\alpha)$. Thus, we have that α dominates β. □

[4] The $\leq$-relation on vectors is defined as the usual component-wise comparison that yields the so-called dominance ordering, which is different from a lexicographic ordering.

Consequently, we have that α dominates β if and only if there exists a perfect matching in a bipartite graph where the edges are defined by the signature relation of values (see Figure 1(b)). Let us denote by D' a set of duplicates of the values in D obtained by appending a prime sign to their names (that is, $D' := \{d' \mid d \in D\}$).

Definition 11 (Dominance Detection Graph). *Given two partial assignments α and β, the* dominance detection graph $DDG(\alpha, \beta)$ *is $(D \cup D', E)$, where $E := \{(d, e') \mid d$ in α dominates e in $\beta\}$ denotes the set of arcs.*

Theorem 1. *The dominance detection problem between two partial assignments α and β for a piecewise interchangeable CSP 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 $DDG(\alpha, \beta)$, with m being the number of values and n the number of variables. Hence all the value and variable symmetries of a piecewise interchangeable CSP can be broken with a polynomial time overhead at every node explored.*

Proof. 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 $DDG(\alpha, \beta)$. The additional complexity denoted in the theorem is due to the necessity of constructing $DDG(\alpha, \beta)$ first. 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. Instead of writing down entire signatures, for each value we hold a sparse list that only contains the non-zero entries of a signature, together with the information to which variable partition an entry in the sparse list belongs. To set up this sparse representation, we first order the variable instantiations in a given partial assignment according to the partition that the corresponding variable belongs to. This can be done in time linear in the number of variables, since this is also the maximum number of symmetry classes that can exist. In this order, we now scan through the partial assignments and set up the sparse signatures. Then, we iterate through the signatures of all the values in α and compare them with all the signatures of the values in β. With the sparse representation of signatures, this takes time $O(m(|\alpha| + |\beta|))$. □

Interestingly, it can also be shown that *every* bipartite graph can also be viewed as a dominance detection graph of a CSP and assignments α and β that can be determined in time linear in the size of the given graph. Therefore, a perfect bipartite matching exists if and only if α dominates β, 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 $T \in \Omega(M) \cap O(M + m^2 + mn)$.

In [12] we have provided a static counterpart of the here considered dynamic structural symmetry breaking for piecewise interchangeable CSPs, that is we have exploited the concept of signature to devise a set of symmetry-breaking constraints that break all the considered symmetries.

Theorem 1 trivially has two interesting consequences. First, dropping the assumed piecewise *value* interchangeability or tightening the assumed *piecewise* interchangeabilities into *full* interchangeabilities will not worsen its tractability result, hence all the symmetries of fully or piecewise *variable*-interchangeable CSPs and of *fully* value- and variable-interchangeable CSPs can be broken with a polynomial time overhead at every node explored. Conversely, when dropping the assumed piecewise *variable* interchangeability, we achieve tractability for the symmetries of fully and piecewise *value*-interchangeable CSPs. We will study these special cases later where we will devise highly efficient symmetry breaking methods that do not require complex matchings to be solved and that minimize the computational overhead needed for symmetry breaking in these special cases.

Second, dropping the assumed piecewise value interchangeability and switching to set-CSPs will not worsen the tractability result. Indeed, set variables that take subsets of a universe of *non*-interchangeable values can be seen as scalar variables that take scalar values from a domain of *non*-interchangeable values, hence the tractability results of symmetry breaking are those for fully or piecewise variable interchangeability of (scalar) CSPs: all the symmetries of fully or piecewise *variable*-interchangeable set-CSPs can be broken with a polynomial time overhead at every node explored.

4 Symmetry-Based Filtering

With Theorem 1, we can break all the symmetries of a piecewise interchangeable CSP in polynomial time when using a symmetry-breaking by dominance detection (SBDD) approach [9, 13]. What is annoying in this setting is that we still have to check at every choice point to see whether it is not dominated by one that was previously expanded, that is 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 for pruning.[5] A brute-force approach could 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 the domains of variables accordingly. The other, fundamentally different type of "filtering" consists in the creation of children that are also not symmetric to each other. Both types need to be addressed to achieve a symmetry-free search tree (which corresponds

[5] With 'filtering', we refer to the idea of domain reduction in constraint programming, whereas with 'pruning', we refer to the detection of a sufficient reason for backtracking.

to the GE-trees in [21]). We distinguish the two types of filtering by naming them differently: *symmetric-ancestor based filtering* and *symmetric-sibling based filtering*.

4.1 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 12 (Ancestor-Symmetry Resistance). *Given a depth-first-search tree T, we say that a choice point β (associated with its homonymous partial assignment β that captures previously committed unary branching decisions) is* ancestor-symmetry resistant *if and only if for all previously fully expanded nodes $\alpha \in T$ (where α is called an* ancestor *of β) and for all variables v and values $d \in Dom(v)$ we have that α does not dominate β & $(v = d)$.*

Assume that we are currently investigating choice point β and that α is some ancestor node that does not dominate β. Observe that instantiating one more variable $v \in V_k$ for some k by setting $v \mapsto e \in D_l$ for some l will change only the signature of e from $sig_\beta(e)$ to $sig_\beta(e)+e_k$, where e_k denotes the unit vector with a 1 in the kth component. We set $\beta' := \beta$ & $(v = e)$. Then, $G_1 := DDG(\alpha, \beta)$ and $G_2 := DDG(\alpha, \beta')$ only differ in that the latter bipartite graph may contain some additional edges that must all be incident to e' in the right partition. If G_2 contains an m-matching, this matching must contain exactly one of those additional edges. Consequently, if α dominates β', then G_1 must contain an $m-1$-matching. Only if this is the case, work needs to be done to make β ancestor-symmetry resistant with respect to α.

So let us assume that G_1 contains an $m-1$-matching. Provided with that matching, using some straightforward 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 (for an introduction to matching theory we refer to [1]): a matching can be viewed as a flow in some network that closely corresponds to the bipartite graph. We consider the usual residual network with respect to that flow that has an additional source node s that connects to all nodes in the first partition, and a sink node t that is connected from all nodes in the second partition. The capacities of the residual network are given by the the residual capacity of edges after the flow has been routed, including reverse edges for edges with positive flow.

Then, a maximum matching (corresponding to a maximum flow) defines two cuts. The first is given by the nodes that are reachable from the source in the residual network. If we denote this set with S, then (S, S^C) is an s-t-cut. The second cut is given by the set of nodes from which the sink is reachable in the residual network. If we denote this set with T, then (T^C, T) is also an s-t-cut. Note that both cuts can be computed in time linear in the size of the given networks after the maximum matching has been computed.

Now, the core observation is that, given those two cuts, the critical edges are exactly those that run from $S \cap D$ to $T \cap D'$: clearly, adding such an edge yields an improving path in the residual network and therefore an m-matching. On the other hand, note that $T \subseteq S^C$ and $S \subseteq T^C$. Any edge added from S to $S^C \setminus T$ would leave the cut (T^C, T) untouched, which proves that no such an edge can improve the matching. Edges that run from $T^C \setminus S$ to T follow analogously.

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 that run between nodes d and e' with $d, e \in D_l$ for some l and for which there exists k such that $sig_\alpha(d) \leq sig_\beta(e) + e_k$. If and only if we find such a pair of nodes, a single extra assignment added to β will result in a successful dominance detection. Precisely, every assignment of e to a previously unassigned variable $v \in V_k$ will result in a dominated choice point. Thus, if we remove e from the domain of v for every unassigned $v \in V_k$, we keep the unique parts of the search space and we never produce any choice points that are symmetric to one that was expanded previously to β.

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

Theorem 2. *For a piecewise interchangeable CSP, we can achieve ancestor-symmetry resistance for a given search node in time* $O(nm^{3.5} + n^2m^2)$.

4.2 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 choosing the next variable to be assigned, but before branching 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 that, 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 one, they both must already be structurally equivalent (see Definition 10), we can avoid producing symmetric siblings by choosing exactly one representative value 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 piecewise interchangeable CSPs 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 Fast Algorithms to Break Value Symmetry

In this section, we review the special case of piecewise interchangeable CSPs with no variable symmetry, which we call piecewise value-interchangeable CSPs. With the previous results, we know already that symmetry breaking can be achieved in polynomial time. Now, we focus on the development of algorithms that break value symmetry with minimal overhead. First, in Section 5.1, we describe our new approach in full detail on the class of fully value-interchangeable CSPs, showing how it leads to the known result [16] that all their value symmetries can be broken by a dedicated search procedure with a *constant* overhead with respect to both time and space at every node explored (Theorem 5). Then, in Section 5.2, we show that this result actually generalises to *piecewise* value-interchangeable CSPs (Theorem 6). Finally, in Section 5.3, we show, again in full detail, that the same result even holds for fully and piecewise value-interchangeable *set*-CSPs (Theorems 7 and 8, respectively).

5.1 Fully Value-Interchangeable CSPs

When all values are interchangeable and no variable symmetry is present, we speak of a fully value-interchangeable CSP.

Definition 13 (Fully Value-Interchangeable CSP). *A CSP $\mathcal{P} = \langle V, D, C\rangle$ is a* fully value-interchangeable CSP *if, for each solution $\sigma \in Sol(\mathcal{P})$ and each bijection b over D, we have $b \circ \sigma \in Sol(\mathcal{P})$.*

In the following, we show that in this case symmetry breaking can be performed with *constant* overhead with respect to both time and space at every node explored. Our method is based on nogoods.

The following theorem gives a fundamental characterization of nogoods for fully value-interchangeable CSPs. It states that nogoods are preserved under value interchanges:

Theorem 3. *Let α be a nogood for a fully value-interchangeable $\mathcal{P} = \langle V, D, C\rangle$ and let $b : D \to D$ be a bijection. Then $b \circ \alpha$ is a nogood for $\mathcal{P}$.*

Proof. Let g be a completion of $b \circ \alpha$ and assume that $g \in Sol(\mathcal{P})$. Since b is a bijection, we have $b^{-1} \circ g \in Sol(\mathcal{P})$. But $(b^{-1} \circ g)(v) = (b^{-1} \circ (b \circ \alpha))(v) = \alpha(v)$, for all $v \in scope(\alpha)$, by the definition of a completion, that is α can be extended into a solution. This contradicts the fact that α is a nogood. Hence $b \circ \alpha$ cannot be extended into a solution and is thus actually a nogood. □

The closure of a nogood α for a fully value-interchangeable CSP is the set of nogoods obtained from α by applying each possible value interchange, or value symmetry, to α:

Definition 14 (Closure of a Nogood). *Let α be a nogood for a fully value-interchangeable $\mathcal{P} = \langle V, D, C\rangle$. The* closure *of α for $\mathcal{P}$, denoted by $Closure(\alpha, \mathcal{P})$, is the set $\{b \circ \alpha \mid b$ is a bijection over $D\}$.*

The main idea of our approach is to try and abstract such closures of nogoods, so that their representation takes polynomial space and that membership to a closure can be tested during search in polynomial time. Using partial evaluation, it will then become possible to write a search procedure that breaks all the value symmetries by never extending any member of the closures of all the nogoods generated during search.

Existential Nogoods and Abstract Nogoods. We now show that the closure of a nogood for a fully value-interchangeable CSP can be characterized compactly. We first introduce the concept of existential nogood, which simplifies the proofs and intuitions. A nogood can be generalized by introducing distinct existentially quantified variables for each value in its image. This can be seen as inverse Skolemization.

Definition 15 (Existential Nogood). *Let α be a nogood for a fully value-interchangeable $\mathcal{P} = \langle V, D, C\rangle$, with $image(\alpha) = \{d_1, \ldots, d_k\}$. The* existential nogood *of α for $\mathcal{P}$, denoted by $Enogood(\alpha, \mathcal{P})$, is the set of all functions $\gamma : scope(\alpha) \to D$ satisfying the condition*

$$\exists e_1, \ldots, e_k \in D : \forall i \in 1 \ldots k : \forall v_j \in \alpha^{-1}(d_i) : \gamma(v_j) = e_i \ \ \& \ \ alldiff(e_1, \ldots, e_k)$$

where $alldiff(a_1, \ldots, a_n)$ holds if all the a_i are different values.

By abuse of language, we identify an existential nogood, which is a set of functions, with the condition that its members have to satisfy.

Example 4. Consider a nogood β, written as a conjunction of equations:

$$\beta(v_1) = 1 \ \ \& \ \ \beta(v_2) = 2 \ \ \& \ \ \beta(v_3) = 3 \ \ \& \ \ \beta(v_4) = 1 \ \ \& \ \ \beta(v_5) = 2$$

The existential nogood of β is the condition

$$\begin{array}{c} \exists e_1, e_2, e_3 \in D : \gamma(v_1) = e_1 \ \ \& \ \ \gamma(v_2) = e_2 \ \ \& \ \ \gamma(v_3) = e_3 \\ \quad \& \ \ \gamma(v_4) = e_1 \ \ \& \ \ \gamma(v_5) = e_2 \ \ \& \ \ alldiff(e_1, e_2, e_3) \end{array}$$

or, more precisely, the set of functions $\gamma : scope(\beta) \to D$ satisfying this condition.

The following lemma indicates that an existential nogood precisely captures the closure of its nogood:

Lemma 2. *Let α be a nogood for a fully value-interchangeable CSP $\mathcal{P}$: $Enogood(\alpha, \mathcal{P}) = Closure(\alpha, \mathcal{P})$.*

Proof. Let $image(\alpha) = \{d_1, \ldots, d_k\}$. By definition of the image, we have that

$$\forall i \in 1 \ldots k : \forall v_j \in \alpha^{-1}(d_i) : \alpha(v_j) = d_i \ \ \& \ \ alldiff(d_1, \ldots, d_k).$$

We first show that $Closure(\alpha, \mathcal{P}) \subseteq Enogood(\alpha, \mathcal{P})$. Let $\gamma \in Closure(\alpha, \mathcal{P})$. This means that there exists a bijection b such that $\gamma = b \circ \alpha$. Thus, γ satisfies

$$\forall i \in 1 \ldots k : \forall v_j \in \alpha^{-1}(d_i) : \gamma(v_j) = b(d_i) \ \ \& \ \ alldiff(b(d_1), \ldots, b(d_k))$$

by definition of a bijection, and hence $\gamma \in \mathit{Enogood}(\alpha, \mathcal{P})$.

We now show that $\mathit{Enogood}(\alpha, \mathcal{P}) \subseteq \mathit{Closure}(\alpha, \mathcal{P})$. Let $\delta \in \mathit{Enogood}(\alpha, \mathcal{P})$. Then there exist some values $a_1, \ldots, a_k \in D$ such that

$$\forall i \in 1 \ldots k : \forall v_j \in \alpha^{-1}(d_i) : \delta(v_j) = a_i \;\; \& \;\; \mathit{alldiff}(a_1, \ldots, a_k).$$

Since $a_1, \ldots, a_k$ are all different, there exists a bijection b satisfying $\forall i \in 1 \ldots k : b(d_i) = a_i$. Hence δ can be rewritten as $b \circ \alpha$ and $\delta \in \mathit{Closure}(\alpha, \mathcal{P})$. □

It is not obvious that membership to an existential nogood can be tested efficiently since it involves an existential quantification. However, due to the nature of the underlying conditions, it is possible to eliminate the existential variables by equating all the universally quantified variables that have the same value and by selecting, for the *alldiff* condition, a representative variable v_{r_i} for each set $\alpha^{-1}(d_i)$. This is precisely the motivation for the concept of abstract nogoods, defined next:

Definition 16 (Abstract Nogood). *Let α be a nogood for a fully value-interchangeable $\mathcal{P} = \langle V, D, C \rangle$. Let $\mathit{image}(\alpha) = \{d_1, \ldots, d_k\}$ and let $v_{r_i} \in \alpha^{-1}(d_i)$, for $1 \leq i \leq k$. The* abstract nogood *of α with respect to $\mathcal{P}$, denoted by $\mathit{Anogood}(\alpha, \mathcal{P})$, is the set of all functions $\gamma : \mathit{scope}(\alpha) \to D$ satisfying the condition*

$$\forall i \in 1 \ldots k : \forall v_j \in \alpha^{-1}(d_i) : \gamma(v_j) = \gamma(v_{r_i}) \;\; \& \;\; \mathit{alldiff}(\gamma(v_{r_1}), \ldots, \gamma(v_{r_k})).$$

Again, by abuse of language, we identify an abstract nogood, which is a set of functions, with the condition that its members have to satisfy.

Example 5. In the existential nogood of β in Example 4, the variables e_1, e_2, and e_3 can be eliminated to produce as abstract nogood of β the following condition:

$$\gamma(v_1) = \gamma(v_4) \;\; \& \;\; \gamma(v_2) = \gamma(v_5) \;\; \& \;\; \mathit{alldiff}(\gamma(v_1), \gamma(v_2), \gamma(v_3))$$

or, more precisely, the set of functions $\gamma : \mathit{scope}(\beta) \to D$ satisfying this condition.

The following lemma indicates that an abstract nogood precisely captures its existential nogood:

Lemma 3. *Let α be a nogood for a fully value-interchangeable CSP $\mathcal{P}$: $\mathit{Enogood}(\alpha, \mathcal{P}) = \mathit{Anogood}(\alpha, \mathcal{P})$.*

Proof. Let $\mathit{image}(\alpha) = \{d_1, \ldots, d_k\}$.

We first show that $\mathit{Enogood}(\alpha, \mathcal{P}) \subseteq \mathit{Anogood}(\alpha, \mathcal{P})$. Let $\gamma \in \mathit{Enogood}(\alpha, \mathcal{P})$. Then there exist some values $a_1, \ldots, a_k \in D$ such that

$$\forall i \in 1 \ldots k : \forall v_j \in \alpha^{-1}(d_i) : \gamma(v_j) = a_i \;\; \& \;\; \mathit{alldiff}(a_1, \ldots, a_k).$$

We must show that $\gamma \in \mathit{Anogood}(\alpha, \mathcal{P})$, i.e., that γ satisfies the formula

$$\forall i \in 1 \ldots k : \forall v_j \in \alpha^{-1}(d_i) : \gamma(v_j) = \gamma(v_{r_i}) \;\; \& \;\; \mathit{alldiff}(\gamma(v_{r_1}), \ldots, \gamma(v_{r_k}))$$

where $v_{r_i} \in \alpha^{-1}(d_i)$, for $1 \leq i \leq k$. Since $a_1, \ldots, a_k$ are all different, there exists a bijection b satisfying $\forall i \in 1 \ldots k : b(d_i) = a_i$. Hence γ can be written as $b \circ \alpha$. Since $\forall i \in 1 \ldots k : \alpha(v_{r_i}) = d_i$, we have that $\gamma(v_{r_i}) = a_i$ and hence $\gamma \in \mathit{Anogood}(\alpha, \mathcal{P})$.

We now show that $\mathit{Anogood}(\alpha, \mathcal{P}) \subseteq \mathit{Enogood}(\alpha, \mathcal{P})$. Let $\gamma \in \mathit{Anogood}(\alpha, \mathcal{P})$. This means that

$$\forall i \in 1 \ldots k : \forall v_j \in \alpha^{-1}(d_i) : \gamma(v_j) = \gamma(v_{r_i}) \;\;\&\;\; \mathit{alldiff}(\gamma(v_{r_1}), \ldots, \gamma(v_{r_k}))$$

where $v_{r_i} \in \alpha^{-1}(d_i)$, for $1 \leq i \leq k$. We must show that $\gamma \in \mathit{Enogood}(\alpha, \mathcal{P})$, i.e., that there exist some values $a_1, \ldots, a_k \in D$ such that

$$\forall i \in 1 \ldots k : \forall v_j \in \alpha^{-1}(d_i) : \gamma(v_j) = a_i \;\;\&\;\; \mathit{alldiff}(a_1, \ldots, a_k).$$

Since $\gamma(v_{r_1}), \ldots, \gamma(v_{r_k})$ are all different and each $\gamma(v_{r_i}) \in D$, we may pick $\gamma(v_{r_i})$ as our choice for a_i. Hence $\gamma \in \mathit{Enogood}(\alpha, \mathcal{P})$. □

For simplicity, we will often denote the abstract nogood condition in terms of another global condition:

$$\forall i \in 1 \ldots k : \mathit{allequal}(\gamma(v_j) \mid v_j \in \alpha^{-1}(d_i)) \;\;\&\;\; \mathit{alldiff}(\gamma(v_{r_1}), \ldots, \gamma(v_{r_k}))$$

where $\mathit{allequal}(a_1, \ldots, a_n)$ holds if all the a_i are the same value.

Example 6. The abstract nogood of β in Example 5 can now be rewritten as:

$$\mathit{allequal}(\gamma(v_1), \gamma(v_4)) \;\;\&\;\; \mathit{allequal}(\gamma(v_2), \gamma(v_5)) \;\;\&\;\; \mathit{alldiff}(\gamma(v_1), \gamma(v_2), \gamma(v_3))$$

Testing Violation of Nogoods. We now show that membership to the closure of a nogood can be tested in linear time.

Lemma 4. *Let α be a nogood for a fully value-interchangeable CSP $\mathcal{P}$ and let θ be a partial assignment for $\mathcal{P}$. There exists a linear-time algorithm to test whether θ violates any nogood in $\mathit{Closure}(\alpha, \mathcal{P})$.*

Proof. Direct consequence of Lemmas 2 and 3 as well as of the fact that the size of an abstract nogood condition is linear in $|V|$. So it suffices to test whether θ satisfies the abstract nogood condition of α whenever $\mathit{scope}(\alpha) \subseteq \mathit{scope}(\theta)$. □

Lemma 5. *Let $\mathcal{P} = \langle V, D, C \rangle$ be a fully value-interchangeable CSP with $D = \{d_1, \ldots, d_m\}$. Let α be a partial assignment for $\mathcal{P}$ with $\mathit{scope}(\alpha) = \{v_{i_1}, \ldots, v_{i_k}\}$. If every $\alpha_i \equiv \alpha \;\&\; (v_{i_{k+1}} = d_i)$ $(1 \leq i \leq m)$ is a nogood for $\mathcal{P}$, then:*

1. *α is itself a nogood for $\mathcal{P}$;*
2. *if a partial assignment θ for $\mathcal{P}$ violates a nogood in $\bigcup_{i=1}^{m} \mathit{Closure}(\alpha_i, \mathcal{P})$, then θ violates a nogood in $\mathit{Closure}(\alpha, \mathcal{P})$.*

Proof. The result 1 follows from Proposition 2, which holds for CSPs in general. Let $\theta_i \in \mathit{Closure}(\alpha_i, \mathcal{P})$, hence $\theta_i = b \circ \alpha_i$ for some bijection $b : D \to D$. Since α_i extends α, it follows that $b \circ \alpha \in \mathit{Closure}(\alpha, \mathcal{P})$ and that if θ violates θ_i, then θ violates $b \circ \alpha$. Hence the result 2. □

Maintaining Nogoods. Lemma 5 indicates that abstract nogoods are needed only for the current *frontier nodes* of the search tree (i.e., the closed nodes whose parents are open). Once its child nodes are explored, the abstract nogood of a parent node subsumes the abstract nogoods of these child nodes. Hence, maintaining the nogood takes space $O(|F||V|)$, where F is the set of frontier nodes. We now formalize this result using variable decomposition trees.

Definition 17 (Variable Decomposition Tree). *A* variable decomposition tree *for a CSP $\mathcal{P} = \langle V, D, C\rangle$ is a search tree where nodes represent partial assignments for $\mathcal{P}$ and nodes are decomposed as follows: given a node representing a partial assignment α, where $scope(\alpha) = \{v_{i_1}, \ldots, v_{i_k}\}$, its child nodes represent the partial assignments $\alpha_i = \alpha \ \&\ (v_{i_{k+1}} = d_i)$ for all $d_i \in D$ and some variable $v_{i_{k+1}} \in V \setminus scope(\alpha)$.*

Note that variable decomposition trees capture both static and dynamic variable orderings, as well as a variety of search strategies (depth-first search, limited-discrepancy search, etc).

Theorem 4. *Let $\mathcal{P}$ be a fully value-interchangeable CSP and let F be the set of frontier nodes in a variable decomposition tree for $\mathcal{P}$.*

1. *Value symmetry breaking for $\mathcal{P}$ requires $O(|F||V|)$ space for storing the nogoods.*
2. *Testing if a partial assignment violates a nogood takes $O(|F||V|^2)$ time in the worst case.*

Proof. The result 1 follows from the fact that the size of an abstract nogood is linear in $|V|$. The result 2 follows from the result 1 and Lemma 4. □

Simplification. The result above can be strengthened considerably, using partial evaluation and the structure of abstract nogoods. We now show that search procedures exploring a variable decomposition tree for a fully value-interchangeable CSP can remove *all* the value symmetries while causing only *constant* overhead with respect to both time and space at every node explored. Before presenting the theoretical results, we illustrate the idea using an example with depth-first search. The basic intuition comes from the structure of the abstract nogoods.

Example 7. Consider the partial assignment

$$\theta(v_1) = 1 \ \&\ \theta(v_2) = 2 \ \&\ \theta(v_3) = 3 \ \&\ \theta(v_4) = 1 \ \&\ \theta(v_5) = 2$$

and assume that depth-first search tries next to label variable v_6, whose set of possible values is $1 \ldots 10$. The failure of $v_6 = 1$ produces the abstract nogood

$$allequal(\gamma(v_1), \gamma(v_4), \gamma(v_6)) \ \&\ allequal(\gamma(v_2), \gamma(v_5)) \ \&\ alldiff(\gamma(v_1), \gamma(v_2), \gamma(v_3)).$$

Since $v_1, \ldots, v_5$ remain instantiated when the next value is tried for v_6, the abstract nogood for this part of this next branch partially evaluates to $\gamma(v_6) = 1$,

imposing that v_6 be labelled with a value different from 1. The failures of $v_6 = 2$ and $v_6 = 3$ produce similar abstract nogoods for the other values already used in θ. Now consider the values not already used in θ and observe what happens for a failed labelling of v_6 with a value in $4 \dots 10$, say 6. The abstract nogood then is

$$\mathit{allequal}(\gamma(v_1), \gamma(v_4)) \ \&\ \mathit{allequal}(\gamma(v_2), \gamma(v_5)) \ \&\ \mathit{alldiff}\,(\gamma(v_1), \gamma(v_2), \gamma(v_3), \gamma(v_6))$$

which partially evaluates to $\mathit{alldiff}\,(1, 2, 3, \gamma(v_6))$. The disjunction of the four partially evaluated abstract nogoods obtained so far is the condition

$$\gamma(v_6) = 1 \ \vee\ \gamma(v_6) = 2 \ \vee\ \gamma(v_6) = 3 \ \vee\ \mathit{alldiff}\,(1, 2, 3, \gamma(v_6))$$

which must not be satisfied by any labelling of v_6. It follows that v_6 need only be labelled with the previously used values in $1 \dots 3$ or with exactly one new value in $4 \dots 10$.

In other words, in a variable decomposition tree, only *some* of the child nodes of a partial assignment θ need to be explored, namely those that label the next variable $v_{i_{k+1}}$ with a value in $\mathit{image}(\theta)$ or with exactly one other value. *Note that this result is independent of the set of constraints.* It is the essence of the labelling procedure for graph coloring in [16] and in the scene allocation problem in [26]. This procedure, which breaks all the value symmetries for fully value-interchangeable CSPs, is formalized in Figure 2 as procedure `fValIlabel`. It uses a function $\mathit{Failure}(\mathcal{P}, \theta)$, which returns **false** if at least one extension of the partial assignment θ is a solution to the CSP $\mathcal{P} = \langle V, D, C \rangle$. In other words, it satisfies the property

$$\mathit{Failure}(\mathcal{P}, \theta) \Rightarrow \forall \beta \in \mathit{Comp}(\theta, \mathcal{P}) \ .\ \neg C(\beta).$$

To prove the correctness of `fValIlabel` and related search procedures, it is useful to introduce the concept of compact variable decomposition tree:

Definition 18 (Compact Variable Decomposition Tree). *A* compact variable decomposition tree *for a fully value-interchangeable CSP $\mathcal{P} = \langle V, D, C \rangle$ is a search tree where nodes represent partial assignments for $\mathcal{P}$ and nodes are decomposed as follows: given a node representing a partial assignment α, where $\mathit{scope}(\alpha) = \{v_{i_1}, \dots, v_{i_k}\}$, its child nodes represent the partial assignments $\alpha_i = \alpha \ \&\ (v_{i_{k+1}} = d_o)$ for all $d_o \in \mathit{image}(\alpha)$ and the partial assignment $\alpha_i = \alpha \ \&\ (v_{i_{k+1}} = d_n)$ for some $d_n \in D \setminus \mathit{image}(\alpha)$, if $D \setminus \mathit{image}(\alpha)$ is not empty, for some variable $v_{i_{k+1}} \in V \setminus \mathit{scope}(\alpha)$.*

Compact variable decomposition trees are complete:

Lemma 6. *Let S be the set of assignments in a compact variable decomposition tree for a fully value interchangeable CSP $\mathcal{P} = \langle V, D, C \rangle$. Then the closure $\{b \circ \alpha \mid \alpha \in S$ and $b : D \to D$ is a bijection$\}$ is equal to $\mathit{Sol}(\mathcal{P})$.*

Proof. This follows directly from the examination above of the nogoods. □

```
bool fValIlabel(P) {
   return fValIlabelA(P,ε);
}
bool fValIlabelA(⟨V, D, C⟩,θ) {
   if scope(θ) = V then
      return C(θ);
   select v in V \ scope(θ);
   A := image(θ);
   if A ≠ D then
      select f in D \ A; A := A ∪ {f};
   forall(d ∈ A)
      θ' := θ & v = d;
      if ¬Failure(⟨V, D, C⟩, θ') then
         if fValIlabelA(⟨V, D, C⟩,θ') then
            return true;
   return false;
}
```

Fig. 2. A labeling procedure for fully value-interchangeable CSPs

A compact variable decomposition tree never extends any nogood generated during search:

Lemma 7. *Let T be a compact variable decomposition tree for a fully value-interchangeable CSP $\mathcal{P}$. The partial assignment of a node in T never extends any nogood generated during the exploration of T (except the one it possibly generates).*

Proof. By Lemma 5, it suffices to show that a partial assignment θ never extends a nogood generated by its siblings or the siblings of one of its ancestors in the tree. The proof is by induction on the depth of the tree. At the depth of θ, the result follows from the inspections of the nogood as discussed earlier. Consider a depth $dp' < dp$ and a nogood α generated by one of the left or right branches at that depth. We can restrict our attention to the projection of θ to the variables instantiated at that depth, i.e., we can restrict our attention to $\theta' : \mathit{scope}(\alpha) \rightarrow D$ satisfying $\forall v \in \mathit{scope}(\alpha) : \theta'(v) = \theta(v)$.

We show that $\theta' \notin \mathit{Closure}(\alpha, \mathcal{P})$. Let v be the variable assigned at depth dp'. Observe that there exists a partial assignment α' such that $\alpha = \alpha' \ \& \ (v = e_1)$ and $\theta' = \alpha' \ \& \ (v = e_2)$, for some $e_1, e_2 \in D$ and $e_1 \neq e_2$. By definition of a compact variable decomposition tree, the values e_1 and e_2 must belong to $\mathit{image}(\alpha') \cup \{d\}$, where $d \notin \mathit{image}(\alpha')$.

Consider the case where $e_1, e_2 \in \mathit{image}(\alpha)$. This means that there exist $v_i, v_j \in \mathit{scope}(\alpha)$ such that $\alpha(v) = \alpha(v_i) \neq \alpha(v_j)$ and $\theta'(v) = \theta'(v_j) \neq \theta'(v_i)$. If $\theta' \in \mathit{Closure}(\alpha, \mathcal{P})$, then it can be rewritten as $\theta' = b \circ \alpha$ for some bijection b. Hence $\theta'(v) = \theta'(v_i)$, which is impossible.

Assume now that $e_1 \in \mathit{image}(\alpha')$ and $e_2 = d$. It follows that $\alpha(v) = \alpha(v_i)$ for some $v_i \in \mathit{scope}(\alpha')$ and that $\theta'^{-1}(d) = \{v\}$. If $\theta' \in \mathit{Closure}(\alpha, \mathcal{P})$, then $\theta' = b \circ \alpha$ for some bijection. Hence $\theta'(v) = \theta'(v_i)$, which is impossible.

Assume finally that $e_2 \in image(\alpha')$ and $e_1 = d$. Then, $\alpha^{-1}(d) = \{v\}$ and $\theta'(v) = \theta'(v_i)$ for some $v_i \in scope(\alpha')$. If $\theta' \in Closure(\alpha, \mathcal{P})$, then $\theta' = b \circ \alpha$ for some bijection b. Hence, $\theta'(v) = b(d)$ & $\theta'^{-1}(b(d)) = \{v\}$, which is impossible since $\theta'(v) = \theta'(v_i)$ for some $v_i \in scope(\alpha')$. □

We can now establish the correctness of the procedure `fValIlabel`:

Theorem 5. *Procedure* `fValIlabel` *breaks all the value symmetries of a fully value-interchangeable CSP with a constant overhead with respect to both time and space at every node explored, i.e., it never extends any member of the closure of any nogood generated during search.*

Proof. The result follows directly from Lemmas 4 to 7. □

Other search strategies, e.g., limited-discrepancy search, can also be adapted to remove all the value symmetries of fully value-interchangeable CSPs with a *constant* overhead with respect to both time and space at every node explored. Experimental results for this known labelling procedure have been reported elsewhere, e.g., in [16, 26].

5.2 Piecewise Value-Interchangeable CSPs

We now derive generalizations for *piecewise* value-interchangeable CSPs of the previous results.

Definition 19 (Piecewise Value-Interchangeable CSP). *A CSP* $\mathcal{P} = \langle V, \sum_l D_l, C \rangle$ *is a* piecewise value-interchangeable CSP *if, for each solution* $\sigma \in Sol(\mathcal{P})$ *and each piecewise bijection* b *over* $\sum_l D_l$*, we have* $b \circ \sigma \in Sol(\mathcal{P})$*.*

Example 8. For scene allocation (see Example 1), we can imagine a version of the problem where the days are divided into morning and afternoon sessions. The actors probably have strong preferences (and thus different fees for these sessions), but the day of the session may still not matter.

We state the main definitions and theorems only, since the derivation is similar to the one for fully value-interchangeable CSPs.

Definition 20 (Closure of a Nogood). *Let* α *be a nogood for a piecewise value-interchangeable CSP* $\mathcal{P} = \langle V, \sum_l D_l, C \rangle$*. The* closure *of* α *for* $\mathcal{P}$*, denoted by* $Closure(\alpha, \mathcal{P})$*, is the set* $\{b \circ \alpha \mid b$ *is a piecewise bijection over* $\sum_l D_l\}$*.*

We now define abstract nogoods for piecewise value-interchangeable CSPs. The key intuition is to separate the values from each D_l.

Definition 21 (Abstract Nogood). *Let* α *be a nogood for a piecewise value-interchangeable CSP* $\mathcal{P} = \langle V, D, C \rangle$*, where* $D = \sum_{l \leq s} D_l$*. Let* $image(\alpha) = \{d_1^1, \ldots, d_{s_1}^1, \ldots, d_1^s, \ldots, d_{s_s}^s\}$*, where* $d_i^l \in D_l$*, and let* $v_{r_i^l} \in \alpha^{-1}(d_i^l)$*, for* $1 \leq i \leq s_l$ *and* $1 \leq l \leq s$*. The* abstract nogood *of* α *with respect to* $\mathcal{P}$*, denoted by*

```
bool pValIlabel(P) {
   return pValIlabelA(P,ε);
}
bool pValIlabelA(⟨V, ∑_{l≤s} D_l, C⟩,θ) {
   if scope(θ) = V then
      return C(θ);
   select v in V \ scope(θ);
   forall(l ∈ 1...s)
      A_l := image(θ) ∩ D_l;
   forall(l ∈ 1...s)
      if A_l ≠ D_l then
         select f in D_l \ A_l; A_l := A_l ∪ {f};
   forall(d ∈ ⋃_l A_l)
      θ' := θ & v = d;
      if ¬Failure(⟨V, ∑_{l≤s} D_l, C⟩, θ') then
         if pValIlabelA(⟨V, ∑_{l≤s} D_l, C⟩, θ') then
            return true;
   return false;
}
```

Fig. 3. A labelling procedure for piecewise value-interchangeable CSPs

Anogood$(\alpha, \mathcal{P})$*, is the set of all functions* $\gamma : scope(\alpha) \to \sum_l D_l$ *satisfying the condition*

$$\forall i \in 1 \ldots s_l : allequal(\gamma(v_j) \mid v_j \in \alpha^{-1}(d_i^l)) \ \&$$
$$\forall i \in 1 \ldots s_l : \forall v_j \in \alpha^{-1}(d_i^l) : v_j \in D_l \ \& \ alldiff(\gamma(v_{r_1^l}), \ldots, \gamma(v_{r_{s_l}^l}))$$

for $1 \leq l \leq s$.

Figure 3 depicts the labelling procedure `pValIlabel` for piecewise value-interchangeable CSPs. It generalizes `fValIlabel` of Figure 2 by considering the already assigned values in the sets D_l, as well as one new value (if any) from each set: the procedure `fValIlabel` is obtained when the partition of D has only one part (that is, when $s = 1$). Its correctness proof is similar to the one of Theorem 5.

Theorem 6. *Procedure* `pValIlabel` *breaks all the value symmetries of a piecewise value-interchangeable CSP with a constant overhead with respect to both time and space at every node explored.*

Experimental results have been reported elsewhere, e.g., for partitioned graph coloring in [27].

5.3 Piecewise Value-Interchangeable Set-CSPs

We now show that symmetry breaking for piecewise value-interchangeable set-CSPs is tractable. Given a finite set S, we denote by 2^S the set of subsets of S.

Definition 22 (Piecewise Set Bijection). *Let $S = \sum_i P_i$ be a partitioned set. A bijection $b : 2^S \to 2^S$ is a* piecewise set bijection *over $2^{\sum_i P_i}$ if b is induced by a piecewise bijection over $\sum_i P_i$.*

Definition 23 (Piecewise Value-Interchangeable Set-CSP). *A set-CSP $\mathcal{P} = \langle V, 2^{\sum_i D_i}, C\rangle$ is a* piecewise value-interchangeable set-CSP *if, for each solution $\sigma \in Sol(\mathcal{P})$ and each piecewise set bijection b over $2^{\sum_i D_i}$, we have $b \circ \sigma \in Sol(\mathcal{P})$.*

We actually establish the definitions and results in full detail for fully value-interchangeable set-CSPs only, but our results generalize to the piecewise case.

Definition 24 (Set Bijection). *A bijection $b : 2^S \to 2^S$ is a* set bijection *over 2^S if $b(T) = \{b'(e_i) \mid e_i \in T\}$ for $T \in 2^S$, where $b' : S \to S$ is a bijection. We say that b is* induced *by b'.*

Definition 25 (Fully Value-Interchangeable Set-CSP). *A set-CSP $\mathcal{P} = \langle V, 2^D, C\rangle$ is a* fully value-interchangeable set-CSP *if, for each solution $\sigma \in Sol(\mathcal{P})$ and each set bijection b over 2^D, we have $b \circ \sigma \in Sol(\mathcal{P})$.*

To get an impression where such problems can be of interest, consider the following example.

Example 9. Let V be any set of v elements, called *varieties*. A *balanced incomplete block design* (BIBD) [7] is a multi-set of b subsets of V, called *blocks*, each of size k (constraint C_1), such that each pair of distinct varieties occurs together in exactly λ blocks (constraint C_2), with $2 \leq k < v$. Implied constraints are that each variety occurs in the same number of blocks (constraint C_3), namely $r = \lambda(v-1)/(k-1)$, as well as that $bk = vr$ and $\lambda < r$. A BIBD is parameterized by a 5-tuple $\langle v, b, r, k, \lambda\rangle$ of parameters, not all of which are independent. Originally intended for the design of statistical experiments, BIBDs also have applications in cryptography and elsewhere. Note that the varieties and the blocks are fully interchangeable. Finding a BIBD means finding a fixed number of same-size subsets of a fully interchangeable set: either find b subsets of size k of the set V, or, dually, find v subsets of size r of the set $\{1, \ldots, b\}$, subject to the constraint C_2.

Definition 26 (Closure of a Nogood). *Let α be a nogood for a fully interchangeable set-CSP $\mathcal{P} = \langle V, 2^D, C\rangle$. The* closure *of α for $\mathcal{P}$, denoted by $Closure(\alpha, \mathcal{P})$, is the set $\{b \circ \alpha \mid b$ is a set bijection over $2^D\}$.*

Existential Nogoods and Abstract Nogoods. We now define existential and abstract nogoods for fully value-interchangeable set-CSPs, first showing the intuition using Example 9. We take the first mentioned modelling approach (namely finding v subsets of size r of the set $\{1, \ldots, b\}$) and, for simplicity, only tackle the full interchangeability of the blocks. We will come back to the full interchangeability of the varieties just after Example 10.

Consider the $\langle 6, 10, 5, 3, 2\rangle$ BIBD (which has one solution modulo all symmetries): we want to find $v = 6$ subsets v_i of size $r = 5$ of the universe $D = \{1, \ldots, 10(= b)\}$, each giving the blocks to which variety i of V belongs, such that each block is mentioned in $k = 3$ subsets and any two subsets have an intersection of size $\lambda = 2$. Consider the (consistent) partial assignment:

$$\alpha(v_1) = \{1, 2, 3, 4, 5\} \ \&\ \alpha(v_2) = \{1, 2, 6, 7, 8\} \ \&\ \alpha(v_3) = \{1, 3, 6, 9, 10\}$$

and assume that α becomes a nogood on backtracking. The *existential nogood* of α is the set of all partial assignments γ satisfying the condition

$$\begin{gathered}\exists e_1, \ldots, e_{10} \in D : \mathit{alldiff}(e_1, \ldots, e_{10}) \ \&\ \gamma(v_1) = \{e_1, e_2, e_3, e_4, e_5\} \\ \&\ \gamma(v_2) = \{e_1, e_2, e_6, e_7, e_8\} \ \&\ \gamma(v_3) = \{e_1, e_3, e_6, e_9, e_{10}\}.\end{gathered}$$

This condition can be rewritten as follows:

$$\begin{gathered}\exists e_1, \ldots, e_{10} \in D : \mathit{alldiff}(e_1, \ldots, e_{10}) \ \&\ (e_1 \in \gamma(v_1) \ \&\ e_1 \in \gamma(v_2) \ \&\ e_1 \in \gamma(v_3)) \\ \&\ (e_2 \in \gamma(v_1) \ \&\ e_2 \in \gamma(v_2)) \ \&\ (e_3 \in \gamma(v_1) \ \&\ e_3 \in \gamma(v_3)) \\ \&\ (e_4 \in \gamma(v_1)) \ \&\ (e_5 \in \gamma(v_1)) \ \&\ (e_6 \in \gamma(v_2) \ \&\ e_6 \in \gamma(v_3)) \\ \&\ (e_7 \in \gamma(v_2)) \ \&\ (e_8 \in \gamma(v_2)) \ \&\ (e_9 \in \gamma(v_3)) \ \&\ (e_{10} \in \gamma(v_3)).\end{gathered}$$

Note that the values 4 and 5 are indistinguishable because they are the only ones to appear only in the first set. Similarly, the value 6 is not indistinguishable from any other value because it is the only value that appears only in the second and third sets. Formally:

Definition 27 (Indistinguishable Values, Cluster). *The values x and y are* indistinguishable *under a partial assignment θ, which is denoted by $x \sim y$, if $x \in \theta(v) \leftrightarrow y \in \theta(v)$ for all $v \in \mathit{scope}(\theta)$. The* clusters *of values that always appear together, and are thus indistinguishable, are the equivalence classes of $\sim$ in D under α.*

In our example, there are seven clusters:

$$\{1\},\ \{2\},\ \{3\},\ \{4, 5\},\ \{6\},\ \{7, 8\},\ \{9, 10\}. \tag{1}$$

The condition of the existential nogood of α can now be rewritten as follows, using seven existentially quantified cluster variables:

$$\begin{gathered}\exists c_1, \ldots, c_7 \subseteq D : \mathit{partition}(D, [c_1, \ldots, c_7], [1, 1, 1, 2, 1, 2, 2]) \\ \&\ \gamma(v_1) = c_1 \cup c_2 \cup c_3 \cup c_4 \ \&\ \gamma(v_2) = c_1 \cup c_2 \cup c_5 \cup c_6 \ \&\ \gamma(v_3) = c_1 \cup c_3 \cup c_5 \cup c_7\end{gathered}$$

where $\mathit{partition}(S, P, N)$ holds if the elements P_i of the set list P are non-empty, mutually disjoint, union up to the set S, and have N_i elements respectively, with the N_i being the elements of the integer list N. Note that the cluster size conditions are necessary in general, but actually implied in this example.[6] If there had been values of D that do not appear in any of the set values for the variables in the scope of α, then they would have formed a cluster by themselves.

[6] Consider a domain of five elements and a partial assignment for two set variables, S_1 and S_2, of size 3 that have one or two elements in common, that is $S_1 = e_1 \cup e_2$ and $S_2 = e_1 \cup e_3$ where e_1, e_2, e_3 are disjoint. Then e_1 can be of size 1 or 2.

Definition 28 (Signature of a Cluster). *The* signature *of a cluster c relative to a partial assignment θ, denoted by $sig(c, \theta)$, is the list of indices i of the variables v_i, which are given a set value by θ, of which c is a subset: $sig(c, \theta) = \{i \mid v_i \in scope(\theta) \ \ \& \ \ c \subseteq \theta(v_i)\}$.*

For instance, $sig(\{3\}, \alpha) = [1, 3]$ because $\{3\}$ is a subset of both $\alpha(v_1)$ and $\alpha(v_3)$. The signatures of the seven clusters in (1) relative to α respectively are:

$$[1, 2, 3],\ [1, 2],\ [1, 3],\ [1],\ [2, 3],\ [2],\ [3]. \tag{2}$$

Eliminating the existentially quantified variables, using the indices in the signatures of the clusters as a reference for what to include and what to exclude, leads to the following condition for the *abstract nogood* of α:

$$\begin{gathered} partition(D, [(\gamma(v_1) \cap \gamma(v_2) \cap \gamma(v_3)),\ (\gamma(v_1) \cap \gamma(v_2)) \setminus \gamma(v_3), \\ (\gamma(v_1) \cap \gamma(v_3)) \setminus \gamma(v_2),\ \gamma(v_1) \setminus (\gamma(v_2) \cup \gamma(v_3)),\ (\gamma(v_2) \cap \gamma(v_3)) \setminus \gamma(v_1), \\ \gamma(v_2) \setminus (\gamma(v_1) \cup \gamma(v_3)),\ \gamma(v_3) \setminus (\gamma(v_1) \cup \gamma(v_2))], [1, 1, 1, 2, 1, 2, 2]) \end{gathered}$$

where the order of the clusters is the same as in (1). If there had been values of D that do not appear in any of the set values for the variables in the scope of α, then their cluster, which would have the empty list as signature, would have been equal to $D \setminus (\gamma(v_1) \cup \gamma(v_2) \cup \gamma(v_3))$, as D is the intersection of an empty collection of sets drawn from D.

We now show that the closure of a nogood for a fully value-interchangeable set-CSP can be characterized compactly and that membership to the closure of a nogood can be tested in polynomial time in this case. We first define the concept of existential nogood.

Definition 29 (Existential Nogood). *Let α be a nogood for a fully value-interchangeable set-CSP $\mathcal{P} = \langle V, 2^D, C \rangle$. Let $c_1, \ldots, c_m$ be the equivalence classes of $\sim$ in the universe D under α, and let $n_1, \ldots, n_m$ be their respective sizes. Let I be the set of indices of the variables of V that are in $scope(\alpha)$. Let J_i be the set of indices of the clusters that are in $\alpha(v_i)$. The* existential nogood *of α for $\mathcal{P}$, denoted by $Enogood(\alpha, \mathcal{P})$, is the set of all functions $\gamma : scope(\alpha) \to 2^D$ satisfying the condition*

$$\begin{gathered} \exists c_1, \ldots, c_m \subseteq D : partition(D, [c_1, \ldots, c_m], [n_1, \ldots, n_m]) \\ \& \ \textstyle\bigwedge_{i \in I} \left(\gamma(v_i) = \bigcup_{j \in J_i} c_j \right). \end{gathered}$$

The following lemma indicates that an existential nogood precisely captures the closure of its nogood:

Lemma 8. *Let α be a nogood for a fully value-interchangeable set-CSP $\mathcal{P}$: $Enogood(\alpha, \mathcal{P}) = Closure(\alpha, \mathcal{P})$.*

Proof. We first show that $Closure(\alpha, \mathcal{P}) \subseteq Enogood(\alpha, \mathcal{P})$. Let γ be a member of $Closure(\alpha, \mathcal{P})$. By the definition of closure, there exists a set bijection b such that $\gamma = b \circ \alpha$. Thus, since α satisfies $Enogood(\alpha, \mathcal{P})$, there exists a set $c_1, \ldots, c_m$

of equivalence classes of $\sim$ in the universe D under α that satisfies the existential formula. Then the set bijection b takes $c_1, \ldots, c_m$ to $b(c_1), \ldots, b(c_m)$, which are the witnesses for γ to satisfy $\mathit{Enogood}(\alpha, \mathcal{P})$. Hence $\gamma \in \mathit{Enogood}(\alpha, \mathcal{P})$.

We now show that $\mathit{Enogood}(\alpha, \mathcal{P}) \subseteq \mathit{Closure}(\alpha, \mathcal{P})$. Let $\delta \in \mathit{Enogood}(\alpha, \mathcal{P})$. The equivalence classes of $\sim$ in D under δ will be $c_1, \ldots, c_m$, which are the witnesses for δ to satisfy $\mathit{Enogood}(\alpha, \mathcal{P})$. Further suppose that the equivalence classes of $\sim$ in D under α are $c'_1, \ldots, c'_m$. Because the sizes of each of c_i and c'_i are equal and both the c_i and the c'_i partition D, there exists a set bijection b on 2^D taking c_i to c'_i. Hence δ can be rewritten as $b \circ \alpha$ and $\delta \in \mathit{Closure}(\alpha, \mathcal{P})$. □

It is not obvious that membership to an existential nogood can be tested efficiently since it involves an existential quantification. However, due to the nature of the underlying conditions, it is possible to eliminate the existential variables and obtain an abstract nogood, as defined next:

Definition 30 (Abstract Nogood). *Let α be a nogood for a fully value-interchangeable set-CSP $\mathcal{P} = \langle V, 2^D, C \rangle$. Let I be the set of indices of the variables of V that are in $\mathit{scope}(\alpha)$. Let E be the list of equivalence classes of $\sim$ in the universe D under α, and let N be the list of their respective sizes. The* abstract nogood *of α with respect to $\mathcal{P}$, denoted by $\mathit{Anogood}(\alpha, \mathcal{P})$, is the set of all functions $\gamma : \mathit{scope}(\alpha) \to 2^D$ satisfying the condition*

$$\mathit{partition}\left(D, \left[\textstyle\bigcap_{j \in sig(e,\alpha)} \gamma(v_j) \;\setminus\; \bigcup_{j \in I \setminus sig(e,\alpha)} \gamma(v_j) \mid e \in E\right], N\right).$$

The following lemma indicates that an abstract nogood precisely captures its existential nogood:

Lemma 9. *Let α be a nogood for a fully value-interchangeable set-CSP $\mathcal{P}$: $\mathit{Enogood}(\alpha, \mathcal{P}) = \mathit{Anogood}(\alpha, \mathcal{P})$.*

Proof. This follows from the definition of the abstract nogood and the fact that the expression

$$\bigcap_{j \in sig(e,\alpha)} \gamma(v_j) \;\setminus\; \bigcup_{j \in I \setminus sig(e,\alpha)} \gamma(v_j)$$

captures exactly each c_i in the existential nogood. □

Maintaining Nogoods and Simplification. Let us now consider depth-first search, for instance, and see what happens when the assignment to v_3 is undone, making α a nogood. By the definition of clusters, the search procedure should treat the elements of a cluster as indistinguishable. Then, imposing an ordering on the elements of each cluster, the idea is to select the i^{th} element of a cluster only when the $(i-1)^{\text{st}}$ element of that cluster has already been selected as a member for the next subset variable.

Figure 4 depicts the labelling procedure `fValIsetLabel` for fully value-interchangeable set-CSPs. It uses a function $\mathit{Failure}'(\mathcal{P}, \theta, v, S)$, which returns

```
bool fValIsetLabel(⟨V, 2^D, C⟩) {
    return fValIsetLabelA(⟨V, 2^D, C⟩, ϵ, [D]);
}
bool fValIsetLabelA(⟨V, 2^D, C⟩, θ, E) {
    if scope(θ) = V then
        return C(θ);
    select v in V \ scope(θ);
    (b, S) := fValIsetLabelB(⟨V, 2^D, C⟩, θ, v, ∅, E);
    if b = true then
        θ' := θ & v = S;
        E' := UPDATE(E, S);
        return fValIsetLabelA(⟨V, 2^D, C⟩, θ', E');
    return false;
}
(bool, set) fValIsetLabelB(⟨V, 2^D, C⟩, θ, v, S, [e_1, e_2, ..., e_m]) {
    if |S| = n then
        return (true, S);
    S' := S ∪ {head(e_1)};
    if ¬Failure'(⟨V, 2^D, C⟩, θ, v, S') then
        (b, S'') := fValIsetLabelB(⟨V, 2^D, C⟩, θ, v, S', [tail(e_1), e_2, ..., e_m]);
        if b = true then
            return (true, S'');
    (b, S'') := fValIsetLabelB(⟨V, 2^D, C⟩, θ, v, S, [e_2, ..., e_m]);
    if b = true then
        return (true, S'');
    return false;
}
```

Fig. 4. A labelling procedure for fully value-interchangeable set-CSPs

false if at least one extension of the partial assignment $\theta \ \& \ v = S \cup T$ for some $T \subseteq D$ is a solution to $\mathcal{P} = \langle V, 2^D, C\rangle$. In other words, it satisfies the property

$$\begin{gathered} \mathit{Failure}'(\langle V, 2^D, C\rangle, \theta, v, S) \Rightarrow \\ \forall T \subseteq D : |S \cup T| = n \ . \ \forall \beta \in \mathit{Comp}(\theta \ \& \ v = S \cup T, \langle V, 2^D, C\rangle) \ . \ \neg C(\beta). \end{gathered}$$

Procedure `fValIsetLabel` also uses a procedure `UPDATE(`E`,`S`)`, which returns the equivalence classes (clusters) of $T \cup S$, with those of T being E.

Definition 31 (Compact Set-Variable Decomposition Tree). *A* compact set-variable decomposition tree *for a fully value-interchangeable set-CSP* $\mathcal{P} = \langle V, 2^D, C\rangle$ *is a search tree where nodes represents partial assignments for* $\mathcal{P}$ *and nodes are decomposed as follows: given a node representing a partial assignment* α*, where* $\mathit{scope}(\alpha) = \{v_{i_1}, \ldots, v_{i_k}\}$ *and* E *is the set of equivalence classes of* $\sim$ *in the universe* D *under* α*, its child nodes represent the partial assignment* $\alpha_i = \alpha \ \& \ (v_{i_{k+1}} = S_i)$ *where the following two conditions hold:*

$$\forall (i,j) : \forall e \in E : |S_i \cap e| = |S_j \cap e| \Rightarrow S_i \cap e = S_j \cap e$$

and

$$\forall (i,j) : \forall e \in E : |S_i \cap e| < |S_j \cap e| \Rightarrow S_i \cap e \subset S_j \cap e$$

The first condition says that if any two sets have the same number of elements from the the same equivalence class then they have the same elements, while the second condition together with the first condition forces the elements of each equivalence class to be picked in a certain order.

Compact set-variable decomposition trees are complete:

Lemma 10. *Let S be the set of assignments in a compact set-variable decomposition tree for a fully value-interchangeable set-CSP $\mathcal{P} = \langle V, 2^D, C\rangle$. The closure $\{b \circ \alpha \mid \alpha \in S$ and b is a set bijection over $2^D\}$ is equal to $Sol(\mathcal{P})$.*

Proof. This follows directly from the examination above of the nogoods and the fact that for any node representing a partial assignment α and any deeper node representing the partial assignment α', the set E' of equivalence classes of $\sim$ in the universe D under α' is a refinement of the set E of equivalence classes of $\sim$ in D under α. □

A compact set-variable decomposition tree never extends any nogood generated during search.

Lemma 11. *Let T be a compact set variable decomposition tree for a fully value-interchangeable set-CSP $\mathcal{P} = \langle V, 2^D, C\rangle$. The partial assignment of a node T never extends any nogood generated during the exploration of T (except the one is possibly generates).*

Proof. The main structure of the proof of this lemma is similar to the one of Lemma 7. Given a partial assignment θ, we have to show that θ does not belong to the closure of some α which is a partial assignment at a lower depth. It suffices to restrict θ to θ', which is θ restricted to $scope(\alpha)$ (that is $scope(\theta') = scope(\alpha)$ and for $\forall v \in scope(theta') : \theta'(v) = \theta(v)$). It suffices to show that $\theta' \notin Closure(\alpha, \mathcal{P})$. Observe that there exists a partial assignment α' such that $\theta' = \alpha' \ \& \ (v = S_1)$ and $\alpha = \alpha' \ \& \ (v = S_2)$. By the definition of a compact set-variable decomposition tree, the sets E'_θ and E_α of equivalence classes of $\sim$ in the universe D under θ and α will be different and neither will be a refinement of the other, hence θ' cannot be in $Closure(\alpha, \mathcal{P})$. □

Theorem 7. *Procedure* `fValIsetLabel` *breaks all the value symmetries of a fully value-interchangeable set-CSP with a constant overhead with respect to both time and space at every node explored.*

Proof. This result follows from Lemmas 10 and 11. □

Procedure `fValIsetLabel` performs what is called *canonical labelling* in [10]. There, it is also shown that canonically labelling along one dimension of a matrix of variables amounts to lexicographically ordering (a flattening of) the other dimensions of that matrix. Experimental results have been reported elsewhere, e.g., in [10].

Theorem 7 generalizes to piecewise value-interchangeable set-CSPs:

Theorem 8. *All the value symmetries of a piecewise value-interchangeable set-CSP can be broken with a constant overhead with respect to both time and space at every node explored.*

Example 10. Reconsider $\langle v, b, r, k, \lambda \rangle$ BIBDs. Rather than having v set variables of size r, such that their pairwise intersections are of size λ and every block is mentioned k times, one can also have a $v \times b$ matrix of zero/one variables, such that there are r ones per row, k ones per column, and scalar products of λ for every pair of distinct rows. The *lex_chain* global constraint [5] of SICStus Prolog 3.10.0, if deployed to lexicographically order the rows of that matrix, breaks the same symmetries as our labelling procedure for the set variables. Unfortunately, that global constraint is very efficient because it is able also to filter the domains while our labelling cannot do the same filtering [18].

Let us now return to the full interchangeability of the v varieties. Breaking these extra $v!$ symmetries at the same time is hard, as they compose with the $b!$ block symmetries into $v! \cdot b!$ symmetries. Lexicographically ordering both the rows and the columns of the mentioned $v \times b$ matrix of zero/one variables does not break all these symmetries, but gives reasonable performance due to the constraint C_2 [11]. This leads to the issue whether a suitable abstract nogood can be formulated and a tractable labelling procedure be derived. In this case, it is not sufficient to store only the nogoods at the frontier nodes in the search tree; nogoods have to be stored from higher up in the search tree, as in SBDS [15] and SBDD [9]. Further, testing if a partial assignment extends a nogood is NP-complete. To see this, consider a BIBD where the blocks are of size 2; a nogood can then be thought of as a graph, each block specifying an edge. Then testing if a partial assignment is in the closure of the nogood is equivalent to subgraph isomorphism, which is NP-complete. A formal proof of this result will be given in the next section.

6 Limits of Efficient Symmetry Breaking

Until now, we have dealt with cases where symmetry could be broken efficiently. Particularly, we have shown how piecewise variable and value symmetries can be broken efficiently, and given extremely low-overhead algorithms for breaking value symmetry only. Unfortunately, as we will see in this section, there are limits to efficient symmetry breaking. We consider set-CSPs with interchangeable variables and values:

Definition 32 (Piecewise Interchangeable Set-CSP). *A set-CSP* $\mathcal{P} = \langle \sum_k V_k, 2^{\sum_l D_l}, C \rangle$ *is a* piecewise interchangeable set-CSP *if, for each solution* $\sigma \in Sol(\mathcal{P})$*, each piecewise bijection* a *over* $\sum_k V_k$*, and each piecewise set bijection* b *over* $2^{\sum_l D_l}$*, we have* $b \circ \sigma \circ a \in Sol(\mathcal{P})$*.*

When trying to break the symmetry in piecewise interchangeable set-CSPs by means of SBDD, we need to solve the following dominance detection problem efficiently.

Definition 33 (Dominating a Set Assignment). *Let* $\mathcal{P} = \langle \sum_k V_k, 2^{\sum_l D_l}, C \rangle$ *be a piecewise interchangeable set-CSP. Set assignment* α dominates *set assignment* β *if and only if there exist a piecewise bijection* a *over* $\sum_k V_k$ *and a piecewise set bijection* b *over* $2^{\sum_l D_l}$ *such that for every* $v \in scope(\alpha)$ *we have* $\beta(a(v)) = b(\alpha(v))$.

We will show that solving this problem is NP-hard, thus proving that SBDD is not able to break piecewise symmetry in set-CSPs efficiently. More precisely, we reduce the corresponding dominance detection problem to subgraph-isomorphism. To achieve the desired reduction, we construct a set assignment from a graph in the following way:

Definition 34 (Set Assignment α_G). *Given an undirected graph* $G = (V, E)$ *with* $c := |V|$*, we create a set of interchangeable values* $N := \{n_1, \ldots, n_c\}$ *and a set of interchangeable variables* $V := \{p_{ij} \mid \{i, j\} \in E\}$*. Then, the* set assignment α_G *is defined as* $\alpha_G := \bigwedge_{\{i,j\} \in E} (p_{ij} = \{n_i, n_j\})$.

Theorem 9. *Given two undirected graphs* $G_1 = (V, E_1)$ *and* $G_2 = (V, E_2)$*, we have that* G_1 *is sub-isomorphic to* G_2 *if and only if* α_{G_1} *dominates* α_{G_2} *when all variables and values are considered to be interchangeable.*

Proof. We start by showing that α_{G_1} dominates α_{G_2} if G_1 is sub-isomorphic to G_2. Let $\sigma : V \to V$ be bijective such that $\{i, j\} \in E_1$ implies $\{\sigma(i), \sigma(j)\} \in E_2$. Then, for all $p_{ij} \in scope(\alpha_{G_1})$ with $\alpha_{G_1}(p_{ij}) = \{n_i, n_j\}$ we have that $\alpha_{G_2}(p_{\sigma(i),\sigma(j)}) = \{n_{\sigma(i)}, n_{\sigma(j)}\}$. Therefore, α_{G_1} dominates α_{G_2}.

Now, let us assume that α_{G_1} dominates α_{G_2}. Then, there exist functions $a : E_1 \to E_2$ and $b : V \to V$ such that for all $p_{ij} \in scope(\alpha_{G_1})$ with $\alpha_{G_1}(p_{ij}) = \{n_i, n_j\}$ we have that $\alpha_{G_2}(p_{a(\{i,j\})}) = \{n_{b(i)}, n_{b(j)}\}$. By construction of α_{G_2}, this is equivalent to $\{n_{b(i)}, n_{b(j)}\} \in E$ for all $\{i, j\} \in E$. Thus, b is a sub-isomorphism between G_1 and G_2. □

With Theorem 9, it is possible to prove the following corollary:

Corollary 1. *The dominance detection problem for piecewise interchangeable set-CSPs is NP-hard.*

Proof. We reduce the problem to subgraph-isomorphism. In order to apply Theorem 9, we need to ensure that both graphs operate over the same set of nodes. When the sets of nodes of the given graphs differ, it is possible to see that G_1 cannot be sub-isomorphic to G_2 if G_1 contains more nodes than G_2. When G_1 actually contains fewer nodes than G_2, it is possible 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 relabeling 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 dominance detection problem for piecewise interchangeable set-CSPs is still

tractable. For example, when the set variables cannot take overlapping sets as values, the algorithm developed in Section 3 can be adapted (by exchanging the roles of values and variables) to break all the symmetries efficiently. Hence the following corollary of Theorem 1:

Corollary 2. *The dominance detection problem for piecewise interchangeable set-CSPs is tractable for non-overlapping sets.*

Note that the dominance detection problem as we consider it here regards arbitrary partial 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 partial 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 piecewise interchangeable set-CSPs efficiently. The specific dominance detection problems that SBDD considers differ from the general dominance detection problem by the fact that the partial assignments α and β that are compared are not arbitrary. We know that there exists exactly one assignment $v = d$ such that $\alpha = \gamma \ \ \& \ \ (v = d)$, while $\beta = \gamma \ \ \& \ \ \delta$, and $v \in Dom(\delta)$ for some partial assignments γ and δ.

We prove that dominance detection even for this limited problem is still NP-hard by using the same idea as before, but this time we only consider complete subgraphs, i.e., we reduce to the clique problem rather than to arbitrary subgraph isomorphism. Given a graph G and a value k, the first assignment is based on a complete graph of size k and it is defined in accordance with Definition 34. The second assignment is based on G with an additional, disconnected component that is a complete graph of size k with just one edge missing. With this setting, the first and second assignments have the same structural relationship as assignments that need to be compared within SBDD. Moreover, the given graph contains a clique of size k if and only if the first assignment dominates the second. Consequently, for piecewise interchangeable set-CSPs, SBDD is not capable of breaking symmetries efficiently.

As a final note on this negative result, we would like to stress that this does not imply that symmetry breaking is NP-hard in general since we do not consider other methods here like remodeling or the adaptation of the branching scheme.

7 Generalizations: Wreath Value-Interchangeability

So far, we have focussed on piecewise symmetry only. In this section, we generalize some of our tractability results to the more complex class of CSPs where each variable is assigned a pair of values (d_1, d_2) from a domain $D_1 \times D_2$. All values in D_1 are interchangeable and, for a fixed value in D_1, all values in D_2 are interchangeable as well. These problems are here called *wreath value-interchangeable* CSPs, because the symmetry group corresponds to a wreath

product of groups [4]. Such problems arise naturally in a variety of applications, e.g., in resource allocation and scheduling.

Example 11. Consider the problem of scheduling a meeting where different groups must meet some day of the week in some room, subject to constraints. The days are fully interchangeable and, on a given day, the rooms are fully interchangeable.

7.1 Wreath Value-Interchangeable CSPs

We now formally define this class of CSPs. Our definitions and results only consider *two* sets of *fully* interchangeable values, for simplicity. They can be generalized to an arbitrary *fixed* number of sets, and to sets of *piecewise* interchangeable values.

Definition 35 (Wreath Bijection). *Let $S = S_1 \times S_2$ be a Cartesian product. A bijection $b : S \to S$ is a* wreath bijection *over $S_1 \times S_2$ if $b(\langle e_1, e_2\rangle) = \langle b_1(e_1), b_2^{e_1}(e_2)\rangle$, where $b_1 : S_1 \to S_1$ is a bijection and each $b_2^{e_1} : S_2 \to S_2$ (for $e_1 \in S_1$) is a bijection.*

Definition 36 (Wreath Value-Interchangeable CSP). *A CSP $\mathcal{P} = \langle V, D_1 \times D_2, C\rangle$ is a* wreath value-interchangeable CSP *if, for each solution $\sigma \in Sol(\mathcal{P})$ and each wreath bijection b over $D_1 \times D_2$, we have $b \circ \sigma \in Sol(\mathcal{P})$.*

Thus, in a wreath value-interchangeable CSP, a value in the domain $D_1 \times D_2$ is assigned to each variable, where the values in D_1 are fully interchangeable, and, for a fixed value in D_1, the values in D_2 are fully interchangeable as well.

We now devise a highly efficient symmetry breaking algorithm for wreath value-interchangeable CSPs.

We use the following notations. If $d = (d_1, d_2)$ is a pair, then $d[1] = d_1$ and $d[2] = d_2$. If T is a set of tuples, then $T[i]$ denotes the set $\{d[i] \mid d \in T\}$ and $\mathit{filter}(T, i, d_i)$ denotes the set $\{d \mid d \in T \ \&\ d[i] = d_i\}$. If $\alpha : D_1 \times D_2 \to D_1 \times D_2$ is an assignment, then $\alpha^{-1}(d_1, D_2)$ denotes the set $\{\alpha^{-1}(d_1, d_2) \mid d_2 \in D_2\}$.

Definition 37 (Closure of a Nogood). *Let α be a nogood for a wreath value-interchangeable CSP $\mathcal{P} = \langle V, D_1 \times D_2, C\rangle$. The* closure *of α for $\mathcal{P}$, denoted by $\mathit{Closure}(\alpha, \mathcal{P})$, is the set $\{b \circ \alpha \mid b$ is a wreath bijection over $D_1 \times D_2\}$.*

We now define the relevant abstract nogoods.

Definition 38 (Abstract Nogood). *Let α be a nogood for a wreath value-interchangeable CSP $\mathcal{P} = \langle V, D_1 \times D_2, C\rangle$. Let $\mathit{image}(\alpha)[1] = \{d_1, \ldots, d_k\}$, let $\mathit{filter}(\mathit{image}(\alpha), 1, d_i) = \{d_1^i, \ldots, d_{l_i}^i\}$, let $v_{r_i} \in \alpha^{-1}(d_i, D_2)$, for $1 \le i \le k$, and let $v_{r_j^i} \in \alpha^{-1}(d_i, d_j)$, for $1 \le i \le k$ and $1 \le j \le l_i$. The* abstract nogood *of α with respect to $\mathcal{P}$, denoted by $\mathit{Anogood}(\alpha, \mathcal{P})$, is the set of all functions*

```
bool wValIlabel(P) {
   return wValIlabelA(P,ε);
}
bool wValIlabelA(⟨V, D1 × D2, C⟩,θ) {
   if scope(θ) = V then
      return C(θ);
   select v in V \ scope(θ);
   A1 := image(θ)[1];
   if A1 ≠ D1 then
      select f in D1 \ A1; A1 := A1 ∪ {f};
   forall(d1 ∈ A1)
      A2 := filter(image(α), 1, d1)[2];
      if A2 ≠ D2 then
         select f in D2 \ A2; A2 := A2 ∪ {f};
      forall(d2 ∈ A2)
         θ' := θ & v = (d1, d2);
         if ¬Failure(⟨V, D1 × D2, C⟩, θ') then
            if wValIlabelA(⟨V, D1 × D2, C⟩,θ') then
               return true;
   return false;
}
```

Fig. 5. A labelling procedure for wreath value-interchangeable CSPs

$\gamma : scope(\alpha) \to D_1 \times D_2$ *satisfying the condition*

$$\begin{array}{ll}
\forall i \in 1 \ldots k: & allequal(\gamma(v_j)[1] \mid v_j \in \alpha^{-1}(d_i, D_2)) \ \& \\
& alldiff(\gamma(v_{r_1})[1], \ldots, \gamma(v_{r_k})[1]) \ \& \\
\forall i \in 1 \ldots l_1: & allequal(\gamma(v_j)[2] \mid v_j \in \alpha^{-1}(d_1, d_i^1)) \ \& \\
& alldiff(\gamma(v_{r_1^1})[1], \ldots, \gamma(v_{r_{l_1}^1})[1]) \ \& \\
\ldots & \\
\forall i \in 1 \ldots l_k: & allequal(\gamma(v_j)[2] \mid v_j \in \alpha^{-1}(d_1, d_i^k)) \ \& \\
& alldiff(\gamma(v_{r_1^k})[1], \ldots, \gamma(v_{r_{l_k}^k})[1])
\end{array}$$

Figure 5 depicts the labelling procedure `wValIlabel` for wreath value-interchangeable CSPs. Its correctness proof is similar to the one of Theorem 5.

Theorem 10. *Procedure* `wValIlabel` *breaks all the value symmetries of a wreath value-interchangeable CSP with a constant overhead with respect to both time and space at every node explored.*

7.2 Wreath Value-Interchangeable Set-CSPs

We now show that symmetry breaking for wreath value-interchangeable *set*-CSPs is also tractable.

Definition 39 (Wreath Set Bijection). *Let $S = S_1 \times S_2$ be a Cartesian product. A bijection $b : 2^S \to 2^S$ is a* wreath set bijection *over $2^{S_1 \times S_2}$ if b is induced by a wreath bijection over $S_1 \times S_2$.*

Definition 40 (Wreath Value-Interchangeable Set-CSP). *A set-CSP* $\mathcal{P} = \langle V, 2^{D_1 \times D_2}, C \rangle$ *is a* wreath value-interchangeable set-CSP *if, for each solution* $\sigma \in Sol(\mathcal{P})$ *and each wreath set bijection* b *over* $2^{D_1 \times D_2}$*, we have* $b \circ \sigma \in Sol(\mathcal{P})$*.*

Consider the following example.

Example 12. Take the set $V = \{v_1, v_2\}$ of set variables over the universe $D_1 \times D_2$, with $D_1 = \{d_1, d_2\}$ and $D_2 = \{e_1, e_2, e_3\}$, such that the set-CSP $\mathcal{P} = \langle V, 2^{D_1 \times D_2}, C \rangle$ is wreath value-interchangeable, the constraint set C being arbitrary. Suppose that we have already tried the partial assignment

$$\alpha_1 = (v_1 = \{(d_1, e_1), (d_1, e_2), (d_2, e_2), (d_2, e_3)\} \ \& \ v_2 = \{(d_2, e_1), (d_2, e_2)\})$$

and that now we are about to investigate the partial assignment

$$\alpha_2 = (v_1 = \{(d_1, e_1), (d_1, e_2), (d_2, e_1), (d_2, e_2)\} \ \& \ v_2 = \{(d_1, e_2), (d_1, e_3)\}).$$

How can we decide whether α_2 is a symmetric variant of α_1 or not? One way to do that is to construct a permutation of D_1, as well as corresponding permutations of D_2, so that the sets in α_2 are transformed into those of α_1.

In order to construct a permutation σ of D_1, let us assess whether d_1 can be mapped to itself. If $\sigma(d_1) = d_1$, then $v_2 = \{(d_1, e_2), (d_1, e_3)\}$ in α_2 cannot be mapped to $v_2 = \{(d_2, e_1), (d_2, e_2)\}$ in α_1, no matter how we permute D_2. Algorithmically, we can infer this by checking whether the number of tuples starting with d_1 in α_2 is the same as the number of tuples starting with $\sigma(d_1)$ in α_1 for all assigned set variables. For v_1, the important tuples in α_2 are (d_1, e_1) and (d_1, e_2). That means that there are two such tuples, which matches the number of respective tuples for v_1 in α_1, namely (d_1, e_1) and (d_1, e_2). For v_2, the respective tuples in α_2 are (d_1, e_2) and (d_1, e_3), i.e., there are two such tuples. In α_1, on the other hand, there are no tuples starting with $\sigma(d_1) = d_1$ at all, which shows that d_1 cannot be mapped to itself.

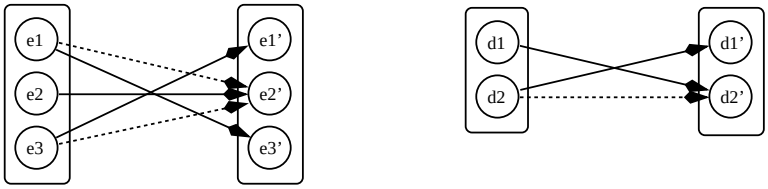
Now let us investigate whether d_1 can be mapped to d_2. First, we check whether the numbers of tuples match. For v_1 we have two tuples starting with d_1 in α_2, and in α_1 we have two tuples starting with d_2. Moreover, for v_2 we have two tuples starting with d_1 in α_2, and also two tuples starting with d_2 in α_1. Therefore, the initial check on setting $\sigma(d_1) = d_2$ is inconclusive. To check fully whether we can construct a permutation σ of D_1 with $\sigma(d_1) = d_2$, we need to find out whether there exists a permutation of D_2 such that the respective tuple sets map exactly, and not just in number. That is, we need to construct a permutation τ of D_2 such that, with $\sigma(d_1) = d_2$, we have

$$\{(\sigma(d_1), \tau(e_1)), (\sigma(d_1), \tau(e_2))\} = \{(d_2, e_2), (d_2, e_3)\} \tag{3}$$

and

$$\{(\sigma(d_1), \tau(e_2)), (\sigma(d_1), \tau(e_3))\} = \{(d_2, e_1), (d_2, e_2)\}, \tag{4}$$

or we need to show that no such permutation exists. The equations above pose the following constraints on the permutation τ that we are trying to construct: $\tau(e_1) \in \{e_2, e_3\}$, $\tau(e_2) \in \{e_2, e_3\} \cap \{e_1, e_2\}$, and $\tau(e_3) \in \{e_1, e_2\}$.



(a) (b)

Fig. 6. Part (a) gives the bipartite graph constructed to assess whether d_1 can be mapped to d_2. Part (b) shows the bipartite graph constructed to find a feasible permutation of D_1 or to show that none exists.

Fortunately, constructing τ or proving that no such permutation exists can be done by solving a maximum matching problem in a bipartite graph. The node set N is defined as the union of the sets $N_1 := \{e_1, e_2, e_3\}$ and $N_2 := \{e'_1, e'_2, e'_3\}$, where the e'_i are copies of the e_i. We define the edge set E in accordance with the constraints as given before, i.e., we add an edge $(e_i, e'_j) \in N_1 \times N_2$ to E if and only if $\tau(e_i) = e_j$ is allowed. Then, a perfect matching in $G = (N, E)$ exists if and only if there exists a permutation τ that satisfies equations (3) and (4). As we can see in Figure 6(a), a maximum matching, and consequently a permutation τ, exists that shows that we can potentially set $\sigma(d_1) = d_2$.

We continue to check whether setting $\sigma(d_2) = d_1$ and $\sigma(d_2) = d_2$ are possible. We find that for both these mappings we can construct a corresponding legal permutation τ of D_2.

Now, equipped with that knowledge, we can try at last to construct σ where we must ensure that $\sigma(d_1) \in \{d_2\}$ and $\sigma(d_2) \in \{d_1, d_2\}$. Following the same idea as before, we check whether such a permutation exists by solving a maximum matching problem in a bipartite graph: see Figure 6(b). Since a perfect matching exists, we have a proof that indeed assignment α_2 is symmetric to α_1. On the other hand, the construction of σ could only have failed if no permutation of D_1 and corresponding permutations of D_2 existed.

Generally, we state:

Theorem 11. *All the value symmetries of a wreath value-interchangeable set-CSP can be broken with a polynomial time overhead at every node explored.*

Proof. As in all pure cases of value symmetry, we only need to check search nodes against their previously expanded siblings. We show how this dominance check can be performed by abstracting from the concrete example above. For all potential mappings $\sigma(d) = e$, and for all set variables v_i that were assigned values in α_1, we first check whether the number of tuples in the set $\alpha_1(v_i)$ starting with e matches the number of tuples in the set $\alpha_2(v_i)$ starting with d. If that is not the case, we note that setting $\sigma(d) = e$ is not feasible. Otherwise, we set up a bipartite graph $G_{d,e} = (N_{d,e}, E_{d,e})$ where $N_{d,e}$ consists of all possible second tuple entries f and their copies f'. An edge (f, g') is an element of $E_{d,e}$ if and only

if, for all set variables v_i that were assigned values in α_1, either $(d, f) \notin \alpha_2(v_i)$ or $(d, f) \in \alpha_2(v_i)$ & $(e, g) \in \alpha_1(v_i)$. We note $\sigma(d) = e$ as feasible if and only if there exists a perfect matching in $G_{d,e}$. Finally, we set up a bipartite graph $G = (N, E)$ where N consists of all possible first tuple entries d and their copies d'. An edge (d, e') is an element of E if and only if $\sigma(d) = e$ is feasible. Then, we report α_2 as dominated by α_1 if and only if there exists a perfect matching in G.

This method either constructs permutations that prove the dominance of α_1 or shows that no such permutation exists. When p denotes the number of possible first tuple entries and q the number of possible second tuple entries, our algorithm can be implemented to run in $O(p^2q^{2.5})$ time. □

Note that the dominance checker that we outlined in the proof above can be generalized for tuples with k entries. However, the run-time is then exponential in k. We leave open whether an efficient labelling algorithm can be formulated to break this type of value symmetry. The point here was to show that wreath value symmetry allows tractable symmetry breaking for set-CSPs.

8 Conclusion

We have theoretically studied several classes of CSPs for which symmetry breaking is tractable. These CSP classes, which encompass many practical problems, feature various forms of value or variable interchangeability and allow symmetry breaking to be performed with a *polynomial* (that is often even a *constant*) overhead with respect to both time and space at every node explored, using dedicated search procedures. Unfortunately, efficient symmetry breaking by such dominance-detection schemes has its limits, as we have identified some CSP classes where dominance detection is intractable.

Table 1 summarizes our main results, where "P (Thm i)" means that breaking all the symmetries mentioned in the corresponding row is feasible with a polynomial overhead with respect to both time and space at every node explored for the corresponding (set-) CSP in the column, as proved in Theorem i. Some of these positive tractability results, namely the ones marked "P (from Thm i)", are trivially derivable as consequences from Theorem i. However, no specialized labeling procedures are given for these particular CSP classes in this paper. The negative tractability results, marked "NP-hard (Cor. i)" and referring to Corollary i, only concern dominance-detection schemes like SBDD; it remains an open research issue whether other schemes can break those symmetries in polynomial time.

In [21] it is proved that all value symmetries of a CSP are polynomial-time tractable; this is proved using group theoretic notions and although the resulting complexities are the order of a low-degree polynomial they are in general not as efficient as the specialized algorithms presented in this paper. A key component in the proofs is the notion of a minimal GE-tree, which is essentially the search tree that results from a search procedure that breaks all symmetry. Compact

Symmetry	CSPs	Set-CSPs
fully value-interchangeable	P (Thm 5)	P (Thm 7)
fully variable-interchangeable	P (from Thm 1)	P (from Thm 1)
fully value- and variable-interchangeable	P (from Thm 1)	NP-hard (Cor. 1)
piecewise value-interchangeable	P (Thm 6)	P (Thm 8)
piecewise variable-interchangeable	P (from Thm 1)	P (from Thm 1)
piecewise value- and variable-interchangeable	P (Thm 1)	NP-hard (Cor. 1)
wreath value-interchangeable	P (Thm 10)	P (Thm 11)

Table 1. Tractability of symmetry breaking and dominance detection

(set-)variable decomposition trees (see Definitions 18 and 31) are GE-trees; in fact all the search procedures in this paper produce GE-trees.

There are many directions for future research. Of particular interest is the study of tractable classes of CSPs exhibiting variable symmetries where the variable set has a more complex structure than the partitions studied in this paper. In particular, when the variable set is obtained by a Cartesian product over some index sets, we get what is known as a *matrix model*. There are many interesting forms of interchangeability in matrix models, such as the full/piecewise/wreath interchangeability of matrix slices (rows, columns, ...) [11]. For many of these forms of variable interchangeability, including their compositions with various forms of value interchangeability, tractability results for symmetry breaking are still missing and finding effective search procedures is a challenging problem. Also, as Corollary 2 has shown, negative tractability results call for the identification of special cases where symmetry breaking is tractable.

Acknowledgements

All authors are partly supported by institutional grant IG2001-67 of STINT, the Swedish Foundation for International Cooperation in Research and Higher Education. Pierre Flener did some of this work while on sabbatical in 2006/07 at Sabancı University in İstanbul, Turkey. The Sweden-based authors were also supported by grant 221-99-369 of VR, the Swedish Research Council, during part of this work. Meinolf Sellmann is supported by the National Science Foundation through the Career: Cornflower Project (NSF award number 0644113). Pascal Van Hentenryck was partly supported by NSF ITR Award DMI-0121495.

References

1. R. Ahuja, T. Magnati, and J. Orlin. *Network Flows*. Prentice Hall, 1993.
2. R. Backofen and S. Will. Excluding symmetries in constraint-based search. In J. Jaffar, editor, *Proceedings of CP'99*, volume 1713 of *LNCS*, pages 73–87. Springer-Verlag, 1999.

3. N. Barnier and P. Brisset. Solving the Kirkman's schoolgirl problem in a few seconds. In P. Van Hentenryck, editor, *Proceedings of CP'02*, volume 2470 of *LNCS*, pages 477–491. Springer-Verlag, 2002.
4. P. Cameron. *Permutation Groups*. Number 45 in London Mathematical Society Student Texts. Cambridge University Press, 1999.
5. M. Carlsson and N. Beldiceanu. Arc-consistency for a chain of lexicographic ordering constraints. Technical Report T2002-18, Swedish Institute of Computer Science, 2002.
6. D. Cohen, P. Jeavons, C. Jefferson, K. E. Petrie, and B. M. Smith. Symmetry definitions for constraint satisfaction problems. In P. van Beek, editor, *Proceedings of CP'05*, volume 3709 of *LNCS*, pages 17–31. Springer-Verlag, 2005.
7. C. J. Colbourn and J. H. Dinitz, editors. *The CRC Handbook of Combinatorial Designs*. CRC Press, 1996.
8. J. M. Crawford, M. Ginsberg, E. Luks, and A. Roy. Symmetry-breaking predicates for search problems. In L. C. Aiello, J. Doyle, and S. C. Shapiro, editors, *Proceedings of KR'96*, pages 148–159. Morgan Kaufmann, 1996.
9. T. Fahle, S. Schamberger, and M. Sellmann. Symmetry breaking. In T. Walsh, editor, *Proceedings of CP'01*, volume 2239 of *LNCS*, pages 93–107. Springer-Verlag, 2001.
10. P. Flener, A. M. Frisch, B. Hnich, Z. Kızıltan, I. Miguel, J. Pearson, and T. Walsh. Symmetry in matrix models. In P. Flener and J. Pearson, editors, *Proceedings of SymCon'01*, 2001. Available at `http://www.it.uu.se/research/group/astra/SymCon01/`.
11. P. Flener, A. M. Frisch, B. Hnich, Z. Kızıltan, I. Miguel, J. Pearson, and T. Walsh. Breaking row and column symmetries in matrix models. In P. Van Hentenryck, editor, *Proceedings of CP'02*, volume 2470 of *LNCS*, pages 462–476. Springer-Verlag, 2002.
12. P. Flener, J. Pearson, M. Sellmann, and P. Van Hentenryck. Static and dynamic structural symmetry breaking. In F. Benhamou, editor, *Proceedings of CP'06*, volume 4204 of *LNCS*, pages 695–699. Springer-Verlag, 2006.
13. F. Focacci and M. Milano. Global cut framework for removing symmetries. In T. Walsh, editor, *Proceedings of CP'01*, volume 2239 of *LNCS*, pages 77–92. Springer-Verlag, 2001.
14. E. C. Freuder. Eliminating interchangeable values in constraint satisfaction problems. In *Proceedings of AAAI'91*, pages 227–233, 1991.
15. I. P. Gent and B. M. Smith. Symmetry breaking during search in constraint programming. In *Proceedings of ECAI'00*, pages 599–603, 2000.
16. M. Kubale and D. Jackowski. A generalized implicit enumeration algorithm for graph coloring. *CACM*, 28(4):412–418, 1985.
17. P. Meseguer and C. Torras. Exploiting symmetries within constraint satisfaction search. *Artificial Intelligence*, 129(1-2):133–163, 2001.
18. J. Pearson. Comma-free codes. In P. van Beek, editor, *Proceedings of AI&M'04*, 2004. Available at `http://rutcor.rutgers.edu/~amai/aimath04/`.
19. J.-F. Puget. On the satisfiability of symmetrical constrained satisfaction problems. In J. Komorowski and Z. Raś, editors, *Proceedings of ISMIS'93*, volume 689 of *LNAI*, pages 350–361. Springer-Verlag, 1993.
20. J.-F. Puget. Symmetry breaking revisited. In P. Van Hentenryck, editor, *Proceedings of CP'02*, volume 2470 of *LNCS*, pages 446–461. Springer-Verlag, 2002.
21. C. M. Roney-Dougal, I. P. Gent, T. Kelsey, and S. Linton. Tractable symmetry breaking using restricted search trees. In R. L. de Mántaras and L. Saitta, editors, *Proceedings of ECAI'04*, pages 211–215. IOS Press, 2004.

22. M. Sellmann and P. Van Hentenryck. Structural symmetry breaking. In *Proceedings of IJCAI'05*, pages 298–303. IJCAI, 2005.
23. I. Shlyakhter. Generating effective symmetry-breaking predicates for search problems. *Electronic Notes in Discrete Mathematics*, 9, 2001. Proceedings of SAT'01.
24. B. M. Smith. Reducing symmetry in a combinatorial design problem. In C. Gervet and M. Wallace, editors, *Proceedings of CP-AI-OR'01*, 2001.
25. B. M. Smith, S. C. Brailsford, P. M. Hubbard, and H. P. Williams. The progressive party problem: Integer linear programming and constraint programming compared. *Constraints*, 1:119–138, 1996.
26. P. Van Hentenryck. Constraint and integer programming in OPL. *INFORMS Journal on Computing*, 14(4):345–372, 2002.
27. P. Van Hentenryck, P. Flener, J. Pearson, and M. Ågren. Tractable symmetry breaking for CSPs with interchangeable values. In *Proceedings of IJCAI'03*, pages 277–282. Morgan Kaufmann, 2003.
28. P. Van Hentenryck, P. Flener, J. Pearson, and M. Ågren. Compositional derivation of symmetries for constraint satisfaction. In J.-D. Zucker and L. Saitta, editors, *Proceedings of SARA'05*, volume 3607 of *LNAI*, pages 234–247. Springer-Verlag, 2005.