New developments in symmetry breaking in search using computational group theory

Tom Kelsey, Steve Linton and Colva Roney-Dougal

School of Computer Science, University of St Andrews,
Fife, Scotland,
{tom, sal, colva}@dcs.st-and.ac.uk

Abstract. Symmetry-breaking in constraint satisfaction problems (CSPs) is a well-established area of AI research which has recently developed strong interactions with symbolic computation, in the form of computational group theory. GE-trees are a new conceptual abstraction, providing low-degree polynomial time methods for breaking value symmetries in CSPs. In this paper we analyse the structure of symmetry groups of CSPs, and implement several combinations of GE-trees and the classical SBDD method for breaking all symmetries. We prove the efficacy of our techniques, and present preliminary experimental evidence of their practical efficiency.

1 Introduction

Constraint systems are a generalization of the Boolean satisfiability problems that play a central role in theoretical computer science. Solving constraint satisfaction problems (CSPs) in general is thus NP-complete; but effective solving of constraint systems arising from real problems, such as airline scheduling, is of enormous industrial importance.

There has been a great deal of research interest in dealing with symmetries in CSPs in recent years. CSPs are often solved using AI search techniques involving backtrack search and propagation. Many approaches to dealing with symmetries (constraints and/or solutions that are interchangeable in terms of the structure of the problem) are themselves based on refinements of AI search techniques. These include imposing some sort of ordering (before search) on otherwise interchangeable elements, posting constraints at backtracks that rule out search in symmetric parts of the tree, and checking that nodes in the search tree are not symmetrically equivalent to an already-visited state. These approaches are collectively known as lexicographic ordering [1], symmetry breaking during search (SBDS) [2, 3], and symmetry breaking by dominance detection (SBDD) [4, 5]. Methods can be, and are, optimised for either finding the first solution or finding all solutions. In this paper we consider only the problem of finding all solutions.

A promising recent approach has been to consider the symmetries of a given CSP as a group of permutations. It then becomes possible to pose and answer symmetric questions using computational group theory (CGT). In effect, the AI

search for solutions proceeds as before, but with fast permutation group algorithms supplying information that restricts search to symmetrically inequivalent nodes. Both SBDS and SBDD have been successfully implemented in this way [6, 7] using GAP–ECL^{i}PS^e, an interface between the ECL^{i}PS^e [8] constraint logic programming system and the GAP [9] CGT system.

An even more recent advance has been the theory of GE-trees [10]. The construction and traversal of such a tree is guaranteed to break all symmetries in any CSP. In the special – but common – case that the CSP has only value symmetries (for example graph colouring problems, where the colours are distinct but interchangeable) a GE-tree can be constructed in low-degree polynomial time. GE-trees provide both a useful analytic framework for comparing symmetry breaking methods, and, for value symmetries, a practical method for efficient search.

In this paper we describe initial results, both theoretical and practical, concerning the integration of GE-tree construction with SBDD. We combine heuristic AI search with mathematical structures and algorithms in order to extend and enhance existing – mainly pure AI – techniques.

In the remainder of this paper we provide a formal framework for CSPs, variable and value symmetries and GE-trees. In the following section, we briefly describe SBDD and make some observations about variations of this algorithm. In the next two sections we will identify and discuss a mathematically special, but common, situation in which we can uncouple variable and value symmetries. We follow this with preliminary experimental results for a range of symmetry breaking approaches involving various combinations of SBDD and GE-tree methods.

1.1 CSPs and symmetries

Definition 1. *A CSP L is a set of constraints $\mathcal{C}$ acting on a finite set of variables $\Delta := \{A_1, A_2, \ldots, A_n\}$, each of which has finite domain of possible values $D_i := D(A_i) \subseteq \Lambda$. A* solution *to L is an instantiation of all of the variables in Δ such that all of the constraints in $\mathcal{C}$ are satisfied.*

Constraint logic programming systems such as ECL^{i}PS^e model CSPs using constraints over finite domains. The usual search method is depth-first, with values assigned to variables at choice points. After each assignment a partial consistency test is applied: domain values that are found to be inconsistent are deleted, so that a smaller search tree is produced. Backtrack search is itself a consistency technique, since any inconsistency in a current partial assignment (the current set of choice points) will induce a backtrack. Other techniques include forward-checking, conflict-directed backjumping and look-ahead.

Statements of the form $(var = val)$ are called *literals*, so a partial assignment is a conjunction of literals. We denote the set of all literals by χ, and generally denote variables in Roman capitals and values by lower case Greek letters.

Definition 2. *Given a CSP L, with a set of constraints $\mathcal{C}$, and a set of literals χ, a* symmetry *of L is a bijection $f : \chi \to \chi$ such that a full assignment A of L satisfies all constraints in $\mathcal{C}$ if and only if $f(A)$ does.*

We denote the image of a literal $(X = \alpha)$ under a symmetry g by $(X = \alpha)g$. The set of all symmetries of a CSP form a *group*: that is, they are a collection of bijections from the set of all literals to itself that is closed under composition of mappings and under inversion. We denote the symmetry group of a CSP by G.

Note that under this definition of a symmetry, it is entirely possible to map a partial assignment that does not violate any specific constraint to a partial assignment which does. Suppose for instance that we had constraints $(X_1 = X_2)$, $(X_2 = X_3)$ where each X_i had domain $[\alpha, \beta]$. Then the symmetry group of the CSP would enable us to freely interchange X_1, X_2 and X_3. However, this means that we could map the partial assigment $(X_1 = \alpha) \wedge (X_3 = \beta)$ (which does not break either of the constraints) to $(X_1 = \alpha) \wedge (X_2 = \beta)$ (which clearly does). This shows that the interaction between symmetries and consistency can be complex.

There are various ways in which the symmetries of a CSP can act on the set of all literals, we now examine these in more detail.

Definition 3. *A* value symmetry *of a CSP is a symmetry $g \in G$ such that if $(X = \alpha)g = (Y = \beta)$ then $X = Y$.*

The collection of value symmetries form a *subgroup* of G: that is, the set of value symmetries is itself a group, which we denote by G^{Val}. We distinguish two types of value symmetries: a group G acts via *pure value symmetries* if for all $g \in G^{\mathrm{Val}}$, whenever $(X = \alpha)g = (X = \beta)$, we have $(Y = \alpha)g = (Y = \beta)$. There are CSPs for which this does not hold. However, at a cost of making distinct, labelled copies of each domain we have the option to assume that G acts via pure value symmetries. If G^{Val} is pure then we may represent it as acting on the values themselves, rather than on the literals: we write αg to denote the image of α under $g \in G^{\mathrm{Val}}$.

We wish to define variable symmetries in an analogous fashion; however we must be a little more cautious at this point. We deal first with the standard case.

Definition 4. *Let L be a CSP for which all of the variables have the same domains, and let G be the full symmetry group of L. A* variable symmetry *of L is a symmetry $g \in G$ such that if $(X = \alpha)g = (Y = \beta)$, then $\alpha = \beta$.*

We need a slightly more general definition than this. Recall that if the value group does not act via pure value symmetries, we may make labelled copies of the domains for each variable: at the end of this process the conditions for Definition 4 no longer hold. However, there is a natural way of identifying the values in each of the domains with one another, and we describe $g \in G$ as being a variable symmetry if, whenever it maps a literal with variable X to a literal with variable Y, it maps corresponding values to corresponding values. Formally, we have the following definition.

Definition 5. *Let L be a CSP with symmetry group G. Fix a total ordering on D_i for each i, and denote the elements of D_i by α_{ij}. A* variable symmetry *is a symmetry $g \in G$ such that if $(A_i = \alpha_{ij})g = (A_k = \alpha_{kl})$ then $l = j$.*

If all variables share a common domain then we recover Definition 4 from Definition 5. Note that in the above definition, we may order at least one domain arbitrarily without affecting whether or not a symmetry is deemed to be a variable symmetry: it is the *relative ordering* of the domains that is crucial.

There may be several possible orderings of the domains, corresponding to different choices of variable group: the value group is often a normal subgroup of G (see section 3), and hence is uniquely determined, but there will usually be several (conjugate) copies of the variable group, corresponding to distinct split extensions. If there is a unique copy of the variable group, as well as a unique copy of the value group, then G is the direct product of these two normal subgroups. However, in the current context, ordering any one domain induces a natural order on all of the remaining domains. This is because our variables either have a common domain, or originally had a common domain which has now been relabelled into several distinct copies (one for each variable).

The collection of all variable symmetries (for a given ordering on each domain) is a subgroup of G, which we denote G^{Var}. We define a *pure variable symmetry* to be a variable symmetry such that if $(A_i = \alpha_{ij})g = (A_k = \alpha_{kj})$ then the value of k does not depend on j.

1.2 GE-trees

In [10] we introduced the GE-tree, a search tree $\mathbf{T}$ for a CSP L with the property that searching $\mathbf{T}$ finds exactly one representative of each equivalence class of solutions under the symmetry group of L. Before we can define a GE-tree, we need a few more group-theoretic and search-based definitions.

We consider only search strategies in which all allowed options for one variable are considered before any values for other variables. This is a common, although not universal, pattern in constraint programming. Therefore, we consider search trees to consist of nodes which are labelled by variables (except for the leaves, which are unlabelled), and edges labelled by values. We think of this as meaning that the variable is set to that value as one traverses the path from the root of the tree toward the leaves. At a node $\mathcal{N}$, the partial assignment given by reading the labels on the path from the root to $\mathcal{N}$ (ignoring the label at $\mathcal{N}$ itself) is the *state* at $\mathcal{N}$. We will often identify nodes with their state, when the meaning is clear. By the *values in* $\mathcal{N}$ we mean the values that occur in literals in the state at $\mathcal{N}$, we denote this set by $\mathrm{Val}(\mathcal{N})$. We define $\mathrm{Var}(\mathcal{N})$ similarly for variables. We will often speak of a permutation as mapping a node $\mathcal{N}$ to a node $\mathcal{M}$, although strictly speaking the permutation maps the literals in the state at $\mathcal{N}$ to the literals in the state at $\mathcal{M}$ (in any order).

Definition 6. *Let G be a group of symmetries of a CSP. The* stabiliser *of a literal $(X = \alpha)$ is the set of all symmetries in G that map $(X = \alpha)$ to itself.*

This set is itself a group. The orbit *of a literal* $(X = \alpha)$*, denoted* $(X = \alpha)^G$*, is the set of all literals that can be mapped to* $(X = \alpha)$ *by a symmetry in* G*. That is*

$$(X = \alpha)^G := \{(Y = \beta) : \exists g \in G \text{ s.t. } (Y = \beta)g = (X = \alpha)\}.$$

The orbit of a node is defined similarly.

Given a collection $\mathcal{S}$ of literals, the *pointwise* stabiliser of $\mathcal{S}$ is the subgroup of G which stabilises each element of $\mathcal{S}$ individually. The *setwise* stabiliser of $\mathcal{S}$ is the subgroup of G that consists of symmetries mapping the set $\mathcal{S}$ to itself.

Definition 7. *A* GE-tree *(group equivalence tree) for a CSP with symmetry group* G *is any search tree* $\mathbf{T}$ *satisfying the following two axioms:*

1. *No node of* $\mathbf{T}$ *is isomorphic under* G *to any other node.*
2. *Given a full assignment* $\mathcal{A}$*, there is at least one leaf of* $\mathbf{T}$ *which lies in the orbit of* $\mathcal{A}$ *under* G*.*

Therefore, the nodes of $\mathbf{T}$ are representatives for entire orbits of partial assignments under the group, and the action of the group on the tree fixes every node. Of course, a GE-tree will be constructed dynamically, and the constraints of a CSP will generally prevent us from searching the whole of $\mathbf{T}$. We define a GE-tree to be *minimal* if the deletion of any node (and its descendants) will delete at least one full assignment.

One of the main results in [10] is a constructive proof of the following Theorem:

Theorem 1. *Let* L *be a CSP with only value symmetries. Then breaking all symmetries of* L *is tractable.*

The theorem is proved by giving a low-degree polynomial algorithm which constructs a minimal GE-tree for L. This algorithm is summarized as follows:

At each node $\mathcal{N}$ in the search tree do:
    Compute the pointwise stabiliser $G_{(\mathrm{Val}(\mathcal{N}))}$ of $\mathrm{Val}(\mathcal{N})$.
    Select a variable X which is not in $\mathrm{Var}(\mathcal{N})$.
    Compute the orbits of $G_{(\mathrm{Val}(\mathcal{N}))}$ on $\mathrm{Dom}(X)$.
    For each orbit $\mathcal{O}$ do:
        Construct a downedge from $\mathcal{N}$ labelled with an element from $\mathcal{O}$.
    End for.
End for.

It is shown in [10] that this can all be done in low-degree polynomial time. The result of only considering this reduced set of edges is that no two nodes in the resulting tree will be equivalent to each other under the subgroup of G that consists of value symmetries.

This theorem has the following immediate corollary:

Corollary 1. *Let L be any CSP with symmetry group G. Then breaking the subgroup of G that consists only of value symmetries is tractable (i.e. can be done in polynomial time).*

This begs the question: how may we break the remaining symmetries of a CSP? We address this question in the remainder of this paper, after briefly describing SBDD.

2 Symmetry breaking by dominance detection

Again let L be any CSP with symmetry group G. During backtrack search we maintain a record $\mathcal{F}$ of *fail sets* corresponding to the roots of completed subtrees. Each fail set consists of those $(var = val)$ assignments made during search to reach the root of the subtree. We also keep track of the set P of current ground variables: variables having unit domain, either by search decision or by propagation.

The next node in the search tree is *dominated* if there is a $g \in G$ and $S \in \mathcal{F}$ such that

$$Sg \subseteq P \quad .$$

In the event that we can find suitable g and S, it is safe to backtrack, since we are in a search state symmetrically equivalent to one considered previously. In practice, the cost of detecting dominance is often outweighed by the reduction in search.

SBDD works well in practice: empirical evidence suggests that SBDD can deal with larger symmetry groups than many other techniques. It is possible to detect dominance without using group theoretic techniques. However, this involves writing a bespoke detector for each problem, usually in the form of additional predicates in the constraint logic system. Using CGT enables generic SBDD, with the CGT system needing only a generating set for G to be able to detect dominance.

A symmetry breaking technique is called *complete* if it guarantees never to return two equivalent solutions. Both SBDD and GE-trees are complete, in fact, SBDD remains complete even when the dominance check is not performed at every node, provided that it is always performed at the leaves of the search tree (i.e. at solutions). This observation allows a trade-off between the cost of performing dominance checks and the cost of unnecessary search. Another possible approach is to use an incomplete, but presumably faster, symmetry breaking technique combined with a separate 'isomorph rejection' step to eliminate equivalent solutions.

The efficient algorithm for breaking value symmetries – GE-tree construction – can safely be combined with SBDD, as shown in Theorem 13 of [10]. The algorithm implied by this theorem performs a dominance check, using the full group G of the CSP L, at each node of a GE-tree for L under G^{Val}.

In the next two sections we will identify and discuss a mathematically special, but common, situation in which we can uncouple variable and value symmetries.

We follow this with preliminary experimental results for a range of symmetry breaking approaches involving combinations of SBDD and GE-tree methods.

3 Complementary variable symmetries

For a great many CSPs, each element of the symmetry group can be uniquely written as a value symmetry followed by a variable symmetry. In this case we say that the variable symmetries form a *complement* to the value symmetries in the full symmetry group. Formally, a subgroup H_1 of a group H is a *complement* to a subgroup H_2 of H if the following conditions hold:

1. H_2 is a *normal* subgroup of H: this means that for all elements $h \in H$ the set $\{hh_2 \ : \ h_2 \in H_2\}$ is equal to the set $\{h_2h \ : \ h_2 \in H_2\}$.
2. $|H_1 \cap H_2| = 1$.
3. The set $\{h_1h_2 : h_1 \in H_1, \ h_2 \in H_2\}$ contains all elements of G.

If this is true for a CSP with symmetry group G when H_1 is G^{Var} and H_2 is G^{Val} then the CSP has *complementary variable symmetry.*

This holds, for instance, for any CSP where G is generated by pure value symmetries and pure variable symmetries. For example, in a graph colouring problem, the symmetries are generated by relabelling of the colours (pure value symmetries) and the automorphism group of the graph (pure variable symmetries).

Before going further, we collect a few facts describing the way in which the variable and value symmetries interact.

Lemma 1. *If G is generated by a collection of pure variable symmetries and some value symmetries, then G^{Val} is a normal subgroup of G.*

Proof. To show that G^{Val} is a normal subgroup of G, we show that for all $g \in G$ and all $h \in G^{\mathrm{Val}}$, the symmetry $g^{-1}hg \in G^{\mathrm{Val}}$.

Let $h \in G^{\mathrm{Val}}$ and let $g \in G$. Then g is a product of variable and value symmetries. Consider the literal $(X_i = \alpha_{ij})g^{-1}hg$. The image of a literal $(X_k = \alpha_{kl})$ under each variable symmetry in g depends only on k, not on l, and each value symmetry fixes the variables in each literal. Therefore as we move through the symmetries that make up g^{-1}, we will map X_i to various other variables, but each of these mappings will be inverted as we apply each of the symmetries that make up g. Thus $(X_i = \alpha_{ij})g^{-1}hg = (X_i = \alpha_{ij'})$ for some j', and so $g^{-1}hg$ is a value symmetry.

We note that if G contains any non-pure variable symmetries then G^{Val} is not, in general, a normal subgroup. Suppose that $g \in G^{\mathrm{Var}}$ maps $(X_i = \alpha_{ij}) \mapsto (X_k = \alpha_{kj})$, and also maps $(X_i = \alpha_{ij_1}) \mapsto (X_l = \alpha_{lj_1})$. Let $g \in G^{\mathrm{Val}}$ map $(X_i = \alpha_{ij}) \mapsto (X_i = \alpha_{ij_1})$. Then

$$\begin{aligned}(X_k = \alpha_{kj})g^{-1}hg &= (X_i = \alpha_{ij})hg \\ &= (X_i = \alpha_{ij_1})g \\ &= (X_l = \alpha_{lj_1}),\end{aligned}$$

but the map $(X_k = \alpha_{kj}) \mapsto (X_l = \alpha_{lj_1})$ is clearly *not* a value symmetry.

A symmetry group G for a CSP can quickly be tested for complementary value symmetry. There are several different ways of doing so, depending on how G has been constructed. If G has been input as a collection of pure value symmetries and a collection of pure variable symmetries then we always have complementary variable symmetry, and the construction of the subgroup of value symmetries and the group of variable symmetries is immediate.

So suppose that G has not been input in this form, and that we have G and a list of the domains for each variable. If the domains are not equal, and we have some value symmetries, then assume also that the bijections between each pair of domains are known.

We first check that the subgroup of value symmetries forms a normal subgroup of G, which can be done in time polynomial in $|\chi|$. We then form the *quotient group* of G by G^{Val}: this basically means that we divide out by the subgroup of all value symmetries, and can also be done in polynomial time. If the size of this quotient group is equal to the size of the group of variable symmetries, then the CSP has complementary variable symmetry, as it is clear that the only permutation of the set of all literals which lies in both the group of variable symmetries and the group of value symmetries is the identity map.

If the variables of the CSP share a common domain, the fastest way to find the group of variable symmetries (if this is not immediately clear from the way in which G has been described) is to compute the pointwise stabiliser in G of each of the values. This can be done in low-degree polynomial time [11].

In the next section we describe a new algorithm for symmetry breaking which is applicable to all CSPs with complementary variable symmetry.

4 SBDD on complements

This approach can be summarized by saying that we construct a GE-tree for the set of value symmetries, and then search this using SBDD. However, we do not use SBDD on the full group, as this would involve also checking the value symmetries once more, but instead carry out SBDD only on the subgroup of variable symmetries.

In more detail, we proceed as follows. At each node $\mathcal{N}$ in the search tree we start by applying the GE-tree algorithm for G^{Val} to label $\mathcal{N}$ with a variable X and produce a short list of possible downedges from $\mathcal{N}$, say $\alpha_1, \alpha_2, \ldots, \alpha_k \in \mathrm{Dom}(X)$.

We now perform dominance detection on each of $\mathcal{N} \cup (X = \alpha_i)$, but in a 2-stage process separating out the variable mapping from the value mapping. That is, using only G^{Var} we check whether $\mathrm{Var}(\mathcal{N}) \cup \{X\}$ is dominated by the variables in any other node. Each time that we find dominance by a node $\mathcal{M}$, this implies that $\mathrm{Var}(\mathcal{M}g) \subseteq \mathrm{Var}(\mathcal{N} \cup \{X\})$. We therefore apply g to the literals in $\mathcal{M}$, and check whether there is an element of G^{Val} that can map the resulting collection of literals to $\mathcal{N} \cup (X = \alpha_i)$ for $1 \leq i \leq k$, bearing in mind that we now know precisely which literal in $\mathcal{M}g$ must be mapped to each literal in

$\mathcal{N} \cap (X = \alpha_i)$, since the value group fixes the variables occurring in each literal. This latter query is therefore a low-degree polynomial time operation.

Only those α_i which are never dominated in this way are used to construct new nodes in $\mathbf{T}$.

Theorem 2. *The tree* $\mathbf{T}$ *constructed as above is a GE-tree for the full symmetry group* G*.*

Proof. We start by showing that no two nodes of this tree are isomorphic under G. Let $\mathcal{M}$ and $\mathcal{N}$ be two distinct nodes, and suppose that there exists $g \in G$ such that $\mathcal{M}g = \mathcal{N}$. By assumption, we can write all elements of the group as a product of a value symmetry and a variable symmetry, so write $g = lr$ where l is a value symmetry and r is a variable symmetry. Suppose (without loss of generality, since all group elements are invertible) that $\mathcal{M}$ is to the right of $\mathcal{N}$ in the search tree, so that we will have found $\mathcal{N}$ first during search. Then the partial assignment $\mathcal{N}' := \mathcal{N}r^{-1}$ is the image of $\mathcal{N}$ under an element of the variable group, and contains the same variables as $\mathcal{M}$. But this means that in our dominance check, we will discover that $\mathcal{M}$ is dominated by $\mathcal{N}$, contradicting the construction of $\mathcal{T}$.

Next we show that any full assignment corresponds to at least one leaf of $\mathbf{T}$. Let $\mathcal{A}$ be a full assignment, and let X_1 be the variable at the root of $\mathbf{T}$. Then for some $\alpha_1 \in \mathrm{Dom}(X_1)$ the literal $(X_1 = \alpha_1) \in \mathcal{A}$. Since the downedges from the root are labelled with orbit representatives of G on $\mathrm{Dom}(X_1)$, there exists $\beta_1 \in \mathrm{Dom}(X_1)$ and $x_1 \in G^{\mathrm{Var}}$ such that $(X_1 = \alpha_1)x_1 = (X_1 = \beta_1)$ and β_1 is the label of a downedge from the root. Thus $\mathcal{A}x_1$ contains a node of the tree at depth 1.

Suppose that $\mathcal{A}x_i$ contains a node $\mathcal{N}$ of the tree at depth i, and suppose that the label at $\mathcal{N}$ is X_{i+1}. Note that since $\mathcal{A}$ is a full assignment, we must have $(X_{i+1} = \alpha_{i+1}) \in \mathcal{A}$, for some $\alpha_{i+1} \in \mathrm{Dom}(X_{i+1})$. We subdivide into two cases.

If there exists β_{i+1} in the orbit of α_{i+1} under $G^{\mathrm{Val}}_{(\mathcal{A})}$ such that β_{i+1} is the label of a downedge from $\mathcal{N}$, then letting $x_{i+1} \in G^{\mathrm{Val}}_{(\mathcal{A})}$ map $\alpha_{i+1} \mapsto \beta_{i+1}$ we see that $\mathcal{A}x_ix_{i+1}$ contains a node at depth $i+1$.

Suppose instead that this is not the case. Then some orbit representative $(X_{i+1} = \beta_{i+1})$, in the orbit of $(X_{i+1} = \alpha_{i+1})$ under $G^{\mathrm{Val}}_{(\mathcal{N})}$, has been selected by the GE-tree technique but then rejected due to SBDD considerations. This means that there is a node $\mathcal{M}$ at depth $j \leq i+1$, which is to the left of $\mathcal{N}$ in $\mathbf{T}$, and which dominates $\mathcal{N} \cup (X_{i+1} = \beta_{i+1})$. Hence there exists an element $g \in G$ such that $(\mathcal{N} \cup (X_{i+1} = \beta_{i+1}))g$ contains $\mathcal{M}$. Let $h \in G^{\mathrm{Var}}_{(\mathcal{N})}$ map $(X_{i+1} = \alpha_{i+1})$ to $(X_{i+1} = \beta_{i+1})$. Then $\mathcal{A}x_ihg$ contains $\mathcal{M}$. Either it is the case that there is a node at depth $i+1$ below $\mathcal{M}$ which can be reached from $\mathcal{A}x_ihg$ using only value symmetries, or there exists a node $\mathcal{M}'$ to the left of $\mathcal{M}$ which dominates some descendant of $\mathcal{M}$ that is contained in an image of $\mathcal{A}x_ihg$. Since this "mapping to left" operation can only be carried out a finite number of times, at some stage we must find a node of depth $i+1$ which is contained in an image of $\mathcal{A}$.

We finish this section with a brief discussion of the expected efficiency gain of this technique over plain SBDD. With SBDD, the cost at each node $\mathcal{N}$ of

determining dominance is potentially exponential: for each completed subtree, we must determine whether or not it is possible to map the set of literals corresponding to the root of that subtree into the set of literals at $\mathcal{N}$. Best known algorithms for this run in moderately exponential time (that is, $O(e^{n^c})$ where $c < 1$ and n is the number of points that the group is acting on: namely $|\chi|$, the sum of the domain sizes). In our hybrid GE-tree and SBDD construction, the selection of a collection of orbit representatives for the down-edges for a node is done in low-degree polynomial time: there is a proof in [10] that the time is no worse than $O\tilde{}(n^4)$, but the actual bound is lower than this. We then perform SBDD on a reduced number of possible nodes, and with a smaller group.

5 Experiments

In [10] we showed that, for many CSPs, the cost of reformulating the problem into one with only value symmetry is clearly outweighed by the gains of the GE-tree construction, which is a polynomial-time method. In this paper we address the more standard question of CSPs with both value and variable symmetries. We take a highly symmetric CSP with known solutions, and compare symmetry breaking combinations.

The queens graph is a graph with n^2 nodes corresponding to squares of a chessboard. There is an edge between nodes iff they are on the same row, column, or diagonal, i.e. if two queens on those squares would attack each other in the absence of any intervening pieces. The colouring problem is to colour the queens graph with n colours. If possible, this corresponds to a set of n solutions to the n queens problem, forming a disjoint partition of the squares of the chessboard.

The problem is described by Martin Gardner. There is a construction for n = 6k+1 or 6k+5, i.e. where n is not divisiuble by either 2 or 3. Other cases are settled on a case by case basis. Our CSP model has the cells of an $n \times n$ array as variables, each having domain $1 \ldots n$. The variable symmetries are those of a square (the dihedral group of order 8). The value symmetries are the $n!$ permutations of the domain values (the symmetric group of degree n).

Our symmetry breaking approaches (using GAP–ECL^{i}PS^e) are:

- SBDD only;
- GE-tree construction for the value symmetries, with SBDD only used to check the symmetric equivalence of solutions (GEtree+iso);
- GE-tree construction for the value symmetries, with SBDD – on the full symmetry group for the CSP – at each node (GEtree+SBDDfull);
- GE-tree construction for the value symmetries, with SBDD – on the symmetry group for the variables – at each node (GEtree+SBDDval).

Table 1. gives the GAP–ECL^{i}PS^{e}cpu times for a range of values for n. It seems clear that GE-tree combined with full SBDD is competitive with SBDD only. It also seems clear that only using SBDD (or any other isomorph rejection method) is not competitive: the search tree is still large (since only value symmetries are being broken), with expensive variable symmetry breaking being postponed until

Table 1. Experimental results

n		5	6	7	8
solutions		1	0	1	0
SBDD	GAP	0.39	0.48	1.01	112.44
	ECL	0.19	0.48	7.35	814.92
	Σ	0.68	0.96	8.36	927.36
GEtree+iso	GAP	0.42	0.37	1.49	127.15
	ECL	0.09	0.35	12.01	1677.26
	Σ	0.51	0.72	13.50	1804.73
GEtree+SBDDfull	GAP	0.42	0.52	1.51	195.15
	ECL	0.06	0.27	6.79	935.74
	Σ	0.48	0.79	8.30	1131.19
GEtree+SBDDval	GAP	0.77	0.93	3.96	930.77
	ECL	0.03	0.32	6.65	1146.46
	Σ	0.80	1.25	10.61	2077.23

the entire tree is searched. Interestingly, the results for GE-tree construction combined with SBDD on the complement of the value symmetries are not as good as expected. This approach combines polynomial time value symmetry breaking with dominance detection in a smaller algebraic structure than the full symmetry group. Therefore, on a heuristic level, we expect faster symmetry breaking than for GE-trees and full-group SBDD. However, the combination of symmetry breaking methods is, in general, an unexplored research area; further investigation is needed in this area.

6 Conclusions

Symmetry breaking in constraint programming is an important area of interplay between artificial intelligence and symbolic computation. In this paper we have identified a number of important special structures and cases that can arise in the action of a symmetry group on the literals of a constraint problem. We have described a number of ways in which known symmetry breaking methods can be safely combined and described some new methods exploiting these newly identified structures.

Our initial experimental results demonstrate the applicability of our theoretical results, with more work to be done to overcome the apparent overheads of combining more than one symmetry breaking technique during the same search process. Other future work includes assessment of new heuristic approaches, including problem reformulation (to obtain a CSP with a more desirable symmetry group than that of a standard CSP model) and using dominance detection only at selected nodes in the tree (as opposed to every node, as currently implemented). We also aim to investigate both the theoretical and practical aspects of further useful ways of decomposing the symmetry group of a CSP.

Acknowledgements

The authors would like to thank Ian P. Gent for extremely helpful discussions. Our work is supported by EPSRC grants GR/R29666 and GR/S30580.

References

1. A.M. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, and T. Walsh. Global constraints for lexicographic orderings. In P. van Hentenryck, editor, *Proceedings of the Eighth International Conference on Principles and Practice of Constraint Programming*, volume 2470 of *Lecture Notes in Computer Science*, pages 93–108. Springer, 2002.
2. R. Backofen and S. Will. Excluding symmetries in constraint-based search. In *Proceedings, CP-99*, pages 73–87. Springer, 1999.
3. I.P. Gent and B.M. Smith. Symmetry breaking in constraint programming. In W. Horn, editor, *Proc. ECAI 2000*, pages 599–603. IOS Press, 2000.
4. Torsten Fahle, Stefan Schamberger, and Meinolf Sellmann. Symmetry breaking. In T. Walsh, editor, *Proc. CP 2001*, pages 93–107, 2001.
5. Filippo Focacci and Michaela Milano. Global cut framework for removing symmetries. In T. Walsh, editor, *Proc. CP 2001*, pages 77–92, 2001.
6. Ian P. Gent, Warwick Harvey, and Tom Kelsey. Groups and constraints: Symmetry breaking during search. In Pascal Van Hentenryck, editor, *Proceedings of the Eighth International Conference on Principles and Practice of Constraint Programming — CP'2002*, volume 2470 of *Lecture Notes in Computer Science*, pages 415–430. Springer, 2002.
7. Ian P. Gent, Warwick Harvey, Tom Kelsey, and Steve Linton. Generic SBDD using computational group theory. In Francesca Rossi, editor, *Proceedings of the Ninth International Conference on Principles and Practice of Constraint Programming — CP'2003*, volume 2833 of *Lecture Notes in Computer Science*, pages 333–347. Springer, 2003.
8. M. G. Wallace, S. Novello, and J. Schimpf. ECLiPSe : A platform for constraint logic programming. *ICL Systems Journal*, 12(1):159–200, May 1997.
9. The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.2*, 2000. `(http://www.gap-system.org)`.
10. Colva M. Roney-Dougal, Ian P. Gent, Tom Kelsey, and Steve A. Linton. Tractable symmetry breaking using restricted search trees. In *Proceedings, ECAI–04*, 2004. To appear.
11. Akos Seress. *Permutation group algorithms.* Number 152 in Cambridge tracts in mathematics. Cambridge University Press, 2002.