Reducing The Number of Constraints Needed During Symmetry Breaking During Search

Justin Pearson

Uppsala University, Information Technology , Department of Computer Systems , Box 337, SE-751 05 Uppsala, Sweden , justin@docs.uu.se

Abstract. Symmetry Breaking During Search (SBDS) is a method to reduce the search space of constraint satisfaction problems by taking into account symmetries inherent in the problem. SBDS works by adding extra constraints during search which disallow symmetric search path being explored. At each node during the search tree SBDS requires an extra constraint for each symmetry of the problem to be applied to a partial assignment. As symmetry groups of problems get large, SBDS can become impractical and some method is needed to prune symmetries. In this paper we give a group theoretical characterization of when symmetries applied to partial assignments are equivalent. This method can then be integrated into SBDS.

1 Introduction and Background

Many Constraint Satisfaction Problems (CSPs) have a great deal of symmetry: that is solutions (and non solutions) are related by bijections on the set of candidate solutions. A symmetry is a bijection on the set of candidate solutions which maps solutions to solutions. For example in solving the n-queens problem, solutions are related by rotation and reflection. Or when scheduling a sporting tournament, often the constraints allow the set of initial plays to be fixed, or that weeks (or periods) are indistinguishable. While analyzing protein folding many solutions are related by the symmetries of three dimensional space [1]. There has recently been a great deal of interest in modifying search procedures to take into account the symmetries in a problem and to stop a search procedure exploring a previously explored symmetrically equivalent paths [11, 2, 9], an early precursor of this work can be found in [3].

The methods in [11, 2, 9] add constraints dynamically during search. At each node during search a constraint (depending on the current partial assignment) has to be added for each symmetry of the problem. It is easy to show that the set of symmetries of a CSP form a group. One of the interesting facts of group theory is that a small number of generators can generate an exponential number of symmetries: for example the set of all permutations on n letters is of size $n!$; or the number of row and column symmetries of an n by m matrix (see [8]) is of size $n!m!$. Both these groups can be generated by a polynomial number of generators. Consequentially methods such as Symmetry Breaking During search (SBDS) [11] can become impractical to implement fully even on very small problems [14]. It should be emphasized that even though breaking all the symmetries of a problem, using methods such as SBDS, can become impractical, large reductions in the search space can still be made using subsets of the set of all symmetries (see [14]).

This paper starts from an observation in [6] that certain symmetry breaking predicates (lexicographic ordering predicates) are equivalent when restricted to a subset of the variables of the problem. The work in [6] shows how to build predicates for boolean problems which force only one solution for each symmetrically equivalent solution, and then using group theoretical methods it is shown how it is possible to reduce the number of symmetry breaking predicates required. This method is a static method that is applied before any search is started.

In this paper we extend SBDS [11] to reduce the number of symmetries required during search. This is done by extending the analysis of symmetry breaking predicates in [6] to include more general symmetry breaking predicates. The analysis is extended in a number of ways: most importantly the analysis is extended to arbitrary symmetries not just permutations on variables. It is also shown that using standard methods from computational group theory [4, 5, 10] that the method can be integrated into the search procedure and most importantly can be applied to arbitrary search orders (even dynamically changing ones).

This paper is organized as follows: we give a very brief and informal introduction to SBDS, then we introduce the notation and definitions concerning CSPs required for the rest of the paper, we define a notion of a symmetry of a CSP and the notion of applying a symmetry to a partial solution, we then prove the main theorem of the paper characterizing when symmetries are equal via the notion of cosets, finally we talk about a prototype implementations and discuss some examples and experiments where symmetry is reduced.

2 Symmetry Breaking During Search

In this section we give a very brief introduction to Symmetry Breaking During Search [11] (most of the results in this paper could be applied to the method found in [2], but for simplicity we only present the results applied to SBDS).

SBDS assumes that the search procedure builds a search tree, where at each branching point a choice is made between some variable v being equal to some value d or not equal to that value d. SBDS works by adding, for each symmetry σ, at each node in the search tree a constraint of the form:

$$H \wedge \sigma(H) \wedge v \neq d \Rightarrow \sigma(v \neq d)$$

where H is the current partial assignment, v is the next variable to be assigned, $\sigma(H)$ is the result of applying the symmetry to the partial assignment (this is defined formally in Definition 6). The constraint $v \neq d$ forces the variable v not to be equal to d finally and $\sigma(v \neq d)$ is the symmetric version of the $v \neq d$ constraint. It can then be shown that adding these constraints during search is sufficient to break all symmetries: that is the search procedure is guaranteed to only find one member of each symmetrically equivalent classes. SBDS works by excluding paths that are symmetrically equivalent to previously explored paths. The added constraints exclude solutions below the $v \neq d$ branch which are symmetric and to find a solution if a solution exists.

To illustrate how symmetries act equivalently on partial assignments, we consider a simple example. Suppose we have the partial assignment (v_1,v_2 are variables and d_1, d_2 are domain elements):

$$H = v_1 \mapsto d_1 \wedge v_2 \mapsto d_2$$

further suppose as symmetries of the problem include the following two symmetries (written as products of cycles and as actions on the variables)

$$\sigma_1 = (v_1, v_3)(v_4, v_5)$$

$$\sigma_2 = (v_1, v_3, v_4)(v_5, v_6)$$

then applying σ_1 and σ_2 to the partial assignment H we get that $\sigma_1(v_1) \mapsto d_1 \wedge \sigma_1(v_2) \mapsto d_2$ and $\sigma_2(v_1) \mapsto d_1 \wedge \sigma_2(v_2) \mapsto d_2$ are *both* equal to

$$v_3 \mapsto d_1 \wedge v_2 \mapsto d_2$$

3 Constraint Satisfaction Problems and Symmetry

In this section we introduce the notation needed to analyse symmetries of CSPs. A very general notion of symmetry is defined, which includes: permutations on variables, permutations on domain elements and symmetries that depend both on variables and domain elements (such as interchangeable elements).

Definition 1. *A* Constraint Satisfaction Problem *(CSP) is a triple:*

$$(V, D, \{R_i(S_i)\}_{i \in I})$$

where for every i in I and each R_i is a subset of the product D^{k_i} for some k_i and S_i is a tuple of elements from V of length k_i.

Normally V is referred to as the set of *variables* and D is referred to as the *domain* of the problem. The pairs $R_i(S_i)$ are constraints and the tuple S_i is often called the scope of the constraint.

Example 1. Given a graph $G = (V, E)$ where E is a subset of $V \times V$ (that is a set of pairs of vertices) and a domain D, the colouring problem with colours from domain D is the CSP

$$(V, D, \{\neq_D (\langle v_1, v_2 \rangle)\}_{\langle v_1, v_2 \rangle \in E})$$

where $\neq_D$ is the relation $\{\langle d_1, d_2 \rangle \mid d_1, d_2 \in D \wedge d_1 \neq d_2\}$. That is for each edge in the graph we have a constraint saying that the two vertices cannot be of the same colour.

Definition 2. *A* solution *of a CSP* $(V, D, \{R_i(S_i)\}_{i \in I})$ *is a map*

$$h : V \to D$$

such that for all i

$$h(S_i) \in R_i$$

where h is extended to a map on tuples of V in the natural way[1]

Then a solution to a CSP is an assignment of variables to domain values such that every constraint is satisfied.

The set of solutions to a constraint satisfaction problem $\mathcal{P}$ will be denoted as $\mathrm{Sol}(\mathcal{P})$. The set of candidate solutions of a problem is the set of all maps h from V to D.

Example 2. Continuing Example 1 a solution is a map $h : V \to D$ such that for every edge $\langle v_i, v_j \rangle$ in the graph $h(v_i) \neq h(v_j)$

A symmetry of a constraint satisfaction problem is a mapping on the set of candidate solutions which takes solutions to solutions; often only symmetries that permute variables or domain elements are considered, these will be referred to as permutation symmetries. The great advantage of permutation symmetries (see below) is that there is a large body of work in computational group theory (see [4]) concerned with efficient algorithms for decision problems and computing with permutation symmetries. In Example 3 it is shown there are symmetries which don't just permute variables or domain elements, but some mixture depending on the map in question.

To define symmetries on constraint satisfaction problems we will first look at symmetry groups on functions. The set of functions from V to D, which will be denoted as $\mathrm{Hom}(V, D)$[2].

Definition 3. *Define* $\mathrm{Sym}(\mathrm{Hom}(V, D))$ *to be the subset, of* $\mathrm{Hom}(V, D)$*, of all bijections (one-to-one and onto function).*

It is easy to show that:

Proposition 1. *The set* $\mathrm{Sym}(\mathrm{Hom}(V, D))$ *is a group under function composition.*

A symmetry is map σ from $\mathrm{Sym}(\mathrm{Hom}(V, D)$ is a map

$$\sigma : \mathrm{Hom}(V, D) \to \mathrm{Hom}(V, D)$$

such that for all f and g in $\mathrm{Hom}(V, D)$, $\sigma(f) = \sigma(g)$ implies $f = g$ (remember σ is a function that takes functions to functions) and for all $g : V \to D$ there exists

[1] That is given a tuple $\langle d_1, \ldots, d_n \rangle$, $h(\langle d_1, \ldots, d_n \rangle)$ is equal to the tuple

$$\langle h(d_1), \ldots, h(d_n) \rangle$$

.

[2] Hom is short for homomorphism

some f such that $\sigma(f) = g$. Requiring that the set of functions forms a group under composition amounts to requiring an identity function, and for every σ we require an inverse: one can form inverses because we require the maps to be bijections on the set $\mathrm{Hom}(V, D)$.

Definition 4. *A* symmetry *of a CSP* $(V, D, \{R_i(S_i)\}_{i\in I})$ *is a bijection* σ *from* $\mathrm{Sym}(\mathrm{Hom}(V, D))$*, such that for all solutions* h *of* $\mathcal{P}$ *the map* $\sigma(h)$ *is a solution of* $\mathcal{P}$*.*

The set of symmetries of a CSP $(V, D, \{R_i(S_i)\}_{i\in I})$ forms a sub-group of $\mathrm{Sym}(\mathrm{Hom}(V, D))$, this group will be referred to as the *symmetry group* of the CSP.

Example 3. Given the constraint problem:

$$(\{v_1, v_2, v_3\}, \{0, 1, 2\}, \{R(\langle v_1, v_2, v_3\rangle)\})$$

where R is the relation containing the tuples $\langle 1, 2, 0\rangle$, $\langle 2, 1, 0\rangle$, $\langle 2, 1, 1\rangle$, $\langle 1, 2, 1\rangle$, $\langle 1, 2, 2\rangle$, and $\langle 2, 1, 2\rangle$. The symmetry group of this CSP is generated by the following symmetries:

$$\sigma_1(h)(v) = \begin{cases} h(v_2) & \text{if } v = v_1 \\ h(v_1) & \text{if } v = v_2 \\ h(v) & \text{otherwise} \end{cases}$$

and

$$\sigma_2(h)(v) = \begin{cases} h(v) & \text{if } v = v_1 \text{ or } v = v_2 \\ (h(v) + 1) \bmod 3 & \text{if } v = v_3 \end{cases}$$

Example 4. Continuing Example 1 if we think of a domain D of size n as the group of integers modulo n, $\mathbf{Z}_n$, then the functions σ_k:

$$\sigma_k(h)(v) = (h(v) + k) \bmod n$$

for each $k \in \mathbf{Z}_n$ is a symmetry of the associated CSP.

Example 5. In [12] a symmetry of a CSP is defined as a permutation $\sigma_0 : V \to V$ of the variable set and a set of maps

$$\{\sigma_v : D \to D\}_{v\in V}$$

This definition only differs from the definition in [12] in that here all the variables have the same domain. Further in [12] a notation of applying a symmetry to a relation is defined. Such a symmetry of the above form can be turned into a map $\sigma \in \mathrm{Sym}(\mathrm{Hom}(V, D))$ by defining

$$\sigma(h)(v) = \sigma_v(h(\sigma_0(v)))$$

a simple counting argument shows that the class in [12] is a subclass of the symmetries considered here.

An important sub-class of symmetries of a CSP is the set of symmetries which only change the role of variables in solution. For example given an assignment $h : V \to D$, a symmetric version, $\sigma(h)$ might have the property $\sigma(h)(v_1) = h(v_2)$, if this is true for all maps h, then the symmetry just permutes variables. These symmetries will be referred as variable permutation symmetries.

Definition 5. *A* variable permutation symmetry*, σ_π, is a function*

$$\sigma_\pi \in \mathrm{Sym}(\mathrm{Hom}(V, D))$$

of the form (where h is in $\mathrm{Hom}(V, D)$*):*

$$\sigma_\pi(h) = h \circ \pi$$

where π is a permutation of V.

It is important to check that any function of the form σ_π is indeed a bijection on the set $\mathrm{Hom}(V, D)$. To see that it is in injective we check that

$$\sigma_\pi(f) = \sigma_\pi(g)$$

implies that $f = g$. This is true since π is a bijection on V and $f \circ \pi = g \circ \pi$ implies that f is equal to g. To check surjectiveness we have to solve the equation:

$$h \circ \pi = g$$

for all g; setting f to be $g \circ \pi^{-1}$ solves the equation. The set of all permutation symmetries forms a subgroup of $\mathrm{Sym}(\mathrm{Hom}(V, D))$.

Example 6. Given a graph and its associated CSP as in Example 1, automorphisms of the graph give rise to symmetries. Given a graph $(V, E \subseteq V \times V)$ a graph automorphism is a bijection $f : V \to V$ such that for all edges $\langle v_i, v_j \rangle$ we have that the edge $\langle f(v_i), f(v_j) \rangle$ is an edge of the graph. Given an automorphism, f, of the graph we can build a symmetry $\sigma_f \in \mathrm{Sym}(\mathrm{Hom}(V, D))$ of the associated CSP by define the function σ_f such that for all h:

$$\sigma_f(h)(v) = h(f(v))$$

That is a graph automorphism gives rise to a permutation of the variables which takes solutions to solutions.

The set of symmetries that act equivalently on a set of variables will be shown to be a coset of the symmetry group. This will allow methods from computational group theory to be used to find equivalent sets of symmetries.

4 Cosets and Equivalent Partial Symmetries

During search, symmetries are applied to partial solutions; the key observation in [6] is that many symmetries do the same thing on certain partial solutions.

This notion can be captured group theoretically by the notion of cosets, thus opening the way to applying methods from computational group theory to the optimization of symmetry exclusion predicates.

Given a bijection

$$\sigma : V \to D$$

which is a symmetry of a CSP $(V, D, \{R_i(S_i)\}_{i\in I})$ then during symmetry breaking we have to apply symmetries to partial solutions. A *partial assignment* is a pair (h', V') where h' is a map from $V' \to D$ and V' is a subset of V. To such a partial assignment we can associate a constraint

$$\{\langle h'(v_1), \ldots, h'(v_k)\rangle\}(\langle v_1, \ldots, v_k\rangle)$$

where $V' = \{v_1, \ldots, v_k\}$.

The following definition defines what it means for a symmetry to be applied to a constraint. The definition is complicated by the fact we have to consider how a symmetry acts not only on the scope of the constraint but how it acts on all the variables. This is because a symmetry might map a variable outside the scope of the constraint.

Definition 6. *Given a constraint $R_i(S_i)$ and a symmetry,*

$$\sigma \in \mathrm{Sym}(\mathrm{Hom}(V, D))$$

the constraint $\sigma(R_i(S_i))$ is the constraint over the variables $\langle v_1, \ldots, v_n\rangle$ (where $V = \{v_1, \ldots, v_n\}$ and $S_i = \{v_1, \ldots, v_k\}$) defined by the relation:

$$\{\langle \sigma(h)(v_1), \ldots, \sigma(h)(v_i), \ldots, \sigma(h)(v_n)\rangle \mid \langle h(v_1), \ldots, h(v_k)\rangle \in R_i\}$$

The new constraint has possibly a larger scope than the original constraint, it is then obtained by considering all assignments that are compatible with the elements of the relation and applying the symmetry σ to that map.

Although this definition is quite complicated and does not present an algorithm; for certain classes of symmetries it is often easy to calculate the effect of a symmetry on a partial assignment. Further it is often the case that the scope of the new constraint can be reduced without effecting the constraint.

Example 7. Suppose we have the variable set $V = \{v_1, \ldots, v_n\}$ and a permutation π on V which swaps v_1 and v_3. Given the partial assignment:

$$v_1 \mapsto d_1 \wedge v_2 \mapsto d_2$$

Then applying a symmetry σ to the partial assignment results in the constraint $\sigma(v_1 = d_1 \wedge v_2 = d_2)$ being equal to the constraint:

$$v_3 = d_1 \wedge v_2 = d_2$$

The reason why we can reduce the scope of the new constraint to $\langle v_2, v_3\rangle$ is that since we look at *all* possible maps which reduce to the constraint (as in Definition 6) the new constraint allows any possible value on v_1 and any v_i where i is not equal to 2 or 3.

We also need to define the action of a symmetry σ on the constraint $v \neq d$ (where $v \in V$ and $d \in D$) in the same way, setting $\sigma(v \neq d)$ to be

$$\{\langle \sigma(h)(v_1), \ldots, \sigma(h)(v_n)\rangle \mid h \in \mathrm{Hom}(V, D) \text{ and } \ h(v) \neq d\}$$

It is easy to show that for any σ and partial assignment (h', V') with the associated constraint H we have that

$$\sigma(H) \wedge \sigma(v \neq d) = \sigma(H \wedge v \neq d)$$

where $H \wedge v \neq d$ is the constraint representing partial assignment H and the constraint that v is not equal to d ($\wedge$ here is been used for the join of two constraints). This property of symmetries was required as an assumption in [11] and here is a consequence of the definition of what it means for a symmetry to be applied to a constraint.

Definition 7. *Given a symmetry, $\sigma \in \mathrm{Sym}(\mathrm{Hom}(V, D))$, we say that it* fixes *a variable $v \in V$ if for* all *maps $h : V \to D$ we have that*

$$h(v) = \sigma(h)(v)$$

The set of symmetries which fix a particular variable forms a group under composition. Let G denote the group of symmetries of a CSP $(V, D, \{R_i(S_i)\}_{i \in I})$ then we will use

$$G_{\{v_1, \ldots v_k\}}$$

to denote the subgroup of symmetries which fix the variables $v_1, \ldots, v_k$. If M and N are sets of variables and $M \subseteq N$ then G_N is a subgroup of G_M.

Proposition 2. *Suppose two symmetries σ and σ' both fix the same set of variables $M = \{v_1, \ldots, v_k\}$ and (h', M) is a partial assignment with an associated constraint H, then*

$$\sigma(H) = \sigma(H)$$

(where $\sigma(H)$ and $\sigma'(H)$ are as in Definition 6).

Proof: From Definition 6 the set $\sigma(H)$ is the set of partial assignments that when restricted to M agree with h' it follows that $\sigma(H)$ is equal to $\sigma(H')$ since σ fixes all the variables in M. $\square$

Given a group G with a subgroup $G' \subseteq G$ a *coset* of G' in G is a set of the form

$$G'\phi = \{\phi \circ \sigma \mid \sigma \in G'\}$$

Where $\circ$ is the composition of elements in the group; most of the groups in this paper are represented as groups of functions so $\circ$ is simply function composition.

Example 8. The group generated by the the permutations $(1, 2, 3)$, $(2, 3, 4)$ has 12 elements, the subgroup generated by $(1, 2, 3)$ has the three elements $()$

(denoting the identity) , $(1,2,3)$ and $(1,3,2)$. The set of right cosets of this subgroup in the full group, is the following 4 sets:

$$\{(),(1,2,3),(1,3,2)\},\{(2,4,3),(1,2)(3,4),(1,4,3)\}$$

$$\{(2,3,4),(1,3)(2,4),(1,4,2)\}$$

and

$$\{(1,2,4),(1,3,4),(1,4)(2,3)\}$$

notice the sets are disjoint and every element in the group is contained in some coset.

Some elementary facts about cosets are useful: cosets are either disjoint or equal (that is $G'\phi \cap G'\psi$ is non-empty implies that $G'\phi = G'\psi$); all cosets are the same size, and if G is finite then the number of distinct cosets of $G' \subseteq G$ is equal to $|G|/|G'|$ (where $|G|$ denotes the number of elements in the set G).

We now come to a very simple lemma which allows us to characterize group theoretically when symmetry breaking constraints do the same thing on partial assignments.

Lemma 1. *If G is a subgroup of symmetries $\mathrm{Sym}(\mathrm{Hom}(V,D))$ and $N \subseteq V$ and G_N is the subgroup of G which fixes all the variables in N (as in definition 7) then all symmetries:*

$$\phi \circ \sigma \in G_N\phi$$

act identically on the variables in N. That is for all $v \in N$ and all maps h in $\mathrm{Hom}(V,D)$ and for all pairs of functions $\phi \circ \sigma$ and $\phi \circ \sigma'$ from $G_N\phi$ we have that

$$(\phi(\sigma)(h))(v) = (\phi(\sigma')(h))(v)$$

for all $v \in N$. Remember ϕ, σ and σ' are functions that take functions to functions so for example $(\phi(\sigma)(h))$ is a function from V to D.

This lemma states that all symmetry functions from the coset $G_N\phi$ act in the same way on the variables from the set N. We will use this later to restrict the number of constraints we have to introduce during search, since our partial assignments don't concern all variables.

Proof: The proof of this is almost trivial since we know that σ and σ' both fix the variables in N, so for v in N we have that $h(v) = \sigma(h)(v) = \sigma'(h)(v)$, then on the variable v we have that $\phi(\sigma(h))(v) = \phi(h)(v) = \phi(\sigma'(h))(v)$ for v in N. □

Thus the symmetries from the cosets $G_N\phi$ on the variables in N are determined by ϕ. Note that any element in $\sigma \in G_N\phi$ acts equivalently on the variables in N, so it does not matter which representative of the coset is picked.

Theorem 1. *For a partial assignment H on the variables N and for all $\phi \circ \sigma$ and $\phi \circ \sigma'$ from the coset*

$$G_{N\cup\{v\}}\phi$$

the constraints

$$H \wedge \phi \circ \sigma(H) \wedge v \neq d \Rightarrow \phi \circ \sigma(v \neq d)$$

and

$$H \wedge \phi \circ \sigma'(H) \wedge v \neq d \Rightarrow \phi \circ \sigma'(v \neq d)$$

are equal.

Proof: This theorem follows from Lemma 1 and Proposition 2. □

What this theorem says that, as we add constraints during search, if the partial assignment only concerns the variables in N then we only have to add one constraint from each coset $G_N\phi$.

5 Algorithms from Computational Group Theory

This section gives some very brief pointers to some algorithms in computational group theory and how they could be integrated into a search procedure such as SBDS. There is a great deal of research into computational group theory see for example [4, 10]). The Schreier–Sims method [4, 5] along with various refinements allows the cosets of sub-groups fixing elements to be calculated from the set of generators of a group. The algorithm works incrementally starting at the first element to be fixed, and then generating a set of strong-generators (these are generators of the group which fix the elements in a particular order) of the group G_N as defined in Definition 7. Even better the algorithm can be modified so that the order that the variables are chosen is not fixed in advance. This method could be integrated into some symmetry-breaking backtrack search procedure such as SBDS, and could be modified to take into account dynamic search orders.

6 Examples of reductions

The number of symmetry breaking predicates that are needed can be calculated using the tool GAP [10] for computational group theory. In all these examples we will look at variable permutation symmetries. Further some experiments have been performed using a prototype implementation of SBDS and reduced SBDS using GNU-Prolog and GAP [10]. The implementation is a GAP program that generates a specialized search procedure for a particular symmetry group either for full SBDS or reduced SBDS. At present all the search procedures for each level in the search tree have to be generated before search, obviously a more efficient method would be to calculate the search predicates on the fly when needed.

Example 9. Suppose $G(n)$ is the group of all permutations on n variables, then when assigning the first variable in the search tree only $n$predicates are needed (regardless of the variable picked), at the second variable $n * (n-1)$ predicates are needed, at the $k+1$ variable, $n*(n-1)*\cdots*(n-k)$ predicates are needed.

Example 10. A concrete example, imagine 9 variables arranged in a 3×3 square and we are allowed to swap any two rows or any two columns (such problems are studied in [8]). This group of row and column swaps has 36 elements in total. It can be shown that if the variables are searched starting at the first row, going along each row and to the next row, then the number of symmetry breaking predicates needed to be introduced at each following level is $9, 18, 18, 36$ and then 36 at each level.

The order that the variables are searched affects the number of symmetry breaking predicates that need to be introduced at each level. Finding the ordering of the variables which introduce the smallest number of symmetry breaking predicates is closely related to the problem of finding a minimal base in a group, which is known to be NP-complete (see [4]). This opens a whole new area of heuristics concerning picking a variable ordering which minimizes the number of symmetry predicates that need to be introduced. The following example gives a simple illustration.

Example 11. Given the permutation group G generated by the generators $(6,7)$, $(3,4,5)$, $(3,5)$, $(5,6)$, and $(1,2)$, this group has 240 elements. Just looking at fixing one element the size of the group $G_{\{1\}}$ is 120, thus there would be 2 symmetry breaking predicates at the first level, while the size of the group $G_{\{5\}}$ is 48, thus there would be 5 predicates. When using reduced SBDS search for equivalence classes of 0/1 vectors of length 7 equivalent under the action of this group, with the variable ordering $1, 2, 3, 4, 5, 6, 7$, it was found that 9725 constraints were asserted, and then with the variable ordering with 5.2.3.4.1.6.7 it was found that 10545 constraints were asserted. It would be interesting to understand how variable ordering heuristics interact with the symmetry group of the problem.

Example 12. The pigeon hole problem is the problem of putting M items into N holes together with the constraint that each hole can only contain one item. Although it is rather obvious that there no solution if M is greater than N many constraint systems have a hard time finding this fact. The results in Table 1 give the number of constraints posted by SBDS and reduced SBDS for various combinations of $N$and M. The last column in the table refers to the number of constraints posted when symmetry breaking constraints are only posted when the number of constraints is greater than the size of the remaining search space. For example with 4 holes and 5 items, at the third hole, $24 = 4!$ constraints should be asserted for each choice point in the search tree, but at this point there are only $25 = 5^2$ possible assignments left, so no symmetry breaking constraints were posted. Finally the instance with 13 holes and 14 items was tried, without any symmetry breaking the search took more than 8 hours. The symmetry group with of the instance with 13 holes is of size $13! = 6227020800$, this is too many constraints to post for full SBDS, even with reduced SBDS after 4 variables there would be more than 17160 constraints. Instead only the first 50 constraints at each level were posted during search, it should be emphasized that these constraints will act differently on the variables at each level. Then to solve the

13 hole problem, the solver took about 17[3] seconds. As a comparison with full SBDS the first 50 constraints of the group were posted at each level, to solve the same instance the solve took about 96 minutes, these results are summarized in Table 2. Of course the 50 constraints in the full SBDS based search could have been picked more carefully and maybe a greater reduction could have been achieved, but it is interesting that even been very naive significant reductions in search time can be obtained with reduced SBDS.

Holes	Items	SBDS constraints	Reduced SBDS constraints	Further reduced
4	5	69	37	14
5	6	476	201	82
6	7	???	???	153
7	8	???	???	1095

Table 1. Results for the Pigeon hole problem (??? - means that there were too many constraints for the implementation to handle)

Hole	Items	Full Search Time	Constraints Asserted	Reduced Search Time
7	8	0.24	1095	0.02
8	9	2.57	2076^*	0.01
9	10	30.34	18724^*	0.01
10	11	390	187294^*	0.01
13	14	>8 Hours	173480^*	16.94

Table 2. Times for the Pigeon hole Problem (All times in Seconds) * means no more than 50 asserted at each level)

Example 13. A Steiner triple system is a pair $(X, \mathcal{B})$ where $\mathcal{B}$ a set of subsets of X of size 3, together with the constraints that the sets in $\mathcal{B}$ are different and that any pair of elements from X are in a unique set $B \in \mathcal{B}$. Here the problem has been modeled naively to illustrate the difference in the number of constraints asserted with SBDS and the reduced SBDS. A Steiner triple system $(X, \mathcal{B})$ is modeled as a list of $3 * |\mathcal{B}|$ variables: that is a triple of variables for each set. In modeling the symmetries of the problem only the fact that within each triple of variables the variables are interchangeable is modeled, but not the fact that the triples themselves are interchangeable. The results are presented in Table 3, show a significant reduction in the number of symmetry breaking predicates that

[3] All these experiments were carried out on a 800Mhz Pentium III PC, runing Linux. All the search was done using compiled Gnu-Prolog code.

have to be asserted. Note in this case the same amount of symmetry was broken using full SBDS and Reduced SBDS since all the required symmetry breaking predicates were asserted.

Domain Size	Number of Sets	SBDS Asserted	Reduced Asserted
3	3	645	48
4	3	5375	350
5	3	36550	6860
5	4	???	24448

Table 3. Steiner Type Problems (??? - means that there were too many constraints for the implementation to handle)

7 Conclusion, Related and Further Work

SBDS[11] and the work in [2] both work well when the total number of symmetries is small. When the symmetry group gets large as in the social golfer problem [13] or as in the symmetry of Matrix problems [8, 14] methods specialized to the particular problem are required. The recent work of Fahle et.al. [7] gives a method where symmetry can be excluded by the notation of dominance detection: which essentially checks if a symmetric assignment has been visited earlier in the search tree. This dominance check has to specially created for the problem. While SBDS and reduced SBDS presented here can be automated once the symmetry group of the problem is known.

In this paper a group theoretical analysis, via the notion of cosets, has then been applied to symmetry breaking predicates. This analysis extends the analysis of symmetry breaking predicates of propositional theories found in [6]. Further, this analysis has been applied to the predicates posted during search in the Symmetry Breaking During Search Method [11]; it has been shown that it is possible to reduce the number of extra constraints that need to be posted during search while still breaking all symmetries. Further this method could be extended to the symmetry breaking method found in [2]. Further in [6] there is a further reduction which depends on implications between lexicographic symmetry breaking constraints from cosets. It is not clear how to extend this analysis to the types of predicates considered in this paper, but work is on the way.

Acknowledgements

I am very grateful for Pierre Flener for reading an early draft of this paper.

References

1. Rolf Backofen. Constraint techniques for solving the protein structure prediction problem. In Michael Maher and Jean-Francois Puget, editors, *Proceedings of* 4th *International Conference on Principles and Practice of Constraint Programming (CP'98)*, volume 1520 of *Lecture Notes in Computer Science*, pages 72–86. Springer Verlag, 1998.
2. Rolf Backofen and Sebastian Will. Excluding symmetries in constraint-based search. In Joxan Jaffar, editor, *Proceedings of* 5th *International Conference on Principles and Practice of Constraint Programming (CP'99)*, volume 1713 of *Lecture Notes in Computer Science*, pages 73–87, Berlin, 1999. Springer–Verlag.
3. C. A. Brown, L. Finkelstein, and P. W. Purdom Jr. Backtrack searching in the presence of symmetry. In *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes,*, volume 357 of *LNCS*, pages 99–110. Springer-Verlag, 1988.
4. G Butler. *Fundamental Algorithms for Permutation Groups*, volume 559 of *Lecture Notes in Computer Science*. Springer Verlag, 1991.
5. Peter J. Cameron. *Permutation Groups*. Number 45 in London Mathematical Society Student Texts. Cambride University Press, 1999.
6. J. Crawford, M. Ginsberg, E. Luks, and A. Roy. Exploiting symmetry by adding symmetry breaking predicates. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR–96)*, Boston, MA, 1996.
7. Schamberger Fahle, Torsten, Stefan, and Meinolf Sellmann. Symmetry breaking. In Toby Walsh, editor, *Principles and Practice of Constraint Programmingo*, volume 2293 of *LNCS*, pages 93–107. Springer Verlag, 2001.
8. Pierre Flener, Alan Frisch, Brahim Hnich, Zeynep Kiziltan, Ian Miguel, Justin Pearson, and Toby Walsh. Symmetry in matrix models. Presented at SymCon'01 a workshop at the 7th International Conference on Principles and Practice of Constraint Programming proceedings available via `http://http://www.csd.uu.se/~pierref/astra/symmetry/`, 2001.
9. Filippo Focacci and Michaela Milano. Global cut framework for removing symmetries. In Toby Walsh, editor, *Principles and Practice of Constraint Programming*, volume 2239 of *LNCS*, pages 77–92. Springer Verlag, 2001.
10. The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.2*, 2000. (`http://www.gap-system.org`).
11. Ian P. Gent and Smith Barbara. Symmetry breaking during search in constraint programming. In *Proceedings of ECAI2000*, pages 599–603, 2000.
12. P. Meseguer and C. Torras. Solving strategies for highly symmetric CSPs. In *Proceedings IJAJ'99*, pages 400–405, 1999.
13. Meinolf Sellmann and Warwick Harvey. Heuristic constraint propagation – Using local search for incomplete pruning and domain filtering of redundant constraints for the social golfer problem. In Narendra Jussien and François Laburthe, editors, *Proceedings of the Fourth International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimisation Problems (CP-AI-OR'02)*, pages 191–204, Le Croisic, France, March, 25–27 2002.
14. Barbara M. Smith and Ian P. Gent. Reducing symmetry in matrix models: SBDS vs. constraints. Presented at SymCon'01 a workshop at the 7th International Conference on Principles and Practice of Constraint Programming proceedings available via `http://www.csd.uu.se/~pierref/astra/symmetry/`, 2001.