# An Efficient Way of Breaking Value Symmetries

**Jean-François Puget**
ILOG,
9 avenue de Verdun,
94253 Gentilly, France,
puget@ilog.fr

### Abstract

Several methods for breaking value symmetries have been proposed recently in the constraint programming community. They can be used in conjunction with variable symmetry breaking methods. However, this combination does not break all symmetries in general. We present a combination of lex constraints and element constraints that can be used to break all combinations of variable and value symmetries. It is the first time to our knowledge that it is possible to break all combinations of value and variable symmetries by adding constraints. This method is quite efficient when the number of symmetries is not too large, as shown by experiments using graceful graph problems. We also present a new global constraint that deals with the case where there are too many value symmetries. Experiments show that this is highly effective.

## Introduction

Symmetries are mappings of a Constraint Satisfaction Problem (CSP) onto itself that preserve its structure as well as its solutions. If a CSP has some symmetry, then all symmetrical variants of every dead end encountered during the search may have to be explored before a solution can be found. Even if the problem is easy to solve, all symmetrical variants of a solution are also solutions, and listing all of them may just be impossible in practice. Breaking symmetry methods try to cure these issues.

Among symmetries, two categories have been studied in detail: variable symmetries, and value symmetries. A variable symmetry is a permutation of variables that leave a given CSP invariant. A value symmetry is a permutation of values that leave the CSP invariant. Both kind of symmetries can be combined.

Many methods have been proposed for breaking variable symmetries, including SBDD(Focacci and Milano 2001)(Fahle, Shamberger, and Sellmann 2001)(Gent et al. 2003), SBDS (Gent, Harvey, and Kelsey 2002), and adding constraints before search (Crawford et al. 1996)(Flener et al. 2002)(Aloul, Sakallah, and Markov 2003)(Puget 2005a). It is worth mentioning that the SBDD and the SBDS methods can be used to break value symmetries as well. The GE-tree search is a method tailored for breaking value symmetries (Roney-Dougal et al. 2004). The search procedure is modified in order to never try values that are symmetrical to values tried before. Another avenue of research aims at transforming value symmetries into variable symmetries, through a reformulation of the problem(Flener et al. 2002)(Law and Lee 2003)(Law and Lee 2005)(Puget 2005c). These methods introduce additional variables as well as channeling constraints between these variables and the original variables of the CSP. Value symmetries on the original CSP induce variable symmetries on the additional variables. Then, variable symmetry breaking techniques can be used. This way, both variable and value symmetries can be broken, but not their combinations. Let us give a simple example of it.



We consider a simple graph coloring problem: how to color the vertices of a square with 4 colors? It can be modeled as a CSP with 4 variables:

$$v_0, v_1, v_2, v_3 \in \{0, 1, 2, 3\},$$

$$v_0 \neq v_1, v_1 \neq v_2, v_2 \neq v_3, v_3 \neq v_0$$

The values in this CSP are interchangeable, i.e. any permutation of values is a symmetry. Then, there are $4! = 24$ value symmetries. There are 8 variable symmetries, corresponding to the 8 symmetries of a square. Therefore, there are $192 = 24 \times 8$ symmetries in this CSP.

There are 84 solutions to this problem. When we break all variable symmetries and all value symmetries, only 4 solutions are left:

$$(v_0, v_1, v_2, v_3) = \begin{cases} (0,1,0,1) \\ (0,1,0,2) \\ (0,1,2,1) \\ (0,1,2,3) \end{cases} \quad (1)$$

However, the third solution is symmetrical to the second one. Indeed, it is:

$$v_0 = 0, v_1 = 1, v_2 = 2, v_3 = 1 \quad (2)$$

Let us apply the value symmetry that swaps 0 and 1. It gives:

$$v_0 = 1, v_1 = 0, v_2 = 2, v_3 = 0$$

Let us apply the variable symmetry that corresponds to a clockwise rotation. It gives:

$$v_1 = 0, v_2 = 1, v_3 = 0, v_0 = 2$$

It is precisely the second solution.

As far as we know, the only published symmetry breaking methods able to remove the third solution are SBDS and SBDD. The SSB method of (Sellmann and Van Hentenryck 2005) cannot be applied to this CSP because the variable symmetries are too complex for that method. The GE-tree method does not break any variable symmetry. The other methods break separately value and variable symmetries, but not all their combinations.

We present in this paper a set of symmetry breaking constraints that are able to remove all combinations of value and variable symmetries. In the above example, the addition of these constraints would leave only 3 solutions.

## Symmetries, Graphs and CSPs

We denote the set of integers ranging from 0 to $n-1$ by $I^n$.

A *constraint satisfaction problem* $\mathcal{P}$ (CSP) with $n$ variables is a triple $\mathcal{P} = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ where $\mathcal{V}$ is a finite set of variables $(v_i)_{i \in I^n}$, $\mathcal{D}$ a finite set of finite sets $(\mathcal{D}_i)_{i \in I^n}$, and every constraint in $\mathcal{C}$ is a subset of the cross product $\bigotimes_{i \in I^n} \mathcal{D}_i$. Without loss of generality, we can assume that $\mathcal{D}_i \subseteq I^k$ for some $k$.

An *assignment* is a member of $\mathcal{S}$, i.e. a vector of values $(a_i)_{i \in I^n}$ such that $a_i \in \mathcal{D}_i$ for all $i \in I^n$, and is denoted $(v_i = a_i)_{i \in I^n}$. A *partial assignment* is sub vector of an assignment.

A *solution* to $(\mathcal{V}, \mathcal{D}, \mathcal{C})$ is an assignment that is consistent with every member of $\mathcal{C}$.

The symmetries we consider are permutations, i.e. one to one mappings (bijections) from a finite set onto itself. Let $S^n$ be the set of all permutations of the set $I^n$. The image of $i$ by the permutation $\sigma$ is denoted $i^\sigma$. A permutation $\sigma \in S^n$ is fully described by the vector $[0^\sigma, 1^\sigma, \ldots, (n-1)^\sigma]$. The product of two permutations $\sigma$ and $\theta$ is defined by $i^{(\sigma\theta)} = (i^\sigma)^\theta$.

Given a permutation $\sigma$ of $I^n$, we define a variable permutation on (partial) assignments as follows:

$$((v_i = a_i)_{i \in I^n})^\sigma = ((v_i = a_{i^\sigma})_{i \in I^n})$$

Such permutation is called a *variable symmetry* if it maps solutions to solutions.

Given a permutation $\theta$ of $I^k$, we define a value permutation on (partial) assignments as follow:

$$((v_i = a_i)_{i \in I^n})^\theta = ((v_i = (a_i)^\theta)_{i \in I^n})$$

Such permutation is called a *value symmetry* if it maps solutions to solutions.

## Lex Leader Solutions

A very powerful symmetry breaking method has been proposed in (Crawford et al. 1996). The idea is to use a lexicographic order to compare solutions. Given two finite sequences $X = (x_0, x_1, \ldots, x_{n-1})$ and $Y = (y_0, y_1, \ldots, y_{n-1})$, we say that $X$ is *lex smaller* than $Y$ (denoted $X \preceq Y$) if and only if :

$$\forall k \in I^n,\ (x_0 = y_0 \wedge \ldots \wedge x_{k-1} = y_{k-1}) \rightarrow x_k \leq y_k \quad (3)$$

Let us consider a solution $(v_i = a_i)_{i \in I^n}$ of the CSP. Let us consider the set of all solutions that are symmetric to it. These solutions are $(v_i = a_i)^\sigma_{i \in I^n}$ where $\sigma$ ranges over the group of symmetries of the CSP. Among all these solutions there is one that is lexicographically smaller than the others. This solution $S$ satisfies the constraint:

$$\forall \sigma \in G,\ S \preceq S^\sigma \quad (4)$$

The above has been widely used for variable symmetries. If $\sigma$ is a permutation of variables that defines a symmetry, then (4) is equivalent to the following constraint:

$$\forall \sigma \in G,\ (v_0, v_1, \ldots v_{n-1}) \preceq (v_{0^\sigma}, v_{1^\sigma}, \ldots, v_{n-1^\sigma}) \quad (5)$$

This constraint can be easily enforced using a global constraint (Frisch et al. 2002)(Carlson and Beldiceanu 2002). This global constraint has been used in various context, such as in (Flener et al. 2002).

## Lex leader Constraint for Value Symmetries

We want to enforce the constraint (4) when there are value symmetries. In order to do so, we will use another global constraint, the *element* constraint. This constraint is implemented in all major constraint programming systems.

An element constraint has the following form:

$$y = A[x]$$

where $A = [a_0, a_1, \ldots, a_{k-1}]$ is an array of integers, $x$ and $y$ are variables. The above element constraint is equivalent to:

$$y = a_x \wedge x \in I^k$$

i.e. it says that $y$ is the $x$-th element of the array $A$. We will only consider injective element constraints, where the values appearing in the array $A$ are pair wise distinct. In this case, the operational semantics of the element constraint is defined by the logical equivalence:

$$\forall i \in I^k,\ x = i \leftrightarrow y = a_i$$

For the sake of clarity, we extend the element constraint to sequences of variables. If $X = (v_i)_{i \in I^n}$ is a finite sequence of variables, then we define $A[X]$ as the application of an element constraint to each element of the sequence:

$$A[X] = (A[v_i])_{i \in I^n}$$

Element constraints can be used to describe applications of finite functions. For instance,

$$y = 3^x \wedge x \in I^4$$

is equivalently expressed through the following element constraint:

$$y = A[x] \wedge A = [1, 3, 9, 27]$$

The following observation is the basis for our new method for breaking value symmetries: element constraint can also be used to represent the effect of value symmetries.

Indeed, let $\theta$ be a value permutation corresponding to a value symmetry. By definition, any assignment of a value $a$ to a variable $x$ is transformed into the assignment of $a^\theta$ to $x$:

$$(x = a)^\theta = (x = a^\theta)$$

Let us consider $x^\theta$. The permutation $\theta$ is represented by the array $A_\theta = [0^\theta, 1^\theta, \ldots, (k-1)^\theta]$. It defines a finite function that maps $a$ to $a^\theta$. The application of this function to $x$ can be expressed by $A_\theta[x]$. Therefore, $x^\theta = A_\theta[x]$. We have represented the effect of the value symmetry by an element constraint.

More generally, if $(a_0, a_1, \ldots, a_{n-1})$ is the sequence of values taken by the variables $\mathcal{V} = (v_0, v_1, \ldots, v_{n-1})$, then $(a_0^\theta, a_1^\theta, \ldots, a_{n-1}^\theta)$ is the sequence of values taken by the variables $A_\theta[\mathcal{V}]$. Therefore, $S^\theta = A_\theta[S]$ for all solutions $S$.

Let $S$ be a lex leader solution. Then, $S \preceq S^\theta$. Using the above, it means that $S$ is a solution of the CSP with the additional constraint:

$$\mathcal{V} \preceq A_\theta[\mathcal{V}]$$

This constraint is the conjunction of $n$ element constraints and one lex constraint.

Let us go back to the example given in the introduction. There is one value symmetry $(01)$ that swaps 1 and 0. It defines a finite function represented by the array $\mathrm{A}_{(01)} = [1, 0, 2, 3]$. The following constraint removes the value symmetry:

$$\mathcal{V} \preceq \mathrm{A}_{(01)}[\mathcal{V}]$$

More generally, let us consider now the case where any symmetry is the composition $\sigma\theta$ of a variable permutation $\sigma$ and a value permutation $\theta$. The variable permutation $\sigma$ is defined by a permutation of $I^n$. The value permutation is defined by a permutation of $I^k$.

If $(a_0, a_1, \ldots, a_{n-1})$ is the sequence of values taken by the variables $X = (v_0, v_1, \ldots, v_{n-1})$, then $(a_{0^\sigma}^\theta, a_{1^\sigma}^\theta, \ldots, a_{(n-1)^\sigma}^\theta)$ is the sequence of values taken by the variables $A_\theta[X^\sigma]$. Therefore, $S^{\sigma\theta} = A_\theta[S^\sigma]$ for all solutions $S$.

Let $S$ be a lex leader solution. Then, $S \preceq S^{\sigma\theta}$. Therefore, $S$ is a solution of the CSP with the additional constraint:

$$\mathcal{V} \preceq A_\theta[\mathcal{V}^\sigma] \tag{6}$$

This constraint is the conjunction of a permutation of the variables, a lex constraint, and a global element constraint.

We can state these constraints for all symmetries.

In our graph coloring example, it gives 192 constraints. Let us look at the symmetry made by a clockwise rotation and a swap of values 0 and 1. The lex constraint for this symmetry is:

$$(v_0, v_1, v_2, v_3) \preceq \mathrm{A}_{(01)}[(v_3, v_0, v_1, v_2)] \tag{7}$$

where $\mathrm{A}_{(01)} = [1, 0, 2, 3]$

Let us check that the solution (2) is not consistent with this constraint. We replace the variables by their values in the constraint. It gives:

$$(0, 1, 2, 1) \preceq \mathrm{A}_{(01)}[(1, 0, 1, 2)]$$

By definition of the element constraint on arrays, it is equivalent to:

$$(0, 1, 2, 1) \preceq (\mathrm{A}_{(01)}[1], \mathrm{A}_{(01)}[0], \mathrm{A}_{(01)}[1], \mathrm{A}_{(01)}[2])$$

By definition of $\mathrm{A}_{(01)}$ it gives:

$$(0, 1, 2, 1) \preceq (0, 1, 0, 2)$$

It is false. Therefore, the solution (2) is pruned by the constraint (7), and by extension, by the constraints (6).

This method requires $M \times N$ constraints when there are $M$ variables symmetries and $N$ value symmetries. This method will not scale well with the number of value symmetries. The next section describes a way to cope with a large number of value symmetries.

## A Global Constraint for Value Symmetries

We present in this section a global constraint that can handle a large number of value symmetries. We assume that all symmetries can be written as the composition $\sigma\theta$ of a variable symmetry $\sigma$, and a value symmetry $\theta$. Let us consider all the constraints (6) for a given $\sigma$:

$$\forall\theta, \mathcal{V} \preceq A_\theta[\mathcal{V}^\sigma] \tag{8}$$

We have seen in the previous section how to enforce this constraint with one constraint per value symmetry $\theta$. We now want to enforce using a single global constraint. In order to define such global constraint, let us first formalize tree search.

We consider complete tree search methods. A variable is selected at each non leaf node. Then, one branch is created for every value in the domain of that variable. We identify a node with the variable assignments that are true at the node. Variables are listed in the order in which they have been assigned during search. Constraints can prune the tree: some nodes are inconsistent. These nodes have no children. Solutions are leaves of the search tree that are not inconsistent. Some constraint propagation algorithm may be applied at every node. It may result in some assignment of variables. In such case, we introduce a sequence of child node, one for each assignment. Therefore, all the variables appear in the path from the root node to a solution.

For instance, the tree search for the graph coloring example given in the introduction with the symmetry breaking constraints described in the previous section is depicted in Fig. 1. In the node $v_0 = 0$, constraint propagation results in $v_1 = 1$. A node is created to represent this partial assignment.

Using (3), the constraint (8) is equivalent to the conjunction of the constraints for all $k \in I^n$ and for all $\theta$:

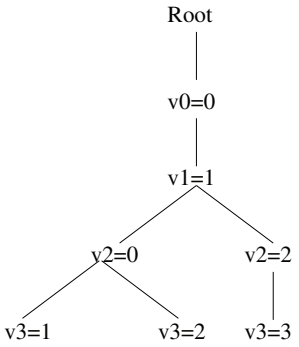


Figure 1: A tree search

$$(v_0 = A_\theta[v_{0^\sigma}] \wedge \ldots \wedge v_{k-1} = A_\theta[v_{(k-1)^\sigma}]) \rightarrow v_k \leq A_\theta[v_{k^\sigma}]$$

We want to develop a forward checking algorithm for these constraints. Assume that we reach a state $\Sigma$ where the first $j$ variables have been instantiated with values $(a_0, a_1, \ldots, a_{j-1})$. Let $K$ be the smallest $i$ such that $v_i$ or $v_{i^\sigma}$ is not instantiated in $\Sigma$. Then, we can use the above constraint with any $k$ such that $k \leq K$. We replace the variables by their values, which gives:

$$(a_0 = A_\theta[a_{0^\sigma}] \wedge \ldots \wedge a_{k-1} = A_\theta[a_{(k-1)^\sigma}]) \rightarrow v_k \leq A_\theta[v_{k^\sigma}]$$

where $a_{i^\sigma}$ is the value assigned to the variable $v_{i^\sigma}$. If the left hand side is not true, then, nothing can be done. If the left hand side is true, then, $\theta$ is such that:

$$\forall i \in I^k,\ a_i = A_\theta[(a_{i^\sigma})] \tag{9}$$

Let $G^\sigma_\Sigma$ be the set of value symmetries that satisfy (9). Then, for any of those $\theta$, we have to enforce the right hand side:

$$\forall \theta \in G^\sigma_\Sigma, v_k \leq A_\theta[v_{k^\sigma}] \tag{10}$$

It is simple to enforce. Let $a_k$ be the minimum value in the domain of $v_k$. Let $b$ be a value in the domain of $v_{k^\sigma}$ in state $\Sigma$. If there exists $\theta \in G^\sigma_\Sigma$ such that $a_k > A_\theta[b]$, then $b$ should be removed from the domain of $v_{k^\sigma}$.

Therefore, in order to enforce (10), it is necessary to remove all the values $b$ from the domain of $v_{k^\sigma}$ such that $a_k > A_\theta[b]$:

$$\forall b\ \exists \theta \in G^\sigma_\Sigma,\ a_k > A_\theta[b] \rightarrow v_{k^\sigma} \neq b \tag{11}$$

It is not difficult to see that it is a sufficient condition as well. Indeed, if it is enforced at each node, then, any solution is a lex leader one. The proof is a mere application of the definitions.

Let us see how it works in our graph coloring example. We consider the variable symmetry $\sigma$ defined by the permutation $[1, 2, 3, 0]$. Let us look at the node $\Sigma = (v_0 = 0, v_1 = 1, v_2 = 2)$. The smallest $k$ such that either $v_k$ or $v_{k^\sigma}$ is not instantiated is 2. Indeed, $v_{2^\sigma}$ is $v_3$ which is not instantiated. The set $G^\sigma_\Sigma$ is the set of all value symmetries $\theta$ such that:

$$a_0 = A_\theta[a_{0^\sigma}] \wedge a_1 = A_\theta[a_{1^\sigma}]$$

By definition of $\sigma$, it gives:

$$a_0 = A_\theta[a_1] \wedge a_1 = A_\theta[a_2]$$

Since $a_0 = 0$, $a_1 = 1$, and $a_2 = 2$, it is equivalent to:

$$0 = A_\theta[1] \wedge 1 = A_\theta[2] \tag{12}$$

Then, for each $\theta$ that satisfies (12), we must enforce:

$$v_k \leq A_\theta[v_{k^\sigma}]$$

i.e. we must enforce:

$$v_2 \leq A_\theta[v_3]$$

It is equivalent to:

$$2 \leq A_\theta[v_3]$$

Formula (11) becomes:

$$\forall b\ \exists \theta,\ 0 = A_\theta[1] \wedge 1 = A_\theta[2] \wedge 2 > A_\theta[b] \rightarrow v_3 \neq b$$

The left had side conditions are true for $b = 1$ and for $b = 2$. We can then remove both 1 and 2 from the domain of $v_3$. It prunes the solution (2) that was discussed in the introduction.

In order to implement our method, one need to efficiently compute the value symmetries that satisfy (9). It can be done using computational group theory algorithms (see (Seress 2003) for instance). We have implemented a special case when any value permutation is a value symmetry. In this case, it is easy to compute $G^\sigma_\Sigma$ from (9). Indeed, (9) is of the form:

$$A_\theta[b_0] = c_0, \ldots, A_\theta[b_{k-1}] = c_{k-1} \tag{13}$$

Let $\mathcal{C}$ be the set of the $c_i$ that appear in (13), and let $\mathcal{B}$ be the set of the $b_i$ that appear in (13). Then, the set of value symmetries $\theta$ that are consistent with (13) are:

$$\begin{array}{rl} \forall i \in I^k, & A_\theta[b_i] = c_i \\ \forall b \in I^n - \mathcal{B}, & A_\theta[b] \in I^n - \mathcal{C} \end{array} \tag{14}$$

Then, (11) becomes:

$$\begin{array}{rlcl} \forall i \in I^k, & a_k > c_i & \rightarrow & v_{k^\sigma} \neq b_i \\ \forall b \in I^n - \mathcal{B}, & a_k > min(I^n - \mathcal{C}) & \rightarrow & v_{k^\sigma} \neq b \end{array} \tag{15}$$

It is straightforward to implement.

It is worth looking at the case where $\sigma$ is the identity. In this case, the above reasoning can be simplified. First of all, $G^{id}_\Sigma$ is now the set of value symmetries $\theta$ such that:

$$\forall i \in I^k,\ a_i = A_\theta[a_i]$$

It is called the point wise stabilizer of $(a_0, a_1, \ldots, a_{k-1})$. This set is denoted $G_{(a_0, a_1, \ldots, a_{k-1})}$. Then, condition (11) becomes simpler. We only have to remove from the domain of $v_k$ all the values $b$ such that there exists $\theta$ in $G_{(a_0, a_1, \ldots, a_{k-1})}$ such that $b > b^\theta$. It is exactly the definition of the GE-tree method of (Roney-Dougal et al. 2004).

## Experimental Results

We have implemented both the constraints (6), and the global constraint that prunes values satisfying (11). Both were implemented using ILOG Solver 6.2(ILOG SA. 2006). All symmetries were computed with the method of (Puget 2005b). We considered several well known examples: graceful graphs, $n \times n$ queen problem, and graph coloring. Running times are measured on a Dell D800 laptop with a 1.4MHz Pentium M processor, running Windows XP.

Graceful graphs were studied in (Petrie and Smith 2003), with updated results in (Petrie 2005). We have compared the constraints of (6) (LEX) to the one of (Petrie 2005) and to our previous work in (Puget 2005c). Table 1 gives the number of solutions, the running time and the number of backtrack for each of the three methods. The last example in Table 1 is a simplified version of the K4×K3 example where one edge value is set. Without this extra setting, the problem was too difficult for the method of (Petrie 2005). (Petrie 2005) uses ECLIPSE on a computer slightly faster than our. Our method is much faster. We believe that a large part of the difference comes from the difference in symmetry breaking methods. Our approach is also better than the one of (Puget 2005c). Indeed, this method does not break all symmetries, as witnessed by the number of solutions found.

| | GE-tree | | (Puget 2005c) | | | LEX | | |
|---|---|---|---|---|---|---|---|---|
| n | SOL | sec. | SOL | BT | sec. | SOL | BT | sec. |
| 5 | 1 | 0.68 | 2 | 0 | 0 | 1 | 1 | 0 |
| 6 | 0 | 0.96 | 0 | 5 | 0 | 0 | 5 | 0 |
| 7 | 1 | 8.36 | 4 | 271 | 0.11 | 1 | 1 | 0.05 |
| 8 | 0 | 927.36 | 0 | 23,794 | 4.27 | 0 | 12,349 | 2.28 |

Table 2. Results for finding all solutions to the $n \times n$ queen problem.

Let us look at another difficult problem, namely the $n \times n$ queen problem taken from (Kelsey, Linton, and Roney-Dougal 2004). The problem is to color a $n \times n$ chessboard with $n$ colors, such that no line (row, column or diagonal) contains the same color twice. This problem can be modeled with $n^2$ variables, one per square of the chess board, and one all different constraint per line. Any permutation of the values is a symmetry. There are also 8 variable symmetries corresponding to the 8 symmetries of a square. We compare our method with the GE-tree method(Kelsey, Linton, and Roney-Dougal 2004), and our method of (Puget 2005c). The GE-tree method could not be used alone, since there are also some variable symmetries. Then, the authors of (Kelsey, Linton, and Roney-Dougal 2004) have combined GE-tree with the SBDD method. Results are shown in Table 2. First fail principle is used: the variable with the smallest domain size is selected during search. It is worth noting that the computer used in (Kelsey, Linton, and Roney-Dougal 2004) (2.4 GHz) is faster than ours (1.4 GHz)

As a last example, let us look at the dodecahedron coloring problem taken from(Gent et al. 2003). The problem is to color the vertices of the dodecahedron with $m$ colors so that no edge has the same color at both ends. It is a standard graph coloring example. We have compared our approach with the GAP-SBDD method (Gent et al. 2003). It would be interesting to compare our method to the methods of (Benhamou 2004) or (Ramani et al. 2004). Unfortunately, no report on the use of these methods for this graph have been published.

The variable symmetries are equivalent to the symmetries of the dodecahedron. There are 120 of them. Curiously, the authors of (Gent et al. 2003) overlooked half of the symmetries. These are obtained using the inversion through the center of the dodecahedron. It makes the comparison somewhat awkward. Anyway, we provide some results in Table 3 for varying numbers of colors. It is worth noting that the computer used in (Gent et al. 2003) is slightly slower (1 GHz) than ours (1.4 GHz).

| | GAP-SBDD | | | us | | |
|---|---|---|---|---|---|---|
| Colors | Sol | BT | sec. | Sol | BT | sec. |
| 3 | 31 | 50 | 0.51 | 17 | 22 | 0 |
| 4 | 117,902 | 109,502 | 879 | 59, 027 | 33,583 | 2.04 |
| 5 | | | | 7,826,402 | 3,218,147 | 184 |
| 6 | | | | 174,936,085 | 57,671,880 | 3583 |

Table 3. Results for finding all colorings of the Dodecahedron.

Our approach is much more scalable and efficient. Of course, the difference of system (Eclipse vs. ILOG SOLVER), and the fact that we take into account twice as many symmetries as the others explains part of the difference. However, additional data provided in (Gent et al. 2003) show that 770 seconds were spent in the symmetry handling code written in the highly efficient GAP system, and only 109 seconds in Eclipse. The total running time for our method is 5 time smaller than the time spent in handling symmetry in the other method.

## Conclusions

We have presented a new way of breaking value symmetries in presence of variable symmetries. We have first shown that any symmetry made out of a value symmetry and a variable symmetry could be broken by a combination of element constraints and lex constraints. It is the first time to our knowledge that it is possible to break all combinations of value and variable symmetries by adding constraints. This method is quite efficient when the number of symmetries is not too large, as shown by experiments using graceful graph problems. We have also derived a new global constraint that deals with the case where there are too many value symmetries. We have shown how to propagate this global constraint efficiently. We have also shown that our method can be related to the GE-tree method when there are no variable symmetries. Our experimental results prove that our approach is significantly faster than any previously published method.

Our method requires to state one global constraint per variable symmetry, regardless of the number of value symmetries. It remains to be seen if we can state less constraints than the number of variable symmetries. It would be interesting to see if we can combine our work for instance with the one of (Puget 2005a).

We have implemented our global constraint only for the case where the value symmetry group is the group of all permutations. It would be interesting to implement it for general groups of symmetries. Such an implementation would be similar to the GE-tree method. Indeed, one merely needs to replace the use of stabilizers in the GE-tree method by the sets of value symmetries that satisfy (11). In the meantime, we can use the combination of lex and element constraints to break any combination of value and variable symmetries.

| | (Petrie 2005) | | | (Puget 2005a) | | | LEX | | |
|---|---|---|---|---|---|---|---|---|---|
| Graph | Sol | BT | sec. | Sol | BT | sec. | Sol | BT | sec. |
| K3×P2 | 4 | 6 | 0.25 | 8 | 83 | 0.01 | 4 | 47 | 0.01 |
| K4×P2 | 15 | 147 | 12.9 | 30 | 1,863 | 0.27 | 15 | 936 | 0.18 |
| K5×P2 | 1 | 4,172 | 1356 | 2 | 53,266 | 6.5 | 1 | 12,371 | 4.7 |
| K6×P2 | | | | 0 | 1,326,585 | 305 | 0 | 575,609 | 318 |
| DW3 | 0 | 48 | 1.95 | | | | 0 | 0 | 0 |
| DW4 | 44 | 1,053 | 36.1 | | | | 44 | 4,053 | 0.57 |
| DW5 | 1,216 | 33,622 | 1,609 | | | | 1,216 | 133,517 | 16.75 |
| DW6 | | | | | | | 35,877 | 6,912,716 | 1,023 |
| K3×K3 | 0 | 1393 | 68 | | | | 0 | 5,574 | 0.76 |
| K4×K3 | | | | | | | 22 | 3,521,832 | 696 |
| K4×K3(*) | 17 | | 38,000 | | | | 17 | 1,450,719 | 293 |

Table 1. Result for finding all graceful colorings.


## Acknowledgements

The author would like to thank Marie Puget for her support and her thorough proof reading.


## References


Aloul, F.A.; Sakallah, K.A., and Markov, I.L. 2003. "Efficient Symmetry Breaking for Boolean Satisfiability." In *Proceedings of IJCAI 2003*, pages 271-276.

Benhamou, B., 2004. "Symmetry in Not-equals Binary Constraint Networks". In *Proceedings of SymCon'04*, pp. 2-8, Toronto, September 2004

Carlsson, M., and Beldiceanu, N. 2002. "Revisiting the Lexicographic Ordering Constraint" SICS Technical report T2002:17

Crawford, J.; Ginsberg, M.; and Luks E.M., Roy, A. 1996. "Symmetry Breaking Predicates for Search Problems". In *Proceedings of KR'96*, pp. 148-159.

Fahle, T.; Shamberger, S.; and Sellmann, M. 2001. "Symmetry Breaking." In Proceedings of CP01 (2001) 93-107.

Flener, P.;Frisch, A. M.; Hnich, B.; Kiziltan, Z.; Miguel, I.; Pearson, J.; and Walsh, T. 2002. "Breaking Row and Column Symmetries in Matrix Models". In *Proceedings of CP'02*, pp. 462-476, 2002

Focacci, F., and Milano, M. 2001. "Global Cut Framework for Removing Symmetries." In Proceedings of CP'01 (2001) 75-92.

Frisch, A. M.; Hnich, B.; Kiziltan, Z.; Miguel, I.; and Walsh, T. 2002. Global Constraints for Lexicographic Orderings. In Proceedings of CP'02, pp 93-108.

Gent, I.P.; and Harvey, W.; and Kelsey, T. 2002. "Groups and Constraints: Symmetry Breaking During Search". In *Proceedings of CP 2002*, pp. 415-430.

Gent, I.P.; Harvey, W.; Kelsey, T.; and Linton, S. 2003. "Generic SBDD Using Computational Group Theory". In *Proceedings of CP 2003*, pp. 333-437

ILOG SA. 2006. ILOG Solver 6.2. User Manual. ILOG, S.A., Gentilly, France, January 2006.

Kelsey, T; Linton, SA.; and Roney-Dougal, CM 2004. "New Developments in Symmetry Breaking in Search Using Computational Group Theory". In *Proceedings AISC 2004*. Springer LNAI. 2004.

Law, Y. C., and Lee, J. H. M. 2003. "Expressing Symmetry Breaking Constraints Using Multiple Viewpoints and Channeling Constraints". In *Proceedings of SymCon 03* (held in conjunction with CP-2003), pages 127-141, 2003.

Law, Y. C., and Lee, J. H. M. 2005. "Breaking Value Symmetries in Matrix Models using Channeling Constraints". In *Proceedings of the 20th Annual ACM Symposium on Applied Computing (SAC-2005)*, pages 375-380, 2005.

Petrie, K., Smith, B.M. 2003. "Symmetry breaking in graceful graphs." In Proceedings of CP'03, LNCS 2833, 930-934, Springer Verlag, 2003.

Petrie, K., Smith, B.M. 2005. "Comparison of Symmetry Breaking Methods in Constraint Programming" In *Proceedings of SymCon05*, the 5th International Workshop on Symmetry in Constraints, 2005

Puget, J.-F. 2005a. "Breaking symmetries in all different problems". In *Proceedings of IJCAI 05*, pages 272-277, 2005.

Puget, J.-F. 2005b. "Automatic detection of variable and value symmetries" In *Proceedings of CP 05*, pages 475-489, 2005.

Puget J.-F. 2005c. "Breaking All Value Symmetries in Surjection Problems" In *Proceedings of CP 05*, pages 490-504, 2005.

Ramani, A.; Aloul, F.A.; Markov, I.L.; Sakallah, K.A. 2004. "Breaking Instance-Independent Symmetries in Exact Graph Coloring." In *Proceedings of DATE 2004*, pages 324-331.

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

Sellmann, M., and Van Hentenryck, P. 2005. Structural Symmetry Breaking In *Proceedings of IJCAI 05*.

Seress, A. 2003. *Permutation Group Algorithms* Cambrige University Press, 2003.