Breaking Row and Column Symmetry in Matrix Models

Pierre Flener[1], Alan M. Frisch[2], Brahim Hnich[1],
Zeynep Kızıltan[1], Ian Miguel[2], Justin Pearson[2], and Toby Walsh[3]
[1] *Uppsala University, Uppsala, Sweden*
[2] *The University of York, York, England*
[3] *Cork Constraint Computation Centre, Cork, Ireland*

Abstract. We identify an important class of symmetries in constraint programming: matrices of decision variables with rows and columns that are symmetric. We show how we can lexicographically order both rows and columns to break such symmetry. Whilst lexicographically ordering rows breaks all row symmetry (and lexicographically ordering columns breaks all column symmetry), lexicographically ordering both rows and columns fails to break all row and column symmetry. Nevertheless, our experimental results show that it is effective at dealing with row and column symmetry. We extend these results to cope with symmetries in 3 or more dimensions, partial symmetries, and symmetric values. Finally, we identify a number of special cases where all row and column symmetry can be eliminated by the addition of some simple symmetry breaking constraints.

1 Introduction

Modelling is one of the most difficult parts of constraint programming. Indeed, Freuder has identified it as the “last frontier” [5]. One source of difficulty is dealing with symmetry efficiently and effectively. Symmetry occurs in many assignment, scheduling, configuration, and design problems. Identical machines in a factory, repeat orders, equivalent time periods and equally skilled workers are just a few of the items likely to introduce symmetry into a constraint program. If we ignore symmetry, the constraint program will waste time considering symmetric but essentially equivalent assignments. As there are often an exponential number of symmetries, this can be very costly. To help tackle this problem, we identify an important class of symmetries that occurs frequently in constraint programs. We show how simple constraints can be added to the model to break such symmetries, and analyse the effectiveness of these methods theoretically and experimentally on balanced incomplete block design (BIBD) generation.

2 Matrix models

A *matrix model* is a constraint program that contains (one or more) matrices of decision variables. For example, a natural model of the round robin tournament scheduling problem (prob026 at www.csplib.org) is a 2-dimensional (2-d) matrix of decision variables, each of which is assigned a value corresponding to the match played in a given week and period [7].

In this case, the matrix is obvious in the solution to the problem: we need a *table* of fixtures. However, many other problems that are less obviously defined in terms of matrices of values can be efficiently represented and effectively solved using matrix models [4]. For example, the rack configuration problem can be modelled with a 2-d 0/1 matrix representing which cards go into which racks [8].

Symmetry is an important aspect of matrix models. Symmetry often occurs because objects within the model are indistinguishable. For example, in the rack configuration problem, racks of the same type are indistinguishable. We can therefore permute any two racks of the same type. That is, we can permute any two columns of the associated matrix. Similarly, in the tournament scheduling problem, weeks and periods are indistinguishable. We can therefore permute any two weeks or any two periods in the schedule. That is, we can permute any two rows or any two columns of the associated matrix. We define a *symmetry* of a matrix model as a bijection on the decision variables in the matrix that preserves solutions. We say that two variables are *indistinguishable* if they occur in the same cycle of one of these symmetry bijections.

Two common types of symmetry in matrix models are row symmetry and column symmetry. The two examples in the last paragraph are row and column symmetries. We define a *column symmetry* of a 2-d matrix model as a bijection on non-identical columns that preserves solutions, and a *row symmetry* as a bijection on non-identical rows that again preserves solutions. We say that two columns (rows) are *indistinguishable* if they occur in the same cycle of one of these symmetry bijections. Note that we ignore bijections between identical rows or columns, as they do not increase the number of symmetric solutions. Row and column symmetry are both trivially symmetries of a matrix model. Note that the reverse does not hold in general (for example, the rotational symmetry of a matrix model is neither a row nor a column symmetry). We say that a matrix model has *row symmetry* (*column symmetry*) iff all non-identical rows (columns) are indistinguishable. We say that a matrix model has *partial row symmetry* (*partial column symmetry*) iff a strict subset of the non-identical rows (columns) are indistinguishable. These definitions can easily be extended to matrix models with more or less than 2 dimensions.

Many examples of row and column symmetry have been observed in a wide variety of constraint programs [4]. For instance, row and column symmetries occur in matrix models of the BIBD problem (prob028 at www.csplib.org), the steel mill slab design problem [4], the social golfers problems (prob010 at www.csplib.org), the template design problem (prob002 at www.csplib.org), the progressive party problem (prob013 at www.csplib.org), and (as argued above) the rack configuration and round robin tournament scheduling problems.

3 Breaking symmetry

There are a number of ways of dealing with symmetry in constraint programming (see Section 7 for a longer discussion). A popular and simple approach is to add constraints which break the symmetry [2].

3.1 Row or column symmetry

One common method to break symmetry is to order the symmetric objects. To break row or column symmetry, we can simply order the rows or columns lexicographically. We say that the rows in a matrix are *lexicographically ordered* if each row is lexicographically bigger than the previous, and *anti-lexicographically ordered* if each row is lexicographically smaller than the previous. Similarly, we say that the columns in a matrix are lexicographically ordered if each column is lexicographically larger than the previous, and anti-lexicographically ordered if each column is lexicographically smaller than the previous. As a lexicographical ordering is total, adding a lexicographical (or anti-lexicographical) ordering constraint to the rows will break all row symmetry. Dually, adding a lexicographical (or anti-lexicographical) ordering constraint to the columns will break all column symmetry.

3.2 Row and column symmetry

Whilst breaking row or column symmetry in a matrix model is easy, breaking *both* row and column symmetry is difficult since the rows and columns intersect. Lexicographically ordering the rows will tend to put the columns into lexicographic order. However, it does not always order the columns lexicographically, and lexicographically ordering the columns can then disrupt the lexicographic ordering on the rows.

Example 1. *Consider a 3 by 4 matrix of 0/1 variables, x_{ij}, with the constraints that $\sum_{ij} x_{ij} = 7$, and $\sum_i x_{ij}.x_{ik} \leq 1$ for $j \neq k$ (i.e. the dot product of any two rows is 1 or less). This model has both row and column symmetry. A solution with lexicographically ordered rows, but not lexicographically ordered columns, is:*

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

A solution with lexicographically ordered columns, but not lexicographically ordered rows is:

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

However, by reordering the last two rows, we get a solution that is lexicographically ordered along both rows and columns:

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

It is not difficult to construct examples that need several rounds of ordering rows then columns. Despite such conflicts, it is always possible to get the rows and columns into lexicographic order.

Theorem 1. *If a 2-d matrix model with row and column symmetry has a solution, then it has a solution with both the rows and columns lexicographically ordered.*

Proof: To show that there is a matrix with lexicographically ordered rows and columns, we give an ordering on matrices (denoted $\leq_{mat}$) that strictly decreases each time we lexicographically order a pair of rows or columns. As this matrix ordering is finite and bounded below, ordering rows and columns must terminate, giving an element with rows and columns lexicographically ordered. To compare two matrices, we simply apply the lexicographic ordering to the sequence formed by appending their rows together in order. Ordering a pair of rows replaces a larger row at the front of this sequence by a smaller row from further down. Hence, ordering a pair of rows moves us down the matrix ordering. Ordering columns also moves us down this matrix ordering. The columns may have a number of values in common at the top. Swapping these columns does not affect the matrix ordering when just considering the corresponding top rows. However, there is then one value in the left column that is replaced by a smaller value in the right column. This moves us down the matrix ordering. The matrix ordering is also finite, as there are only a finite number of permutations of the values in the matrix, and bounded below, namely by a matrix with rows and columns lexicographically ordered. □

The last result shows that we can lexicographically order both rows and columns. Dually, we can anti-lexicographically order both rows and columns. However, we cannot always lexicographically order the rows and anti-lexicographically order the columns. Lexicographically ordering the rows will tend to push the largest values to the bottom left of the matrix. Anti-lexicographically ordering the columns will tend to push the larger values to the top right. For this reason, the two orders can conflict.

Example 2. *Consider a 2 by 2 matrix of 0/1 variables, x_{ij}, with the constraints that $\sum_i x_{ij} = 1$ and $\sum_j x_{ij} = 1$ (i.e., every row and column has a single 1 in it). This matrix model has both row and column symmetry, and has two symmetric solutions:*

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

The first solution has rows and columns that are lexicographically ordered, whilst the second has rows and columns that are anti-lexicographically ordered. There is no solution in which rows are lexicographically ordered and columns are anti-lexicographically ordered.

Lexicographically ordering the rows (resp. columns) breaks all row (resp. column) symmetry. We were therefore very surprised to discover that lexicographically ordering the rows and columns does not break all row and column symmetry.

Example 3. *Consider a 3 by 3 matrix of 0/1 variables, x_{ij}, with the constraints that $x_{i1} + x_{i2} + x_{i3} \geq 1$ and $\sum_{ij} x_{ij} = 4$. This model has both row and column symmetry. There are three symmetric solutions that have lexicographically ordered rows and columns:*

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \quad \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix}$$

Whilst lexicographically ordering the rows and columns in a matrix model does not break all row and column symmetry, our experimental result (see Section 6) suggest that it breaks enough symmetry to be useful practically.

4 Extensions

We consider a number of extensions which extend the utility of these results considerably.

4.1 Higher dimensions

Many problems can be efficiently and effectively modelled using matrix models with more than two dimensions. For example, the social golfers problem can be modelled with a 3-d matrix whose dimensions correspond to weeks, groups, and players [12]. A variable x_{ijk} in this matrix is true iff in week i player j plays in group k. This matrix model is symmetric along each of its three dimensions: weeks are indistinguishable, as are groups and players. We can generalize lexicographical ordering constraints to three or more dimensions to break such symmetry.

Consider a 2-d matrix model. If we look along a particular dimension, we see 1-d vectors at right angles to this axis. To break symmetry, we simply order these vectors lexicographically. Now consider a 3-d matrix model. If we look along a particular dimension, we see 2-d "slices" of the matrix which are orthogonal to this axis. To break symmetry, we simply need to order these slices. A simple way is to flatten each slice onto a vector, and lexicographically order them. And in d dimensions, we see slices which are $d-1$ dimensional hypercubes, which can be compared by flattening them onto vectors and lexicographically ordering them.

Definition 1 (multi-dimensional lexicographical ordering).
For 2-dimensional matrices:
$\text{lex}^2(X[][])$ *iff* $\forall i, j$ $\text{lex}(X[i][], X[i+1][])$ *and* $\text{lex}(X[][j], X[][j+1])$.
For 3-dimensional matrices:
$\text{lex}^3(X[][][])$ *iff* $\forall i, j, k$ $\text{lex}(\text{flat}(X[i][][]), \text{flat}(X[i+1][][]))$,
$\text{lex}(\text{flat}(X[][j][]), \text{flat}(X[][j+1][]))$ *and* $\text{lex}(\text{flat}(X[][][k]), \text{flat}(X[][][k+1]))$.
For a d-dimensional matrix:
$\text{lex}^d(X[][]\ldots[])$ *iff* $\forall i, j, \ldots, l$ $\text{lex}(\text{flat}(X[i][]\ldots[]), \text{flat}(X[i+1][]\ldots[]))$,
$\text{lex}(\text{flat}(X[][j]\ldots[]), \text{flat}(X[][j+1]\ldots[]))$, $\ldots$, $\text{lex}(\text{flat}(X[][]\ldots[l]), \text{flat}(X[][]\ldots[l+1]))$.
Where $\text{flat}(X[]) = X[]$ *and* $\text{flat}(X[][]\ldots[]) = \text{app}(\text{flat}(X[1][]\ldots[]), \ldots \text{flat}(X[n][]\ldots[]))$

As in the 2-dimensional case, we can show that this multi-dimensional lexicographical ordering breaks some of the symmetry. Unfortunately, it does not break all symmetry as the 2-dimensional counter-examples easily generalize to three or more dimensions.

Theorem 2. *If a d-dimensional matrix model ($d \geq 1$) with symmetry along each dimension has a solution, then it has a solution under the multi-dimensional lexicographical ordering.*

Proof: We give a proof for 3 dimensions. The proof generalizes to higher dimensions in a straightforward way. As in the 2-dimensional case, we give an ordering on 3-dimensional matrices that strictly decreases each time we re-order a pair of slices. This ordering is simply that formed by flattening the 3-dimensional matrices and comparing the resulting vectors lexicographically. Note that, in each subcase of the definition of 3-dimensional lexicographical ordering, we flatten the remaining 2 dimensions of the matrix in the same order as we flatten all 3 dimensions in the 3-dimensional matrix ordering. This ensures compatibility between the two orderings. Suppose we swap two slices from the first dimension of the 3-dimensional matrix so that their flattened 2-dimensional slices are ordered. Trivially this must reduce the 3-dimensional ordering. Suppose we swap two slices from the second dimension of the 3-dimensional matrix so that their flattened 2-dimensional slices are ordered. Then the flattened slices may have a number of elements in common at their start. Swapping these will not affect the 3-dimensional matrix ordering. However, we will eventually meet an element in the first flattened 2-dimensional slice which is being swapped for a smaller element in the second. This moves us down the 3-dimensional matrix ordering. A similar argument holds for swapping two slices in the third and final dimension of the 3-dimensional matrix. Thus, swapping two slices in any of the three dimensions moves us down the 3-dimensional matrix ordering. The 3-dimensional matrix ordering is finite as there are only a finite number of permutations of the values in a 3-dimensional matrix, and bounded below by the sequence with values in increasing ordering. Hence, swapping slices must terminate, giving us a minimal element with respect to the multi-dimensional lexicographical ordering. □

4.2 Partial symmetry

We may only have partial row or column symmetry in a matrix model. For example, in the rack design problem, only those columns which correspond to racks of the same type are symmetric to each other. We cannot lexicographically order the columns in such a situation. It is easy, however, to generalize lexicographical ordering constraints to break such partial row or column symmetry. For each subset of rows (or columns) which are symmetric, we post a lexicographical ordering constraint on this subset. In fact, we do not even need to do this. Suppose we have a partial column symmetry. We add an extra first row to the matrix, in which we label identically those columns in the same equivalence class. Then lexicographically ordering the columns will break this partial column symmetry. Lexicographically ordering with this extra first row will separate the columns into equivalence classes which need to be ordered separately.

4.3 Value symmetry

We can also deal with symmetric values using the same techniques we have developed for dealing with symmetric variables. A variable X that takes a single value or a set of values from a domain of n indistinguishable values can be replaced by a vector of n variables, each with the 0/1 domain. This converts indistinguishable values into indistinguishable rows or columns. Consider, for example, a 2-d matrix model of the progressive party problem [4]. A variable x_{ij} in this matrix takes as values the number of the host boat visited by guest i in period j. Now host boats of the same capacity are indistinguishable. We can turn this partial value symmetry into partial variable symmetry by channelling into a 3-d 0/1 matrix (that has no value symmetry). A variable y_{ijk} in this new matrix is 1 iff the host boat k is visited by guest i in period j. Channeling constraints of the form $x_{ij} = k$ iff $y_{ijk} = 1$ link the two matrices together. The 3-d matrix has partial row symmetry along the k dimension. We can therefore use lexicographical ordering constraints to break this partial symmetry. The advantage of this approach is that we can use multi-dimensional lexicographical ordering to deal simultaneously with symmetric variables and symmetric values. We can also use the techniques outlined in the last section to deal with values which are only partially symmetric.

5 Breaking all row and column symmetry

We now identify three special cases where all row and column symmetry can be easily broken. These cases provide insight into why it is hard to break all row and column symmetry in general.

If all the values in the matrix are *distinct* (e.g. in the magic square problem), ordering the row and column with the largest value (which puts the largest value in the bottom-right corner) breaks all symmetry. It is therefore the symmetry between identical values that makes it difficult to break all row and column symmetry.

Theorem 3. *If a 2-dimensional matrix model with row and column symmetry has a solution and all values in the matrix are distinct, then it has a unique solution with the largest value placed in the bottom-right corner and the last row and last column ordered.*

Proof: The row occupied by the largest value contains distinct values that can be permuted by ordering the columns. By ordering this row, we break all possible column symmetry. Similarly, the column occupied by the largest value contains distinct values that can be permuted by ordering the rows. By ordering this column, we break all possible row symmetry. □

In fact, our proof shows that we break all symmetry even if the other rows and columns contain repeated values. Ordering the row and column with the largest value will fix all the other values in the matrix in a unique way. We do not therefore need every value in the matrix to be distinct (although this is sufficient to make the row and column with the largest value contain no repeated values).

Even when matrices have repeated values, it is still possible in certain situations to break all row and column symmetry by means of some simple symmetry breaking constraints. In particular, 0/1 matrices with a single 1 in each row can have all row and column symmetry broken. Such matrix models are quite common. For example, the 2-d matrix models used in the rack configuration problem have this form [8]. Lexicographically ordering both rows and columns fails to break all symmetry in such models.

Example 4. *Consider a 2 by 3 matrix of 0/1 variables, with the constraints that $\sum_{ij} x_{ij} = 3$, and $\sum_i x_{ij} = 1$ for all j. There are two symmetric solutions with lexicographically ordered rows and columns:*

$$\begin{pmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{pmatrix}$$

If we add the constraint that the columns are also ordered by their sum, then all row and column symmetry is broken. Note that we can have the column sums in increasing or decreasing order, depending on which is preferable.

Theorem 4. *Given a 2-dimensional 0/1 matrix that has a single 1 in each row, ordering its rows lexicographically, and its columns lexicographically and by their sums breaks all row and column symmetry.*

Proof: The top row contains a single 1. Suppose it occurs anywhere but at the top right. Then the column it occurs in will be lexicographically larger than the last column (which must start with a 0). Hence the top right corner must contain a 1. Suppose that in the next row down, the 1 occurs to the right of where it does in this row. Then the next row is not lexicographically larger. Suppose that it occurs more than one column across to the left. Then the columns in between are not lexicographically larger. Hence, the 1 in the next row down must occur either directly below or one column to the left. The only freedom is in how many consecutive rows have 1s in the same column. This symmetry is broken by ordering the sums of the columns. □

One final situation in which all row and column symmetry can be easily broken is when every row sum is different. Again, the row sums can be in increasing or decreasing order depending on which is preferable.

Theorem 5. *Given a 2-dimensional 0/1 matrix where the row sums are all different, ordering its rows by their sums as well as its columns lexicographically breaks all row and column symmetry.*

Instance	distinct #sol	total #sol	row & col lex #sol	row & col lex time	set 1st row & col #sol	set 1st row & col time	row lex #sol	row lex time	col lex #sol	col lex time
$\langle 7,7,3,3,1\rangle$	1	$\geq 2.5M$	1	1.05	216	8.1	30	2.9	30	3.6
$\langle 6,10,5,3,2\rangle$	1	$\geq 2.5M$	1	0.95	17,280	332	60,480	3,243	12	2
$\langle 7,14,6,3,2\rangle$	4	$\geq 2.5M$	24	10.63	$\geq 90,448$	–	$\geq 68,040$	–	465	55.4
$\langle 9,12,4,3,1\rangle$	1	$\geq 2.5M$	8	28.14	$\geq 5,340$	–	≥ 342	–	840	1,356
$\langle 8,14,7,4,3\rangle$	4	$\geq 2.5M$	92	171	$\geq 5,648$	–	$\geq 2,588$	–	$\geq 5,496$	–
$\langle 6,20,10,3,4\rangle$	$*$	$\geq 2.5M$	21	10.3	$\geq 538,272$	–	$\geq 429,657$	–	73	19.6

Table 1: Experimental results on BIBD instances, where ”$-$” indicates one clock hour, ”$*$” indicates that the number of distinct solutions is unknown, and ”M” stands for millions.

Proof: If a 2-dimensional matrix model with row and column symmetry has a solution, then it has a solution with the rows ordered by their sum and the columns ordered lexicographically. We can order the rows by their sums and then freely order the columns lexicographically without violating the sum of any row. If we now swap any two rows, we will break the order on their sums. Any column permutation following this row permutation cannot undo this change as column permutations do not change row sums. Thus, we have broken all row and column symmetry. □

6 Experimental results

To test the ability of lexicographical ordering constraints to break row and column symmetry, we ran some experiments on BIBD generation. This is a standard combinatorial problem from design theory. It has applications in experimental design and cryptography (see prob028 at www.csplib.org for more details).

A BIBD is an arrangement of v distinct objects into b blocks, such that each block contains exactly k distinct objects, each object occurs in exactly r different blocks, and every two distinct objects occur together in exactly λ blocks. A BIBD instance is thus determined by its parameters $\langle v, b, r, k, \lambda\rangle$. One way of modelling a BIBD is in terms of its incidence matrix, which is a v by b 0/1 matrix with exactly r ones per row, k ones per column, and with a scalar product of λ between any pair of distinct rows [4]. This matrix model has row and column symmetry since we can permute any rows or columns freely without affecting any of the constraints. This type of symmetry is often broken by setting the first row and the first column. However, this breaks less symmetry than lexicographically ordering both the rows and columns.

Table 1 shows our experimental results on some BIBD instances. We use the ECLIPSE toolkit as this provides a lexicographical ordering constraint. The instances in this table are also used in [9] and [10]. For reasons of space, we only present a representative sample of our experiments. We enforced lexicographical ordering between neighbouring pairs of rows and columns. This will ensure any two rows or columns are lexicographically ordered. We also include the results when we set the first row and the first column, and when we impose lexicographical ordering constraints only on the rows or only on the columns.

There are at least three ways of labelling the variables: along one row and then down one column, along the rows or along the columns. The best labelling strategy varies, so the table reports the best results achieved among these three strategies.

For each instance, we show the total number of distinct solutions (denoted by distinct #sol), the total number of solutions (denoted by total #sol) when no symmetry breaking constraints are used, and the total number of solutions (denoted by #sol) found as well as the runtimes (in seconds or a "$-$" whenever 1 clock hour is exceeded) for each of the symmetry breaking techniques.

Column lexicographical ordering constraints are much more efficient than row lexicographical ordering constraints. This is true for many other instances (which are not shown in the table). We conjecture that the scalar product constraint so tightly constrains the rows that little work is left to be done by the row lexicographical ordering constraints. The column lexicographical ordering constraints act orthogonally and so are more constraining.

The results confirm that lexicographically ordering rows and columns breaks most of the row and column symmetry. In a recent study [10], a binary CSP model encoded in SAT that breaks symmetries in a different way was proposed to solve several BIBD instances using SATZ, WSAT, and CLS. All its instances could be solved fast enough with our 2-d 0/1 matrix model using row and column lexicographical ordering constraints. For example, our model solves the instance $\langle 8, 14, 7, 4, 3\rangle$ in 171 seconds, and this instance was not solved in several hours with any algorithm or encoding in [10].

7 Related work

There is currently much interest in symmetry in constraint satisfaction. Existing approaches can be broadly categorised into four types. The first, as advocated here, adds symmetry-breaking constraints to the initial model in an attempt to remove symmetry *before* search starts [11]. A second method adds symmetry-breaking constraints *during* search to prune symmetric branches (e.g Backofen and Will's approach [1] or Gent and Smith's SBDS [6]). A disadvantage of methods like SBDS is that they require the symmetries to be explicitly listed and there are an exponential number of row and column symmetries. Recently, Gent and Smith have looked at combining symmetry breaking before and during search [14]. They report promising results with combined methods that break some of the symmetry using row sum and partial column lexicographical ordering. A third approach is to break symmetry by means of a *heuristic variable-ordering* that directs the search towards subspaces with a high density of non-symmetric states (e.g. [9]). Lastly, it is sometimes possible to *remodel* a problem to remove symmetry, for example via the use of set variables. However, this can produce a more complex model [13].

All of these approaches would benefit from an efficient means of automatic symmetry detection. However, symmetry detection has been shown to be graph-isomorphism complete in the general case [3]. Therefore, it is often assumed that the symmetries are known by the user. Since matrices of decision variable are common in constraint programs [4], and rows and columns in such matrices are often symmetric, making matrices first-class objects in the modelling language would give a heuristic symmetry-detection technique obvious clues as to where to look.

8 Conclusions

We have identified an important class of symmetries in constraint models: row and column symmetry. We have shown how we can lexicographically order both rows and columns to break such symmetry. Whilst lexicographically ordering rows breaks all row symmetry (and lexicographically ordering columns breaks all column symmetry), lexicographically ordering both rows and columns fails to break all row and column symmetry. Nevertheless, our experimental results show that it is effective at dealing with row and column symmetry. We have extended these results to cope with symmetries in 3 or more dimensions, partial symmetries, and symmetric values. Finally, we have identified a number of special cases where all row and column symmetry can be eliminated by means of adding some simple symmetry breaking constraints.

In our future work, we intend to look at ways of identifying row and column symmetry automatically, and at methods for enforcing these symmetry breaking constraints even more efficiently and effectively.

Acknowledgments

This work is partially supported by grant 221-99-368 of VR and grant GR/N16129 of ESPRC. The last author was supported by an ESPRC advanced research fellowship. We thank Warwick Harvey and the members of the APES group, especially Barabara Smith, for their helpful discussions.

References

[1] R. Backofen and S. Will, Excluding symmetries in concurrent constraint programming, in *Proc. of the 5th Int. Conf. on Principles and Practice of Constraint Programming*, ed., J. Jaffar, LNCS, pp. 73–87. Springer-Verlag, 1999.

[2] J. Crawford, G. Luks, M. Ginsberg, and A. Roy, Symmetry breaking predicates for search problems, in *Proc. of the 5th Int. Conf. on Knowledge Representation and Reasoning, (KR '96)*, pp. 148–159, 1996.

[3] J.M. Crawford, A theoretical analysis of reasoning by symmetry in first-order logic, in *Proc. of the AAAI workshop on tractable reasoning*, 1992.

[4] P. Flener, A. Frisch, B. Hnich, Z. Kızıltan, I. Miguel, and T. Walsh, Matrix modelling, in *Proc. of the CP-01 Workshop on Modelling and Problem Formulation*, 2001. Also available as technical report APES-36-2001 from http://www.dcs.st-and.ac.uk/∼apes/reports/apes-36-2001.ps.gz.

[5] E. Freuder, Modelling: The final frontier, in *Proc. of PACLP'99*, 1999.

[6] I.P. Gent and B.M. Smith, Symmetry breaking in constraint programming, in *Proc. of the 14th European Conf. on AI*, ed., W. Horn, pp. 599–603, 2000.

[7] P. Van Hentenryck, L. Michel, L. Perron, and J-C. Régin, Constraint programming in OPL, in *Proc. of PPDP'99*, ed., G. Nadathur, LNCS, pp. 97–116. Springer-Verlag, 1999.

[8] Z. Kızıltan and B. Hnich, Symmetry breaking in a rack configuration problem, in *Proc. of IJCAI-2001 Workshop on Modelling and Solving Problems with Constraints*. 2001.

[9] P. Meseguer and C. Torras, Exploiting symmetries within constraint satisfaction search, *Artificial Intelligence*, **129**(1–2), 133–163, 2001.

[10] S.D. Prestwich, Balanced incomplete block design as satisfiability, in *Proc. of the 12th Irish Conf. on AI and Cognitive Science*, 2001.

[11] J.-F. Puget, On the satisfiability of symmetrical constrained satisfaction problems, in *Proc. of ISMIS'93*, eds., J. Komorowski and Z.W. Ras, LNAI, pp. 350–361. Springer-Verlag, 1993.

[12] B.M. Smith, Reducing symmetry in a combinatorial design problem, in *Proc. of CP-AI-OR'01*, 2001. Available as Research Report 2001.01, School of Computing, University of Leeds.

[13] B.M. Smith, Reducing symmetry in a combinatorial design problem, in *Proc. of the IJCAI-2001 workshop on Modelling and Solving Problems with Constraints*, pp. 105–112, 2001.

[14] B.M. Smith and I.P. Gent, Reducing symmetry in matrix models:sbds v. constraints, in *Proc. of the CP-2001 workshop on Symmetry in Constraints*, 2001.