Symmetry in Matrix Models

Pierre Flener[1], Alan Frisch[2], Brahim Hnich[1], Zeynep Kızıltan[1], Ian Miguel[2], Justin Pearson[1], Toby Walsh[2]
[1]Uppsala University, Uppsala, Sweden
[2]University of York, York, England

Abstract

Many constraint satisfaction problems (such as scheduling, assignment, and configuration) can be modelled as constraint programs based on matrices of decision variables. In such matrix models, symmetry is an important feature. We study and generalise symmetry-breaking techniques, such as lexicographic ordering, and propose a labelling technique achieving the same effect.

1 Introduction

A *matrix model* contains (one or more) matrices of decision variables. For example, a natural model of a sports scheduling problem contains a 2-d matrix of decision variables, each of which is assigned a value corresponding to the match played in a given week and period [VMPR99]. In this case, the matrix is obvious in the solution to the problem: we need a *table* of fixtures. However, as we demonstrate in [FFH$^+$01], many other problems that are less obviously defined in terms of matrices of values can be efficiently represented and effectively solved using a matrix model. For example, the rack configuration problem can be modelled with a 2-d 0/1 matrix representing which cards go into which racks [KH01].

By identifying the central role played in many constraint programs by matrices of decisions variables, we can identify some patterns common to constraint programs taken from a wide range of domains. For example, common types of row and column constraint are often posted on such matrices. And, as we show in this paper, we can often deal with symmetry in such models in a uniform way. We thereby reduce the burden on the user of eliminating symmetry from their constraint programs. The results in this paper only hold for matrix models with a single matrix of decision variables.

The rest of this paper is organised as follows. We first define symmetry in matrix models. In a 2-d matrix model where all rows (or columns) are permutable, we prove that lexicographically ordering the rows (or columns) breaks all row (or column) symmetry. We also show that it is always possible to get the rows and columns lexicographically ordered despite the fact that the rows and the columns intersect. These results are extended to matrices of arbitrary numbers of dimensions as well as to matrices that contain partial row (or column) symmetry. We then discuss two practical strategies to efficiently deal with symmetry in matrix models: the usage of global constraints as well as the encoding of symmetry-breaking constraints into the labelling algorithm. Unfortunately, lexicographically ordering the rows and columns does not break all symmetry in a 2-d matrix model. However, we identify special cases where all symmetry can be broken, and show how the same techniques developed here for symmetrical variables can be used to deal with symmetrical values. Furthermore, we show that the methods presented here perform well in practice. Finally, we compare with related work, conclude, and outline future work.

2 Symmetry in matrix models

Symmetry is an important aspect of matrix models. It is usually the result of objects within the model being 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. 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. It is possible to define a more general notion of symmetry involving the permutation of both variables and values, but this is not needed here.

Two common types of symmetry in matrix models are row symmetry and column symmetry. 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 non-identical 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 rows (columns) are indistinguishable. We say that a matrix model has *partial row symmetry* (*partial column symmetry*) iff a strict subset of the rows (columns) are indistinguishable. These definitions can be easily extended to matrix models with more or less than 2 dimensions.

3 Lexicographic ordering

For ease of presentation, we here restrict ourselves to 2-d matrices. The extension to matrices of any number of dimensions is in the next section.

One common method to break row or column symmetry in a matrix model is to place lexicographic ordering constraints on the rows or columns. We say that the rows in a matrix are *lexicographically ordered* if each row is lexicographically larger (denoted $\geq_{lex}$) than the previous, and *anti-lexicographically ordered* if each row is lexicographically smaller (denoted $\leq_{lex}$) 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. The following result shows that lexicographically ordering the rows breaks all row symmetry.

Theorem 1 *If a 2-d matrix model with row symmetry has a solution, then lexicographically (or anti-lexicographically) ordering the rows breaks all row symmetry.*

Proof: As we ignore identical rows, if one row is placed after another, then it must be lexicographically larger. Swapping the later row with the earlier one would break the lexicographic ordering constraint. Hence, no two rows can be swapped and all row symmetry is broken. QED.

The dual result for columns of course also holds. 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. Ordering the rows to break the row symmetry can conflict with ordering the columns to break the column symmetry. 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. Now, $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ and $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ are the only (and symmetric) solutions. 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.

In fact, 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 them can then disrupt the lexicographic ordering on the rows. Consider, for example, a 2 by 3 matrix of variables, x_{ij}, with the constraints that $x_{ij} \in \{0, 1, 2\}$, $\sum_{ij} x_{ij} = 5$, $\sum_{ij} x_{ij}^2 = 7$, and $\sum_i x_{ij} \leq 2$. This model has both row and column symmetry. Now, $\begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 2 & 0 \end{pmatrix}$ is a solution with lexicographically ordered rows. Ordering the columns gives the solution $\begin{pmatrix} 0 & 1 \\ 1 & 1 \\ 0 & 2 \end{pmatrix}$, but destroys the lexicographic ordering on the rows. We need to reorder the rows to give a solution $\begin{pmatrix} 0 & 1 \\ 0 & 2 \\ 1 & 1 \end{pmatrix}$ that is lexicographically ordered along both rows and columns. 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 2 *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. 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 thus actually 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 both rows and columns lexicographically ordered. QED.

Unfortunately, lexicographically ordering the rows and columns *can* leave symmetry in a 2-d matrix model. 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. Now, $\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}$, $\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$, and $\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix}$ are symmetric solutions that have lexicographically ordered rows and columns. When we have more than two values, there are even smaller examples. For instance, consider a 2 by 2 matrix of variables, x_{ij}, with the constraints that $x_{ij} \in \{0, 1, 2\}$, $x_{i1} + x_{i2} \geq 1$, $\sum_{ij} x_{ij} = 3$, and $\sum_{ij} x_{ij}^2 = 5$. This model also has row and column symmetry, and $\begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix}$ and $\begin{pmatrix} 0 & 1 \\ 2 & 0 \end{pmatrix}$ are symmetric solutions that both have lexicographically ordered rows and columns.

4 Higher dimensions

A number of problems can be efficiently and effectively modelled using matrix models with more than 2 dimensions. For example, the social golfers problem (prob010 at www.csplib.org) can be modelled with a 3-d matrix whose dimensions correspond to weeks, groups, and players [FFH+01]. The entry X_{ijk} is 1 iff in week i player k plays in group j. This matrix model is symmetric along each of its three dimensions (the weeks are indistinguishable, as are the groups and players). We can generalise lexicographic ordering constraints to 3 or more dimensions to break such symmetry.

Consider a 2-d matrix. If we look along a particular dimension, we see a number of vectors at right angles to this axis.

To break symmetry, we order these vectors. A simple way is to compare them lexicographically. Now consider a 3-d matrix. If we look along a particular dimension, we see a number of 2-d "slices" of the matrix that are orthogonal to this axis. To break symmetry, we order these slices. A simple way is to flatten each slice onto a vector, and lexicographically compare the result. In n dimensions, we see slices that are $n-1$-d hypercubes that can be compared by flattening them onto vectors and lexicographically comparing the results.

Definition 1 (Multi-dimensional lexicographic ordering)

- *For 2-d matrices:* $\text{lex}^2(X[][])$ *iff*
  $\forall i\ \text{lex}(X[i][], X[i+1][])$ *and*
  $\forall j\ \text{lex}(X[][j], X[][j+1])$.
- *For 3-d matrices:* $\text{lex}^3(X[][][])$ *iff*
  $\forall i\ \text{lex}(\text{flat}(X[i][][]), \text{flat}(X[i+1][][]))$,
  $\forall j\ \text{lex}(\text{flat}(X[][j][]), \text{flat}(X[][j+1][]))$, *and*
  $\forall k\ \text{lex}(\text{flat}(X[][][k]), \text{flat}(X[][][k+1]))$.
- *For d-dimensional matrices:* $\text{lex}^d(X[][]\ldots[])$ *iff*
  $\forall i\ \text{lex}(\text{flat}(X[i][]\ldots[]), \text{flat}(X[i+1][]\ldots[]))$,
  $\forall j\ \text{lex}(\text{flat}(X[][j]\ldots[]), \text{flat}(X[][j+1]\ldots[]))$, ..., *and*
  $\forall l\ \text{lex}(\text{flat}(X[][]\ldots[l]), \text{flat}(X[][]\ldots[l+1]))$.

where $\text{flat}(X[][]\ldots[]) =$
$\text{append}(\text{flat}(X[1][]\ldots[]), \text{flat}(X[2][]\ldots[]), \ldots, \text{flat}(X[n][]\ldots[]))$
and $\text{flat}(X[]) = X[]$.

As in the 2-d case, we can show that this multi-dimensional lexicographic ordering breaks *some* of the symmetry. Unfortunately, it does not break all symmetry as the 2-d counter-examples easily generalise to 3 or more dimensions.

Theorem 3 *If an n-d matrix model ($n \geq 1$) with symmetry along each dimension has a solution, then it has a solution under the n-d lexicographic ordering.*

Proof: We give a proof for 3 dimensions. The proof generalises to higher dimensions in a straightforward way. As in the 2-d case, we give an ordering on 3-d matrices that strictly decreases each time we re-order a pair of slices. This ordering is simply that formed by flattening the 3-d matrices and comparing the resulting vectors lexicographically. Note that, in each sub-case of the definition of 3-d lexicographic ordering, we flatten the remaining 2 dimensions of the matrix in the same order as we flatten all 3 dimensions in the 3-d matrix ordering. This ensures compatibility between the two orderings. Suppose we swap two slices from the first dimension of the 3-d matrix so that their flattened 2-d slices are ordered. Trivially, this must reduce the 3-d ordering. Suppose we swap two slices from the second dimension of the 3-d matrix so that their flattened 2-d 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-d matrix ordering. However, we will eventually meet an element in the first flattened 2-d slice that is being swapped for a smaller element in the second. This moves us down the 3-d matrix ordering. A similar argument holds for swapping two slices in the third and final dimension of the 3-d matrix. Thus, swapping two slices in any of the 3 dimensions moves us down the 3-d matrix ordering. The 3-d matrix ordering is finite, as there are only a finite number of permutations of the values in the matrix, and bounded below, namely by a matrix with all dimensions lexicographically ordered. QED.

5 Partial symmetry

We often have only partial row or column symmetry in a 2-d matrix model. For example, if the first row in a 2-d matrix is already given, then only those columns that have identical values in the first row are symmetric with each other. It is easy to generalise lexicographic ordering constraints to break partial row or column symmetry. For each subset of rows (or columns) that are symmetric, we impose a lexicographic ordering. 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 that are permutable. Then lexicographically ordering the columns will break this partial column symmetry as the recursive definition of lexicographic ordering separates the columns into disjoint sets of permutable columns that have to be ordered.

6 GAC$_{\text{lex}}$

Even though multi-dimensional lexicographic ordering fails to break all symmetry, even less symmetry may be broken in practice. The problem is that explicitly defining a lexicographic ordering on two sequences of decision variables requires a large number of ordering constraints on the individual decision variables to deal with all the different cases. For this reason, matrix models are sometimes constructed with only sufficient constraints to ensure that the first few decision variables are lexicographically ordered. This will break even less symmetry than a complete multi-dimensional lexicographic ordering. We can, however, enforce generalised arc-consistency (GAC) on a pair of 1-d matrices efficiently. The ECLIPSE constraint solver provides a language primitive for $\leq_{lex}$, but it does not enforce GAC.

Enforcing GAC on the lexicographic ordering constraints between *every* pair of rows in a 2-d matrix does not guarantee global consistency (i.e., it does not guarantee that the rows can be lexicographically ordered). Indeed, for $\leq_{lex}$, consider a 2 by 3 matrix of variables, x_{ij}, with $x_{11} = x_{22} = \{0, 2\}$, $x_{21} = x_{23} = \{1\}$, $x_{12} = \{1, 2\}$, and $x_{13} = \{2\}$. Then the lexicographic ordering constraints between every pair of rows are GAC. However, there is no globally consistent solution with $x_{11} = \{2\}$. Also, for $<_{lex}$, consider a 2 by 5 matrix of 0/1 variables. The strict lexicographic ordering constraints between every pair of rows are GAC. However, there is no globally consistent solution as there are only 4 possible distinct rows.

7 Canonical labelling

Global constraints like GAC$_{\text{lex}}$ are one strategy for efficiently dealing with symmetry in matrix models. An alternative approach is to compile the symmetry-breaking constraints directly into the labelling algorithm. We provide the user with fast labelling routines — here called *canonical labelling* — that only return values consistent with the (implicit) symmetry-breaking constraints.

The definition of canonical labelling is in two parts, first the notion of a 1-d slice of an n-d matrix (such as a row or column of a 2-d matrix) being canonically labelled with respect to a group of permutations, then further the notion of a matrix being canonically labelled with respect to a sequence of groups and a particular choice of 1-d slice.

Definition 2 *A 1-d slice, S, of a matrix is* canonically labelled *with respect to a permutation group G if $S \leq_{lex} g(S)$ for all $g \in G$, where $g(S)$ is the result of partially applying the permutation g to the slice S.*

That is, the slice S is the smallest lexicographically ordered slice in the collection of all permutations of that slice in G. For example, a row of 1s and 0s is canonically labelled with respect to the group of all permutations of the columns if it is of the form

$$0 \dots 01 \dots 1$$

Definition 3 *A matrix is* canonically labelled *with respect to a sequence of 1-d slices if there exists a sequence of permutation groups $G_0 \supseteq G_1 \supseteq \cdots \supseteq G_n$ such that each 1-d slice, i, is canonically labelled with respect to the group G_i.*

Normally G_0 is taken to be the group of all row and column permutations of the matrix. In a canonically labelled matrix each slice can be thought of as a refinement or a splitting of the previous slice. Each new group G_i has less symmetries than the group G_{i-1} thus the sequence of groups represents a gradual exclusion of the symmetries of the matrix.

The following example illustrates the exclusion of symmetries on a 2-d matrix. To see how each group is a refinement of the previous group we introduce the following notation. Given a row of length n from a 2-d matrix, let $[1 \dots k_1][k_1+1 \dots k_2] \dots [k_{n-1}+1 \dots n]$ denote the group generated by $\text{Sym}_S(1 \dots k_1) \cup \text{Sym}_S(k_1+1 \dots k_2) \cup \cdots \cup \text{Sym}_S(k_{n-1}+1 \dots n)$, where $\text{Sym}_S(i \dots j)$ is the group of permutations of the columns $i, \dots, j$.

Now, the following matrix is canonically labelled with respect to row-slices and the sequence of groups indicated next to each row:

$$\begin{array}{cl} \left(\begin{array}{c} 0000111 \\ 0011001 \\ 0101010 \end{array}\right) & \begin{array}{l} G_0 = \text{All permutations} \\ G_1 = [1 \dots 4][5 \dots 7] \\ G_2 = [1,2][3,4][5,6][7] \end{array} \end{array}$$

For a 2-d matrix, canonically labelling the rows makes the columns lexicographically ordered.

Theorem 4 *If x is a 2-d matrix with canonically labelled rows, then*

$$x_{i,1} \dots x_{i,n} \leq_{lex} x_{i+1,1} \dots x_{i+1,n} \tag{1}$$

for all $1 \leq i < m$. In other words, the columns of x are in lexicographic order.

Proof: Consider two arbitrary adjacent columns, columns i and $i+1$. If x has these two columns identical, then they are in lexicographic order. Otherwise, let j be the smallest row such that $x_{i,j} \neq x_{i+1,j}$. It suffices to show that $x_{i,j} < x_{i+1,j}$. Since for all $1 \leq k < j$ we have that $x_{i,k} = x_{i+1,k}$, the variables $x_{i,j}$ and $x_{i+1,j}$ can be permuted (by the definition of canonical labelling). Hence, by the definition of canonical labelling, we have that $x_{i,j} \leq x_{i+1,j}$. Since $x_{i,j} \neq x_{i+1,j}$, it must be the case that $x_{i,j} < x_{i+1,j}$. QED.

This theorem has a converse that lexicographic ordering in one dimension implies canonical labelling in the other.

Theorem 5 *Let x be a matrix such that (1) for all $1 \leq i < m$. Then x is canonically labelled. That is, for all $1 \leq i < m$ and $1 \leq j \leq n$, if $x_{i,j}$ and $x_{i+1,j}$ can be permuted, then $x_{i,j} \leq x_{i+1,j}$.*

Proof: Consider two arbitrary adjacent columns, columns i and $i+1$. If these two columns are labelled identically, then they are canonically labelled. Otherwise, let j be the smallest row such that $x_{i,j} \neq x_{i+1,j}$. Observe that $x_{i,k}$ and $x_{i+1,k}$ can be permuted if and only if $1 \leq k \leq j$. Thus it suffices to show that $x_{i,k} \leq x_{i+1,k}$ for all $1 \leq k \leq j$. In the case in which $k < j$, we have that $x_{i,k} = x_{i+1,k}$, since j is the smallest row at which they differ. In the case in which $k = j$, by (1) it must be the case that $x_{i,k} < x_{i+1,k}$. QED.

Having proved that canonically labelling the rows (or columns) implies lex-ordering the columns (or rows) together with Theorem 2 (if there is a solution then there is a row and column lex-ordered solution) implies that for any 2-d matrix there always is an equivalent matrix that is canonically labelled along both dimensions. This can even be shown directly for n-d matrices. A strategy to achieve double lexicographic ordering is then to canonically label both the rows and the columns at the same time.

8 All symmetries

The symmetry group of a matrix model with row and column symmetry is generated by two sets of generators, namely $R = \{r_{ij}\}$ where i and j run over the rows in the matrix, and $C = \{c_{ij}\}$ where i and j run over the columns in the matrix. A generator r_{ij} swaps the two rows i and j, while a generator c_{ij} swaps the two columns i and j. The first important observation is that R generates the complete symmetry group on the rows (i.e., any permutation of the rows is possible), and the converse is true of C.

Theorem 6 *The group of symmetries of a matrix model is isomorphic to the group $R' \times C'$, where R' is the group of all possible row permutations and C' is the group of all possible column permutations.*

Proof: Observe that given a row generator $r \in R$ and a column generator $c \in C$, we have $rc = cr$, that is swapping two rows and then swapping two columns gives the same result as swapping the columns first and then swapping the rows. Then any element

$$g_1 g_2 \cdots g_n$$

of the symmetry group, where g_i is from R or C, can be rewritten by swapping the elements around as a word

$$r_1 r_2 \cdots r_m c_1 \cdots c_{m'}$$

where all r_k are from R and all c_k are from C. Since $R' \cap C'$ is just the identity element, and since R' and C' are both normal subgroups of the symmetry group of the matrix, we have the desired result. QED.

As a corollary, for an n by m matrix, the symmetry group has $n!m!$ elements. The question is then, given multi-dimensional lexicographic ordering or its equivalent, multi-dimensional canonical labelling, when are *all* symmetries broken?

We first identify a special case where all the row and column symmetries can be easily broken. If all the values in the matrix are *distinct*, ordering the row and column with the largest value (thereby placing it 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 7 *If a 2-d 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. QED.

In fact, we will thus 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 last row and last column contain no repeated values).

There are other cases where all the symmetries are broken, but we first need to introduce the following notation. We say that a 0/1-valued m by n matrix belongs to the *class* $[r_1, \ldots, r_n] \times [c_1, \ldots, c_m]$ if the c_i and r_j are non-decreasingly ordered, for each i there exists a distinct column with sum c_i, and for each j there exists a distinct row with sum r_j. Obviously, this class will be empty if $\sum r_j \neq \sum c_i$. The sum $r_1 + \cdots + r_n = k = c_1 + \cdots + c_m$ is referred to as a *partition* of k.

Two questions arise about these classes of matrices. First, for a given class, how many distinct (that is non-symmetric) matrices are there? (This is the orbit-counting problem in group theory.) Second, for a given class, does multi-dimensional lexicographic ordering break all symmetries? The duals of the following results also hold.

Theorem 8 *The class* $[r_1, \ldots, r_n] \times [c_1, \ldots, c_m]$ *is empty if there is a* j *with* $r_j > m$.

Proof: If there are only n rows, then a column cannot contain more than n ones. QED.

This simple theorem is remarkably useful in removing cases when classifying matrices by counting rows and columns. In fact, we conjecture that there is no solution in the class $[r_1, \ldots, r_n] \times [c_1, \ldots, c_m]$ if *and only if* the condition of the previous theorem or its dual holds.

Matrices of the class $[1, \ldots, 1] \times [c_1, \ldots, c_m]$ can have all row and column symmetry broken, but not by lexicographic ordering alone. Such matrix models are quite common. For example, the 2-d matrix models we used in the rack configuration problem all have this form [KH01]. Lexicographically ordering both rows and columns fails to break all symmetry in such models. For instance, $\begin{pmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}$ and $\begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{pmatrix}$ are symmetric, but have lexicographically ordered rows and columns. However, if we also add the constraint that the columns are ordered by their sum, then all row and column symmetry is broken.

Theorem 9 *Given a 2-d 0/1 matrix of the class* $[1, \ldots, 1] \times [c_1, \ldots, c_m]$*, ordering its rows lexicographically as well as 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. QED.

Another special case is when a row (column) contains only ones. The following result shows that this row (column) affects neither the number of distinct solutions nor the impact of lexicographic ordering.

Theorem 10 *Consider an* m *by* n *matrix in the class* $[r_1, \ldots, r_n] \times [c_1, \ldots, c_m]$*, where* $r_n = m$*. The number of distinct solutions under row and column symmetry is equal to the number of distinct solutions under row and column symmetry of the class* $[r_1, \ldots, r_{n-1}] \times [c_1 - 1, \ldots, c_m - 1]$*. Further, lexicographic ordering of both the rows and the columns breaks all the symmetries in the class* $[r_1, \ldots, r_n] \times [c_1, \ldots, c_m]$ *if and only if it breaks all the symmetries in the class* $[r_1, \ldots, r_{n-1}] \times [c_1 - 1, \ldots, c_m - 1]$.

Proof: There is a row of all ones, so any permutation will leave this row unchanged, so equivalent matrices in $[r_1, \ldots, r_{n-1}] \times [c_1 - 1, \ldots, c_m - 1]$ are equivalent in $[r_1, \ldots, r_n] \times [c_1, \ldots, c_m]$ upon adding an extra row of all ones. Further, there are no new equivalent matrices in $[r_1, \ldots, r_n] \times [c_1, \ldots, c_m]$ that are not in the class with this row deleted. To see that lexicographic ordering of $[r_1, \ldots, r_{n-1}] \times [c_1 - 1, \ldots, c_m - 1]$ breaks all the symmetries if and only if it breaks all the symmetries of $[r_1, \ldots, r_n] \times [c_1, \ldots, c_m]$, observe that the rows of all ones only has one place to go in the lexicographic ordering, namely at the bottom. QED.

Given a partition $\lambda = \alpha_1 + \cdots + \alpha_p$ with all $\alpha_i > 0$, we say that the partition $\lambda^* = \beta_1 + \cdots + \beta_{\alpha_p}$ is *conjugate to* λ if every β_{α_p+1-j} is the number of indices i for which

$alpha_i \geq j$. This can be visualised by arranging λ as left-aligned rows of bullets and then reading the columns to get the conjugate partition λ^*. For example, the partition $2+2+3$ has the conjugate $1+3+3$:

$$\begin{array}{lll} \bullet & \bullet & \\ \bullet & \bullet & \\ \bullet & \bullet & \bullet \end{array}$$

Theorem 11 *The set of matrices in the class $[\lambda] \times [\lambda^*]$, where λ^* is the conjugate partition to λ, contains only one distinct matrix, and further lexicographically ordering the rows and columns breaks all symmetries.*

Proof: Observe that given conjugate partitions λ and λ^*, there is at least one row of all ones and one column of all ones. We use the previous theorem and its dual version on the columns to delete the row of ones and the column of ones from consideration. Then, using a similar argument, any row or column of zeroes can be deleted. This sub-matrix then belongs to a class $[\mu_1] \times [\mu_2]$ where μ_1 and μ_2 are again conjugate. Repeat this argument until an empty or 1 by 1 matrix (with a 1 value) is obtained, where all symmetries are trivially broken by lexicographic ordering and further there is only one distinct such matrix. QED.

9 Symmetrical values

We can often deal with symmetrical values using the same techniques developed for symmetrical variables. By introducing a 0/1 matrix whose final dimension corresponds to the value, we can convert indistinguishable values into symmetrical variables. For example, in the social golfers problem (prob010 at www.csplib.org), players are indistinguishable and introducing an extra dimension into the matrix corresponding to the players converts this value symmetry into variable symmetry [FFH$^+$01].

10 Experimental results

Balanced incomplete block design (BIBD) generation is a standard combinatorial problem from design theory. It has applications in experimental design and cryptography (prob028 at www.csplib.org). 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 explained 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 [FFH$^+$01].

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, (these instances are also treated in [MT01] and [Pre01]). We here only present a representative sample of our experiments. We enforced lexicographic ordering between neighbouring pairs of rows and columns. With row and column lexicographic ordering constraints, we labelled along one row and then down one column, as this is more efficient than labelling along the rows or down the columns on these instances. However, there are some instances (not shown in the table) where labelling along the rows is much more efficient than labelling the rows and columns. With the first row and column set, the best labelling strategy varies, so the table reports the best results achieved among these three strategies. With row lexicographic-ordering constraints, the best strategy is to label the columns, and with column lexicographic-ordering constraints, the best strategy is to label the rows. Column lexicographic-ordering constraints are much more efficient than row lexicographic-ordering constraints. This is true for many other instances (which are not shown in the table). Note that lexicographically ordering the columns (or rows) is equivalent to canonically labelling the rows (or the columns). We conjecture that the λ constraint so tightly constrains the rows that little work is left to be done by the row lexicographic-ordering constraints. The column lexicographic-ordering constraints act orthogonally and so are more constraining.

The results confirm that row and column lexicographic ordering breaks more symmetry than the other symmetry-breaking methods. It is also the fastest method on all the problem instances we tried. In a recent study [Pre01], 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 much faster with our 2-d 0/1 matrix model using row and column lexicographic-ordering constraints. For example, our model solves the instance $\langle 8, 14, 7, 4, 3 \rangle$, which was not solved in several hours with any algorithm or encoding in [Pre01], in just 238.50 seconds.

11 Related work

There is currently much interest in the exploitation of symmetry in constraint satisfaction problems. 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 [Pug93]. A second method adds symmetry-breaking constraints *during* search, as embodied by the SBDS system [GS00]. Upon backtracking, SBDS adds symmetry-breaking constraints that prune parts of the search tree symmetrical to those already explored. See also [BW99] for a related approach. Third, symmetry may also be broken through a *heuristic variable-ordering*, directing the search towards subspaces with a high density of non-symmetric states by breaking as many symmetries as possible with each new variable assignment. The variety-maximisation heuristic presented in [MT01] is a combination of this technique with the popular smallest-domain variable-ordering heuristic. Lastly, it is often possible to *remodel* a problem to remove symmetry, for example via the use of set variables. In general, though, this can produce a very complex model, as presented in [Smi01].

All of these approaches would benefit from an efficient

		double lex		set 1st row & col		row lex		col lex	
Instance	#sol	#sol	time	#sol	time	#sol	time	#sol	time
$\langle 7,7,3,3,1 \rangle$	1	1	1.05	216	8.1	30	2.9	30	3.6
$\langle 6,10,5,3,2 \rangle$	1	1	0.95	17280	332.0	60480	3243.0	12	2.0
$\langle 7,14,6,3,2 \rangle$	4	24	10.63	–	–	–	–	465	55.4
$\langle 9,12,4,3,1 \rangle$	1	8	28.14	–	–	–	–	840	1356.0
$\langle 8,14,7,4,3 \rangle$	4	92	238.50	–	–	–	–	–	–

Table 1: Experimental results on BIBD instances, where "–" indicates that no solution could be found in one CPU hour.

means of automatic symmetry detection. However, symmetry detection has been shown to be graph-isomorphism complete in the general case [Cra92]. Therefore, it is often assumed that the symmetries are known by the user. Since matrices are such a common occurrence in models of constraint satisfaction problems [FFH$^+$01], with rows and columns 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.

12 Conclusions

Matrix models have been studied from a symmetry point of view. Different types of symmetries in matrix models have been identified, such as (partial) row and column symmetry. Furthermore, methods have been developed to efficiently deal with symmetry in matrix models, either using global constraints, such as $\leq_{lex}$, or encoding the symmetry-breaking constraints directly into the labelling algorithm. In order to further reduce the burden on the user of eliminating symmetry from their constraint programs, our next step is to identify such symmetries automatically.

Acknowledgements

This research is supported by grant 221-99-369 of the Swedish Research Council (VR) and by EPSRC grant GR/N16129. The last author is supported by an EPSRC advanced research fellowship.

References

[BW99] R. Backofen and S. Will. Excluding symmetries in concurrent constraint programming. In J. Jaffar, editor, *Proceedings of the 5th International Conference on Principles and Practice of Constraint Programming*, pages 73–87. Springer-Verlag, 1999. Lecture Notes in Computer Science 1713.

[Cra92] J.M. Crawford. A theoretical analysis of reasoning by symmetry in first-order logic. In *Proceedings of the AAAI workshop on tractable reasoning*, 1992.

[FFH$^+$01] P. Flener, A. Frisch, B. Hnich, Z. Kızıltan, I. Miguel, and T. Walsh. Matrix modelling. Technical Report APES-36-2001, APES group, 2001. Available from http://www.dcs.st-and.ac.uk/~apes/reports/apes-36-2001.ps.gz. In *Proceedings of the CP-01 Workshop on Modelling and Problem Formulation*. International Conference on the Principles and Practice of Constraint Programming, 2001.

[GS00] I.P. Gent and B.M. Smith. Symmetry breaking in constraint programming. In W. Horn, editor, *Proceedings of the 14th European Conference on Artificial Intelligence*, pages 599–603, 2000.

[KH01] Z. Kızıltan and B. Hnich. Symmetry breaking in a rack configuration problem. In *Proceedings of the IJCAI-2001 Workshop on Modelling and Solving Problems with Constraints*, pages 83–88, 2001. International Joint Conference on Artificial Intelligence, 2001.

[MT01] P. Meseguer and C. Torras. Exploiting symmetries within constraint satisfaction search. *Artificial Intelligence*, vol. 129, no. 1–2, pp. 133–163, 2001.

[Pre01] S.D. Prestwich. Balanced incomplete block design as satisfiability. In *Proceedings of the 12th Irish Conference on Artificial Intelligence and Cognitive Science*, 2001.

[Pug93] J.-F. Puget. On the satisfiability of symmetrical constrained satisfaction problems. In J. Komorowski and Z.W. Ras, editors, *Proceedings of ISMIS'93*, pages 350–361. Springer-Verlag, 1993. Lecture Notes in Artificial Intelligence 689.

[Smi01] B.M. Smith. Reducing symmetry in a combinatorial design problem. In *Proceedings of the IJCAI-2001 workshop on Modelling and Solving Problems with Constraints*, pages 105–112, 2001. International Joint Conference on Artificial Intelligence, 2001.

[VMPR99] P. Van Hentenryck, L. Michel, L. Perron, and J.-C. Régin. Constraint programming in OPL. In G. Nadathur, editor, *Principles and Practice of Declarative Programming*, pages 97–116. Springer-Verlag, 1999. Lecture Notes in Computer Science 1702.