Symmetry-Breaking Constraints for Matrix Models

Zeynep Kiziltan[1] and Barbara M. Smith[2]

[1] Department of Information Science, Uppsala University, Sweden.
Zeynep.Kiziltan@dis.uu.se
[2] School of Computing and Engineering, University of Huddersfield, England.
b.m.smith@hud.ac.uk

Abstract. Many CSPs can be effectively represented and efficiently solved using matrix models, in which the matrices may have symmetry between their rows and/or columns. Eliminating all such symmetry can be very costly as there are in general exponentially many symmetries. Cost-effective methods have been proposed to break much of the symmetry, if not all. In this paper, we continue with this line of research, and propose several symmetry-breaking constraints. Experimental results confirm their value.

1 Introduction

Symmetry in a CSP model is an important issue as the exploration of symmetric but essentially equivalent branches in a search tree may significantly slow down the search process. This has interested many researchers in recent years, and several techniques have been developed to address the issue of eliminating symmetry in CSP models.

An important class of symmetries in constraint programming arises from matrices of decision variables where any two rows and any two columns can be interchanged [2]. For example, in the 2-d matrix model of the social golfers problem[1], the weeks are indistinguishable, and so are the groups. Any solution is thus symmetric to another obtained by interchanging any two rows representing two periods, and/or any two columns representing two weeks. In attempting to find all solutions[2], an exponential number of search states will be visited unnecessarily if row and column symmetries are not eliminated.

On the other hand, eliminating all symmetries is not so easy, since the effort required may also be exponential. In such a case, removing all symmetries is not cost-effective: the overhead introduced is as bad as the wasted search effort with no symmetry breaking. Methods to reduce significantly the row and column symmetry in matrix models with only a polynomial effort have been proposed. For instance, we can lexicographically order both the rows and columns [2],

[1] prob010 at www.csplib.org
[2] Note that symmetry elimination is beneficial in exhaustive search as opposed to in first-solution search [6].

or we can apply Symmetry Breaking During Search (SBDS) [5] on the columns and combine this with adding symmetry-breaking constraints on the rows before search starts. Such constraints must not get in the way of permuting the columns. One strategy is to insist that the vectors are ordered by their sums [7].

In this paper, we first explore other symmetry-breaking constraints that can be posed on a matrix model so as to remove a significant amount of row and column symmetry. We then empirically show the effectiveness of the proposed constraints in comparison with the related work.

2 Related work

In recent years, many techniques have been developed towards eliminating symmetry in CSPs models. One can for instance specify some constraints that are satisfied by a subset of the solutions within a set of equivalent symmetric solutions of a CSP model. Ideally, only one solution in an equivalence class satisfies these constraints, and thus all symmetries are eliminated. These constraints are called 'symmetry-breaking' constraints. There are at least two ways of posting symmetry-breaking constraints in constraint programming: by adding symmetry-breaking constraints to the model before search starts (e.g. [1]), or during search (e.g. SBDS [5]). In the rest of this paper, we will refer to the former when we talk about adding symmetry-breaking constraints.

In [2], it is shown that if a 2-d matrix model with row and column symmetry has a solution then it has a solution with the rows and columns ordered lexicographically. Hence, imposing lexicographic ordering constraints on both the rows and the columns (called double-lex) does not remove any unique solutions. The lexicographic ordering constraints are symmetry-breaking constraints and are posted between every adjacent pair of rows or columns. For an $n \times n$ matrix model with row and column symmetry, $O(n)$ symmetry-breaking constraints are posted. Whilst imposing lexicographic ordering constraints on the rows (columns) breaks all row (column) symmetry, double-lex does not break all row and column symmetries. However, experimental results show that double-lex is in practice effective at dealing with row and column symmetries.

In theory, SBDS can be used to eliminate all row and column symmetries of a matrix model. However, since, an $n \times m$ matrix has $n!m! - 1$ symmetries other than identity, breaking all symmetries would require as many SBDS functions, so SBDS can only be used for small matrices (e.g. 3×3 and 4×4). Therefore, in [7], using a subset of the full SBDS functions is proposed, to reduce rather than eliminate row and column symmetries in large matrices. Since row symmetry alone (or column symmetry alone) can entirely be eliminated by row (column) transpositions, a promising subset is the row transpositions *and* the column transpositions, which require only $O(n^2)$ SBDS functions for an $n \times n$ matrix. Adding all combinations of a row transposition and a column transposition to the row and column transpositions would eliminate even more symmetry. This, however, increases the number of SBDS functions to $O(n^4)$.

In [4], an implementation of SBDS combined with the GAP system (Groups, Algorithms and Programming) is presented, which allows the symmetries of a CSP to be represented by the generators of the group. This is a promising approach, but the experiments reported show that the current implementation is much slower than double-lex ordering to solve a BIBD problem represented by a 6×10 matrix.

In [7], combining SBDS with adding symmetry-breaking constraints is explored. One can for instance use column transpositions in SBDS to remove the column symmetry (called col-trans). This would require only $O(n^2)$ SBDS functions. To reduce the row symmetry, one can add constraints that order the rows by their sums. This method (called col-trans+row-sum) does not break all symmetries because it does not consider combinations of row and column permutations. However, it is very practical for reducing symmetry in large matrices.

3 Symmetry-breaking constraints

The effect of col-trans can also be achieved by posing lexicographic constraints on the columns (called col-lex), because either of col-trans or col-lex breaks all column symmetry. Col-trans+row-sum thus removes the same amount of symmetry as col-lex combined with row-sum (col-lex+row-sum). There are other factors which might make one of these strategies preferable to the other: this is discussed in Section 5.

The following theorem is given in [2]:

Theorem 1. *Given a 2-d matrix model 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.*

If a 2-d matrix model with row and column symmetry has a solution, then it has a solution with the rows ordered by their sums. Now any two rows may not be permuted even if any column permutation follows this row permutation. We can now order the columns lexicographically as the row sums are invariant to column permutations. Hence, all symmetry is broken.

What happens when some row sums are equal? In this case, even if we order the rows by their sums and the columns lexicographically, we do not break all symmetry because any two rows with the same sum can be swapped, and then the columns can be lexicographically ordered. We now show that when some row sums are repeated, we can order the rows by their sums and the columns lexicographically, *and* order the rows having the same sum lexicographically. This will remove more row and column symmetry.

Theorem 2. *If a 2-d matrix model with row and column symmetry has a solution, then it has a solution with the rows ordered by their sums and the columns ordered lexicographically, as well as the rows with equal sum ordered lexicographically.*

Proof. We can order the rows of any matrix by their sums. Now the rows having different sums may not be permuted by Theorem 1. On the other hand, the rows with equal sum can freely be permuted. Hence, imposing an ordering on the rows by their sums makes the model have *partial* row symmetry[3]. We reduce partial row symmetry, and column symmetry by imposing a lexicographic ordering on every subset of the rows that can be permuted (in this case the rows with equal sums), and a lexicographic ordering on the columns [2]. □

The method in Theorem 2, called col-lex+row-sum(+row-lex), reduces to double-lex if the row sums are all the same. If the row sums are all different, then it reduces to col-lex+row-sum, which in that case breaks all symmetry [2]. This method thus combines the power of double-lex when the row sums are all the same, with the power of row-sum constraints when the row sums are all different. When the row sums are neither all the same nor all different, col-lex+row-sum(+row-lex) eliminates more symmetry than col-lex+row-sum. For instance, the solution $\begin{pmatrix} 0\,0\,1 \\ 1\,1\,0 \\ 0\,1\,1 \end{pmatrix}$ is symmetric to $\begin{pmatrix} 0\,0\,1 \\ 0\,1\,1 \\ 1\,1\,0 \end{pmatrix}$, and is eliminated by col-lex+row-sum(+row-lex) but not by col-lex+row-sum.

However, note that col-lex+row-sum(+row-lex) can also leave symmetry in a 2-d matrix model. Consider a 3×3 0/1 matrix model that has both row and column symmetry. The solutions $\begin{pmatrix} 0\,0\,1 \\ 0\,1\,0 \\ 1\,0\,1 \end{pmatrix}$ and $\begin{pmatrix} 0\,0\,1 \\ 0\,1\,0 \\ 1\,1\,0 \end{pmatrix}$ are symmetric solutions that have rows ordered by their sums, columns ordered lexicographically, as well as the rows with equal sums ordered lexicographically.

We can also treat each row of the matrix as a multiset, i.e. as a set with repetitions, and insist that each row, as a multiset, should be no greater than the rows below it. We can, for instance, imagine the values in rows 1 and 2 sorted in descending order: the largest value in row 1 must be no greater than the largest value in row 2; if they are the same, we compare the 2nd largest values, and so on. Multiset ordering of the rows is stronger than row-sum ordering because non-identical rows with the same sum may be different when considered as multisets. Hence, combining multiset row ordering with col-lex (called col-lex+row-multiset) would reduce more symmetry than col-lex+row-sum does. For instance, the matrix $\begin{pmatrix} 0\,1\,2\,3 \\ 0\,3\,3\,3 \\ 1\,2\,3\,3 \end{pmatrix}$ is symmetric to $\begin{pmatrix} 0\,1\,2\,3 \\ 1\,2\,3\,3 \\ 0\,3\,3\,3 \end{pmatrix}$, and is eliminated by col-lex+row-multiset but not by col-lex+row-sum.

We can achieve multiset ordering by assigning a weight to each value, summing the weights along each row and constraining the sums to be non-decreasing. Since we first order the rows in increasing order of maximum element, and consider other elements in the rows only if there is a tie, the weight should increase with the value. We want to ensure that, for instance, for any possible value k,

[3] A matrix model has partial row (resp. column) symmetry iff strict subset(s) of the rows (resp. columns) of one of its matrices are indistinguishable [2].

a row containing say one element with value k and $n-1$ 0s has greater weight than a row in which each of the n elements is $k-1$, where n is the number of columns. A suitable weighting assigns the weight n^r to the value r. Thus the first row in the example has total weight $n^k + n - 1$ and the second row has weight n^k. The ordering can be implemented by introducing new constrained variables w_{ij} such that $w_{ij} = n^k$ iff $x_{ij} = k$. Then the constraint that row i is not greater than row $i+1$, both considered as multisets, is: $\sum_j w_{i,j} \leq \sum_j w_{i+1,j}$.

With this implementation, multiset ordering appears to be no more expensive than row-sum ordering. However, it will not be possible to implement it in this way unless n^l is manageable, where l is the largest possible value in the matrix. Moreover, if there are only two possible values in the domains, col-lex+row-multiset is equivalent to col-lex+row-sum.

It is possible to improve col-lex+row-multiset even further. We can imagine that two non-identical rows may have the same weighted sum, so that multiset ordering is not able to distinguish between them. We may, however, distinguish them by ordering them lexicographically. We can easily show that we do not lose any solutions by this method.

Theorem 3. *Given a 2-d matrix model where the row weighted-sums are all different, ordering its rows by their weighted-sums as well as its columns lexicographically breaks all row and column symmetry.*

Proof. The proof of theorem 1 can easily be adapted for weighted-sums. □

Theorem 4. *If a 2-d matrix model with row and column symmetry has a solution, then it has a solution with the rows ordered by their weighted-sums and the columns ordered lexicographically, as well as the rows with equal weighted-sum ordered lexicographically.*

Proof. The proof of theorem 2 can easily be adapted for weighted-sums. □

With this new method, called col-lex+row-multiset(+row-lex), if the row weighted-sums are all the same then we will end up with double-lex. Also, when the domain size is 2, the strategy will specialise into col-lex+row-sum(+row-lex) which is stronger than col-lex+row-sum. If the row weighted-sums are all different then we will end up with col-lex+row-multiset, which breaks all symmetry when the row weighted-sums are all different.

This method thus combines the power of double-lex when the row weighted-sums are all the same, with the power of col-lex+row-sum(+row-lex) when the domain size is 2, and with the power of weighted-sum constraints when the row weighted-sums are all different. Hence, col-lex+row-multiset(+row-lex) eliminates more symmetry than col-lex+row-multiset. For instance, the matrix $\begin{pmatrix} 0\,1\,2\,3 \\ 3\,3\,3\,0 \\ 0\,3\,3\,3 \end{pmatrix}$ is symmetric to $\begin{pmatrix} 0\,1\,2\,3 \\ 0\,3\,3\,3 \\ 3\,3\,3\,0 \end{pmatrix}$, and is eliminated by col-lex+row-multiset(+row-lex) but not by col-lex+row-multiset. However, note that col-lex+row-multiset(+row-lex) can also leave symmetry in a 2-d matrix model.

	3x3 matrices					4x4 matrices	
Strategy	2 vals	3 vals	4 vals	5 vals	6 vals	2 vals	3 vals
SBDS	36	738	8240	57675	289716	317	90492
no symmetry-breaking	512	19683	262144	$\geq$ 1.5M	$\geq$ 1.5M	65536	$\geq$ 1.5M
double-lex	45	1169	14178	102251	520017	650	250841
col-lex+row-sum	42	1007	11174	75715	368154	567	190671
col-lex+row-sum(+row-lex)	39	832	9264	63829	316329	420	120281
col-lex+row-multiset	42	863	9128	61555	302386	567	136665
col-lex+row-multiset(+row-lex)	39	804	8710	59716	296337	420	109545
row & col. transpositions +combinations	36	786	8985	63052	315428	353	114966

Table 1. Number of matrices found using different symmetry-breaking strategies.

4 Experimental results

We have carried out some experiments to compare different ways of reducing symmetry in a matrix model with row and column symmetry. In the experiments, we ignore any constraints on the matrix to be constructed. We only specify the size of the matrix and the domain size of the elements in the matrix, and then want to find a set of matrices such that no two can be generated from each other by permuting the rows and/or columns. We here consider small matrices only, so that we can compare the results with the results of SBDS, which eliminates symmetry entirely but is applicable, at present, only to small matrices. Table 1 shows how many solutions each technique finds, in comparison with the number of distinct solutions given by SBDS, and also in comparison with no symmetry-breaking. Note that M stands for a million.

Note that using SBDS does not hinder the variable and value ordering; if we order the matrices in any symmetry equivalence class according to the variable and value ordering being used for the search, then the first matrix in each equivalence class is the one that will be found. On the other hand, if we are adding symmetry-breaking constraints to the model before search, we have to be more careful in choosing the search strategy: the constraints should be designed with a variable and value ordering in mind (or v.v.); otherwise, the first matrix in an equivalence class according to the ordering may conflict with the symmetry-breaking constraints.

The lexicographic ordering constraints discussed in [2] are consistent with a variable ordering which considers the top row, left to right, 2nd row, left to right, and so on, and the values in ascending order. This ordering has been used for the experiments described here.

As seen in Table 1, col-lex+row-sum(+row-lex) breaks more symmetry than double-lex and col-lex+row-sum. Col-lex+row-multiset is a better strategy than col-lex+row-sum(+row-lex) when there are more than 3 possible values for each element of the matrix. On the other hand, col-lex+rom-sum(+row-lex) is a better strategy than col-lex+row-multiset when the domains of the matrix elements

have 2 or 3 values. Table 1 shows that by improving col-lex+row-multiset by lexicographically ordering equivalent rows under multiset ordering, we obtain better results than any other approximation technique mentioned so far.

Another possibility for reducing symmetry is to use SBDS with row transpositions, column transpositions, and combinations of one of each (just the transpositions without the combinations is a poor strategy, as shown in [7]). For 3×3 matrices with 2 or 3 possible values, and for 4×4 matrices with 2 values, row and column transpositions plus combinations gives better results than col-lex+row-multiset(+row-lex). For the other domains, col-lex+multiset-row(+row-lex) is the best approximation technique. As the matrix size enlarges, SBDS functions may not be manageable, so col-lex+row-multiset(+row-lex) may be preferable. However, whether multiset ordering is practicable on larger matrices depends on whether it can be implemented efficiently: the method described here would only be feasible for small matrices and a small number of values. In such a case, col-lex+row-sum(+row-lex) would be a practical approximation technique.

5 Discussions

From the experiments presented here, the best approximation to eliminating symmetry in matrices with more than three possible values, of those we have tried, is multiset ordering on one dimension as well as lexicographic ordering on the vectors that are equivalent under multiset ordering, combined with *either* lexicographic ordering *or* SBDS using just the transposition symmetries on the other dimension. Both strategies will result in the same number of matrices being generated. Which of them is the better choice in practice probably depends on other factors, such as the efficiency of the implementation of the multiset ordering and lexicographic ordering constraints, or the effect of enforcing GAC on them. Frisch et al. in [3] propose a linear time global consistency algorithm for maintaining GAC on lexicographic ordering between two vectors. One factor that favours SBDS is that whichever solution is found first in an equivalence class, symmetry-breaking constraints added during search eliminate the symmetric solutions. On the other hand, symmetry-breaking constraints added to the model implicitly specify which solution is to be found, before search starts. This may slow down the search, as the specified solution may not be found easily in the search tree.

Whether a strategy involving multiset ordering is actually practicable depends on whether the ordering can be implemented efficiently: the method described here would only be feasible for small matrices and a small number of possible values. For large matrices, row-sum ordering with lexicographic ordering on the rows with equal row-sums, combined with *either* lexicographic ordering *or* SBDS using just the transposition symmetries on the columns, or the same strategy with rows and columns reversed, would be a practical approximation technique.

The experiments reported omit any problem constraints. Even though the symmetry-breaking constraints mentioned do not interfere with problem con-

straints, the effect of reducing symmetry could be very problem dependent. For instance, if the rows of a matrix are constrained to have the same sum (e.g. the matrix model of BIBD[4]), then row-sum ordering will not be effective, and thus col-lex+row-sum will specialise into col-lex, and col-lex+row-sum(+row-lex) will specialise into double-lex. Likewise, if the rows of a matrix are constrained to have the same number of occurrences of every possible value (e.g. the matrix model of the sports scheduling problem[5]), then every row will be identical when seen as a multiset. This will reduce col-lex+row-multiset to col-lex, and col-lex+row-multiset(+row-lex) to double-lex.

Acknowledgements

We would like to thank Alan Frisch for suggesting multiset ordering, and also Pierre Flener, Brahim Hnich and Toby Walsh for valuable discussions.

References

1. J. Crawford, G. Luks, M. Ginsberg, and A. Roy. Symmetry breaking predicates for search problems. In *Proc. of KR'96, the 5th Int. Conf. on Knowledge Representation and Reasoning*, pp. 148–159, 1996.
2. P. Flener, A. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, J. Pearson, and T. Walsh. Breaking row and column symmetry in matrix models. In *Proc. of CP'2002*. Springer, 2002.
3. A. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, and T. Walsh. Global constraints for lexicographic orderings. In *Proc. of CP'2002*. Springer, 2002.
4. I.P. Gent, W. Harvey, and T. Kelsey. Groups and constraints: symmetry breaking during search. In *Proc. of CP'2002*. Springer, 2002.
5. I.P. Gent and B.M. Smith. Symmetry breaking in constraint programming. In *Proc. of ECAI'00, the 14th European Conf. on AI*, pp. 599–603. IOS Press, 2000.
6. S. Prestwich. First-solution search with symmetry breaking and implied constraints. In *Proc. of Formul'01 , the CP'01 Workshop on Modelling and Problem Formulation*, 2001.
7. B.M. Smith and I.P. Gent. Reducing symmetry in matrix models: SBDS vs. constraints. Technical report APES-31-2001. Available from http://www.dcs.st-and.ac.uk/∼apes/reports/apes-31-2001.ps.gz, 2001.

[4] prob028 at www.csplib.org
[5] prob026 at www.csplib.org