On The Complexity and Completeness of Static Constraints for Breaking Row and Column Symmetry*

George Katsirelos[1], Nina Narodytska[2], and Toby Walsh[2]

[1] CRIL-CNRS, Lens, France, email: gkatsi@gmail.com
[2] NICTA and University of NSW, Sydney, Australia, email: {nina.narodytska,toby.walsh}@nicta.com.au

Abstract. We consider a common type of symmetry where we have a matrix of decision variables with interchangeable rows and columns. A simple and efficient method to deal with such row and column symmetry is to post symmetry breaking constraints like DOUBLELEX and SNAKELEX. We provide a number of positive and negative results on posting such symmetry breaking constraints. On the positive side, we prove that we can compute in polynomial time a unique representative of an equivalence class in a matrix model with row and column symmetry if the number of rows (or of columns) is bounded and in a number of other special cases. On the negative side, we show that whilst DOUBLELEX and SNAKELEX are often effective in practice, they can leave a large number of symmetric solutions in the worst case. In addition, we prove that propagating DOUBLELEX completely is NP-hard. Finally we consider how to break row, column and value symmetry, correcting a result in the literature about the safeness of combining different symmetry breaking constraints. We end with the first experimental study on how much symmetry is left by DOUBLELEX and SNAKELEX on some benchmark problems.

1 Introduction

One challenge in constraint programming is to develop effective search methods to deal with common modelling patterns. One such pattern is row and column symmetry [1]: many problems can be modelled by a matrix of decision variables [2] where the rows and columns of the matrix are fully or partially interchangeable. Such symmetry is a source of combinatorial complexity. It is therefore important to develop techniques to deal with this type of symmetry. We study here simple constraints that can be posted to break row and column symmetries, and analyse their effectiveness both theoretically and experimentally. We prove that we can compute in polynomial time the lexicographically smallest representative of an equivalence class in a matrix model with row and column symmetry if the number of rows (or of columns) is bounded and thus remove all symmetric solutions. We are therefore able for the first time to see how much symmetry is left by these commonly used symmetry breaking constraints.

* Supported by ANR UNLOC project, ANR 08-BLAN-0289-01 and the Australian Government's Department of Broadband, Communications and the Digital Economy and the ARC.

2 Formal background

A constraint satisfaction problem (CSP) consists of a set of variables, each with a domain of values, and a set of constraints specifying allowed values for subsets of variables. When solving a CSP, we often use propagation algorithms to prune the search space by enforcing properties like domain consistency. A constraint is *domain consistent* (*DC*) iff when a variable in the scope of a constraint is assigned any value in its domain, there exist compatible values in the domains of all the other variables in the scope of the constraint. A CSP is domain consistent iff every constraint is domain consistent. An important feature of many CSPs is symmetry. Symmetries can act on variables or values (or both). A *variable symmetry* is a bijection σ on the variable indices that preserves solutions. That is, if $\{X_i = a_i \mid i \in [1, n]\}$ is a solution then $\{X_{\sigma(i)} = a_i \mid i \in [1, n]\}$ is also. A *value symmetry* is a bijection θ on the values that preserves solutions. That is, if $\{X_i = a_i \mid i \in [1, n]\}$ is a solution then $\{X_i = \theta(a_i) \mid i \in [1, n]\}$ is also. A simple but effective method to deal with symmetry is to add *symmetry breaking constraints* which eliminate symmetric solutions. For example, Crawford *et al.* proposed the general lex-leader method that posts lexicographical ordering constraints to eliminate all but the lexicographically least solution in each symmetry class [3]. Many problems are naturally modelled by a matrix of decision variables with variable symmetry in which the rows and/or columns are interchangeable [1]. We say that a CSP containing a matrix of decision variables has row symmetry iff given a solution, any permutation of the rows is also a solution. Similarly, it has column symmetry iff given a solution, any permutation of the columns is also a solution.

Running example: *The Equidistant Frequency Permutation Array (EFPA) problem is a challenging problem in coding theory. The goal is to find a set of v code words, each of length $q\lambda$ such that each word contains λ copies of the symbols 1 to q, and each pair of code words is Hamming distance d apart. For example, for $v = 4$, $\lambda = 2$, $q = 3$, $d = 4$, one solution is:*

$$\begin{array}{c} 0\,2\,1\,2\,0\,1 \\ 0\,2\,2\,1\,1\,0 \\ 0\,1\,0\,2\,1\,2 \\ 0\,0\,1\,1\,2\,2 \end{array} \qquad \text{(a)}$$

This problem has applications in communication theory, and is related to other combinatorial problems like finding orthogonal Latin squares. Huczynska et al. [4] consider a model for this problem with a v by $q\lambda$ array of variables with domains 1 to q. This model has row and column symmetry since we can permute the rows and columns and still have a solution.

3 Breaking row and column symmetry

To break all row symmetry we can post lexicographical ordering constraints on the rows. Similarly, to break all column symmetry we can post lexicographical ordering constraints on the columns. When we have both row and column symmetry, we can post a DOUBLELEX constraint that lexicographically orders both the rows and columns

[1]. This does not eliminate all symmetry since it may not break symmetries which permute both rows and columns. Nevertheless, it is often effective in practice.

Running example: *Consider again solution* (a)*. If we order the rows of* (a) *lexicographically, we get a solution with lexicographically ordered rows and columns:*

$$\begin{array}{c} 0\,2\,1\,2\,0\,1 \\ 0\,2\,2\,1\,1\,0 \\ 0\,1\,0\,2\,1\,2 \\ 0\,0\,1\,1\,2\,2 \end{array} \quad \begin{array}{c} \text{order} \\ \Rightarrow \\ \text{rows} \end{array} \quad \begin{array}{c} 0\,0\,1\,1\,2\,2 \\ 0\,1\,0\,2\,1\,2 \\ 0\,2\,1\,2\,0\,1 \\ 0\,2\,2\,1\,1\,0 \end{array} \tag{b}$$

Similarly if we order the columns of (a) *lexicographically, we get a different solution in which both rows and columns are again ordered lexicographically:*

$$\begin{array}{c} 0\,2\,1\,2\,0\,1 \\ 0\,2\,2\,1\,1\,0 \\ 0\,1\,0\,2\,1\,2 \\ 0\,0\,1\,1\,2\,2 \end{array} \quad \begin{array}{c} \text{order} \\ \Rightarrow \\ \text{cols} \end{array} \quad \begin{array}{c} 0\,0\,1\,1\,2\,2 \\ 0\,1\,0\,2\,1\,2 \\ 0\,1\,2\,0\,2\,1 \\ 0\,2\,2\,1\,1\,0 \end{array} \tag{c}$$

All three solutions are thus in the same row and column symmetry class. However, both (b) *and* (c) *satisfy the* DOUBLELEX *constraint. Therefore* DOUBLELEX *can leave multiple solutions in each symmetry class.*

The lex-leader method breaks all symmetry by ensuring that any solution is the lexicographically smallest in its symmetry class [3]. This requires linearly ordering the matrix. Lexicographically ordering the rows and columns is consistent with a linearization that takes the matrix in row-wise order (i.e. appending rows in order). We therefore consider a complete symmetry breaking constraint ROWWISELEXLEADER which ensures that the row-wise linearization of the matrix is lexicographically smaller than all its row or column permutations, or compositions of row and column permutations.

Running example: *Consider the symmetric solutions* (a) *to* (c)*. If we linearize these solutions row-wise, the first two are lexicographically larger than the third. Hence, the first two solutions are eliminated by the* ROWWISELEXLEADER *constraint.*

ROWWISELEXLEADER breaks all row and column symmetries. Unfortunately, posting such a constraint is problematic since it is NP-hard to check if a complete assignment satisfies ROWWISELEXLEADER [5, 6]. We now give our first major result. We prove that if we can bound the number of rows (or columns), then there is a polynomial time method to break all row and column symmetry. For example, in the EFPA problem, the number of columns might equal the fixed word size of our computer.

Theorem 1 *For a n by m matrix, we can check if a complete assignment satisfies a* ROWWISELEXLEADER *constraint in $O(n!nm \log m)$ time.*

Proof: Consider the matrix model $X_{i,j}$. We exploit the fact that with no row symmetry and just column symmetry, lexicographically ordering the columns gives the lex-leader assignment. Let $Y_{i,j} = X_{\sigma(i),j}$ be a row permutation of $X_{i,j}$. To obtain $Z_{i,j}$, the smallest column permutation of $Y_{i,j}$ we lexicographically sort the m columns of $Y_{i,j}$ in $O(nm \log(m))$ time. Finally, we check that $[X_{1,1}, \ldots, X_{1,m}, \ldots, X_{n,1}, \ldots, X_{n,m}] \leq_{\text{lex}} [Z_{1,1}, \ldots, Z_{1,m}, \ldots, Z_{n,1}, \ldots, Z_{n,m}]$, where $\leq_{\text{lex}}$ is the lexicographic comparison of

two vectors. This ensures that $X_{i,j}$ is lexicographically smaller than or equal to any column permutation of this row permutation. If we do this for each of the $n! - 1$ non-identity row permutations, then $X_{i,j}$ is lexicographically smaller than or equal to any row permutation. This means that we have the lex-leader assignment. This can be done in time $O(n!nm \log m)$, which for bounded n is polynomial. □

This result easily generalizes to when rows and columns are partially interchangeable. In the experimental section, we show that this gives an effective method to break *all* row and column symmetry.

4 Double Lex

When the number of both rows and columns is large, breaking all row and column symmetry is computationally challenging. In this situation, we can post a DOUBLELEX constraint [1]. However, as we saw in the running example, this may not break all symmetry. In fact, it can leave $n!$ symmetric solutions in an $2n \times 2n$ matrix model.

Theorem 2 *There exists a class of* $2n$ *by* $2n$ *0/1 matrix models on which* DOUBLELEX *leaves* $n!$ *symmetric solutions, for all* $n \geq 2$.

Proof: Consider a $2n$ by $2n$ matrix model with the constraints that the matrix contains $3n$ non-zero entries, and each row and column contains between one and two non-zero entries. This model has row and column symmetry since row and column permutations leave the constraints unchanged. There exists a class of symmetric solutions to the problem that satisfy a DOUBLELEX constraint of the form:

$$\begin{matrix} 0 & I^R \\ I^R & P \end{matrix}$$

Where 0 is a n by n matrix of zeroes, I^R is the reflection of the identity matrix, and P is any permutation matrix (a matrix with one non-zero entry on each row and column). For example, as there are exactly two possible permutation matrices of order 2, there are two symmetric 4 by 4 solutions with lexicographically ordered rows and columns:

$$\begin{matrix} 0\,0\,0\,1 \\ 0\,0\,1\,0 \\ 0\,1\,1\,0 \\ 1\,0\,0\,1 \end{matrix} \quad \text{and} \quad \begin{matrix} 0\,0\,0\,1 \\ 0\,0\,1\,0 \\ 0\,1\,0\,1 \\ 1\,0\,1\,0 \end{matrix}$$

In general, there are $n!$ row and column symmetries of P. Hence, DOUBLELEX leaves $n!$ symmetric solutions. □

Having decided to break row and column symmetry with DOUBLELEX, how do we propagate it? One option is to decompose it into two LEXCHAIN constraints, one on the rows and the other on the columns. A LEXCHAIN constraint ensures that a sequence of vectors are lexicographically ordered. Enforcing domain consistency on each LEXCHAIN constraint takes polynomial time [7]. However, this decomposition hinders

propagation. For example, in the matrix of decision variables with domains:

$$\begin{matrix} 0/1 & 0/1 & 1 \\ 0/1 & 0 & 1 \\ 1 & 1 & 1 \end{matrix}$$

LEXCHAIN constraints on the rows and columns ensure the second row is lexicographically larger than the first row and lexicographically smaller than the third, and the second column is lexicographically larger than the first column and lexicographically smaller than the third. Both such LEXCHAIN constraints are DC. However, the corresponding DOUBLELEX constraint is not since there is no solution in which the top left variable is set to 1. We might therefore consider a specialized propagator for the DOUBLELEX constraint. Unfortunately, whilst checking a DOUBLELEX constraint takes polynomial time, enforcing DC on this constraint is NP-hard. Thus, even when posting just DOUBLELEX to break row and column symmetry, there are computational limits on our ability to prune symmetric branches from the search tree.

Theorem 3 *Enforcing DC on the* DOUBLELEX *constraint is NP-hard.*

Proof: (Outline) We reduce an instance of 1-in-3SAT on positive clauses to a partially instantiated instance of the DOUBLELEX constraint with 0/1 variables. The constructed DOUBLELEX constraint has a solution iff the 1-in-3 SAT formula is satisfiable. Hence, it is NP-hard to enforce DC on the DOUBLELEX constraint [5], even with a bounded number of values. The full proof appears in [8]. □

5 Special cases

We consider two special cases where we can check a constraint that breaks all row and column symmetry in polynomial time. In both cases, we show that we can do even better than check the constraint in polynomial time. We prove that in these cases we can enforce DC on a constraint that breaks all row and column symmetry in polynomial time. This provides a counterpoint to our result that enforcing DC on DOUBLELEX is NP-hard in general.

5.1 All-different matrices

An all-different matrix is a matrix model in which every value is different. It was shown in [1] that when an all-different matrix has row and column symmetry, then ROWWISELEXLEADER is equivalent to ensuring that the top left entry is the smallest value, and the first row and column are ordered. Let ORDER1STROWCOL be such a symmetry breaking constraint.

Theorem 4 *DC can be enforced on* ORDER1STROWCOL *in polynomial time.*

Proof: Consider the n by m matrix model $X_{i,j}$. We post $O(nm)$ constraints: $X_{1,1} < \ldots < X_{n,1}$, $X_{1,1} < \ldots < X_{1,m}$, $X_{1,1} < X_{1+i,1+j}$ for $1 \leq i < n$ and $1 \leq j < m$. The constraint graph of this decomposition is acyclic. Therefore enforcing DC on the

decomposition achieves DC on ORDER1STROWCOL. Each constraint in the decomposition can be made DC in constant time (assuming we can change bounds in constant time). Hence, DC can be enforced on ORDER1STROWCOL in $O(nm)$ time. □

Note that, when applied to an all-different matrix with row and column symmetry, the general method for breaking symmetry in all-different problems proposed in [9] will post binary inequalities logically equivalent to ORDER1STROWCOL.

5.2 Matrix models of functions

A matrix model of a function is one in which all entries are 0/1 and each row sum is 1. If a matrix model of a function has row and column symmetry then ROWWISELEXLEADER ensures the rows and columns are lexicographically ordered, the row sums are 1, and the sums of the columns are in decreasing order, as was shown in [10, 11, 1]. We denote this symmetry breaking constraint as DOUBLELEXCOLSUM. Enforcing DC on DOUBLELEXCOLSUM takes polynomial time, in contrast to partial row and column interchangeability in matrix models of functions, which is NP-hard [12].

Theorem 5 *DC can be enforced on* DOUBLELEXCOLSUM *in polynomial time.*

Proof: We will show that DOUBLELEXCOLSUM can be encoded with a set of REGULAR constraints. Consider the n by m matrix model $X_{i,j}$. For each row i we introduce an extra variable Y_i and a REGULAR constraint on $[X_{i,1}, \ldots, X_{i,m}, \#, Y_i]$ where $\#$ is a delimiter between $X_{i,m}$ and Y_i. Each REGULAR constraint ensures that exactly one position in the ith row is set to 1 and the variable Y_i stores this position. The automaton's states are represented by the 3-tuple $\langle s, d, p\rangle$ where s is the row sum, d is the current position and p records the position of the 1 on this row. This automaton has $4m$ states and a constant number of transitions from each state, so the total number of transitions is $O(m)$. The complexity of propagating this constraint is $O(m^2)$. We also post a REGULAR constraint over $Y_1, \ldots, Y_n$ to ensure that they form a decreasing sequence of numbers and the number of occurrences of each value is decreasing. The first condition ensures that rows and columns are lexicographically ordered and the second condition ensures that the sums of the columns are decreasing. The states of this automaton are 3-tuples $\langle v, s, r\rangle$ where v is the last value, s is the number of occurrences of this value, and r is the number of occurrences of the previous value. This automaton has $O(n^2m)$ states, while the number of transition from each state is bounded. Therefore propagating this constraint requires time $O(n^3m)$. This decomposition is logically equivalent to the DOUBLELEXCOLSUM constraint, therefore it is sound. Completeness follows from the fact that the decomposition has a Berge acyclic constraint graph. Therefore, enforcing DC on each REGULAR constraint enforces DC on DOUBLELEXCOLSUM in $O(m^2n + n^3m)$ time. □

6 Value symmetry

Problems with row and column symmetry also often contain value symmetries. For example, the EFPA problem has row, column and value symmetry. We therefore turn to the problem of breaking row, column and value symmetry.

Running example: *Consider again the solution* (a)*. If we interchange the values 1 and 2, we get a symmetric solution:*

$$\begin{array}{ccc} 0\,2\,1\,2\,0\,1 & & 0\,1\,2\,1\,0\,2 \\ 0\,2\,2\,1\,1\,0 & \Rightarrow & 0\,1\,1\,2\,2\,0 \\ 0\,1\,0\,2\,1\,2 & (1\,2) & 0\,2\,0\,1\,2\,1 \\ 0\,0\,1\,1\,2\,2 & & 0\,0\,2\,2\,1\,1 \end{array} \quad \text{(d)}$$

In fact, all values in this CSP are interchangeable.

How do we break value symmetry in addition to breaking row and column symmetry? For example, Huczynska *et al.* write about their first model of the EFPA problem:

> *"To break some of the symmetry, we apply lexicographic ordering (lex-ordering) constraints to the rows and columns . . . These two constraint sets do not explicitly order the symbols. It would be possible to order the symbols by using value symmetry breaking constraints. However we leave this for future work."* (page 53 of [4])

We turn to this future work of breaking row, column and value symmetry.

6.1 Double Lex

We first note that the interaction of the problem and DOUBLELEX constraints can in some circumstances break all value symmetry. For instance, in our (and Huczynska *et al.*'s) model of the EFPA problem, *all* value symmetry is already eliminated. This appears to have been missed by [4].

Running example: *Consider any solution of the EFPA problem which satisfies* DOUBLELEX *(e.g.* (b) *or* (c)*). By ordering columns lexicographically,* DOUBLELEX *ensures that the first row is ordered. In addition, the problem constraints ensure λ copies of the symbols 1 to q to appear in the first row. Hence, the first row is forced to be:*

$$\overbrace{1\ldots1}^{\lambda}\overbrace{2\ldots2}^{\lambda}\ldots\overbrace{q\ldots q}^{\lambda}$$

All value symmetry is broken as we cannot permute the occurrences of any of the values.

6.2 Puget's method

In general, value symmetries may remain after we have broken row and column symmetry. How can we eliminate these value symmetries? Puget has given a general method for breaking any number of value symmetries in polynomial time [13]. Given a surjection problem in which all values occur at least once,[3] he introduces variables Z_j to represent the index of the first occurrence of each value:

$$X_i = j \Rightarrow Z_j \leq i$$
$$Z_j = i \Rightarrow X_i = j$$

[3] Any problem can be turned into a surjection problem by the addition of suitable new variables.

Value symmetry on the X_i is transformed into variable symmetry on the Z_j. This variable symmetry is especially easy to break as the Z_j take all different values. We simply need to post appropriate ordering constraints on the Z_j. Consider, for example, the inversion symmetry which maps 1 onto m, 2 onto $m-1$, etc. Puget's method breaks this symmetry with the single ordering constraint: $Z_1 < Z_m$. Unfortunately Puget's method for breaking value symmetry is not compatible in general with breaking row and column symmetry using RowWiseLexLeader. This corrects Theorem 6 and Corollary 7 in [13] which claim that, provided we use the same ordering of variables in each method, it is compatible to post lex-leader constraints to break variable symmetry and Puget's constraints to break value symmetry. There is no ordering of variables in Puget's method which is compatible with breaking row and column symmetry using the lex-leader method (or any method like DoubleLex based on it).

Theorem 6 *There exist problems on which posting* RowWiseLexLeader *and applying Puget's method for breaking value symmetry remove all solutions in a symmetry class irrespective of the ordering on variables used by Puget's method.*

Proof: Consider a 3 by 3 matrix model with constraints that all values between 0 and 8 occur, and that the average of the non-zero values along every row and column are all different from each other. This problem has row and column symmetry since we can permute any pair of rows or columns without changing the average of the non-zero values. In addition, it has a value symmetry that maps i onto $9-i$ for $i > 0$. This maps an average of a onto $9-a$. If the averages were all-different before they remain so after. Consider the following two solutions:

$$\begin{array}{ccc} 0\,2\,3 & & 0\,2\,3 \\ 4\,8\,5 & \text{and} & 4\,1\,5 \\ 7\,6\,1 & & 7\,6\,8 \end{array}$$

Both matrices satisfy RowWiseLexLeader as the smallest entry occurs in the top left corner and both the first row and column are ordered. They are therefore both the lex leader members of their symmetry class.

Puget's method for breaking value symmetry will simply ensure that the first occurrence of 1 in some ordering of the matrix is before that of 8 in the same ordering. However, comparing the two solutions, it cannot be the case that the middle square is both before *and* after the bottom right square in the given ordering used by Puget's method. Hence, whichever ordering of variables is used by Puget's method, one of these solutions will be eliminated. All solutions in this symmetry class are thus eliminated. □

We can pinpoint the mistake in Puget's proof which allows him to conclude incorrectly that his method for value symmetry can be safely combined with variable symmetry breaking methods like DoubleLex. Puget introduces a matrix of 0/1 variables $Y_{ij} \iff X_i = j$ and observes that variable symmetries σ on variables X_i correspond to row symmetries on the matrix Y_{ij}, while value symmetries θ of the variables X_i correspond to column symmetries of the matrix. Using the lex-leader method on a column-wise linearisation of the matrix, he derives the value symmetry breaking constraints on the Z variables. Finally, he claims that we can derive the variable symmetry breaking constraints on the X variables with the same method (equation (13) of [13]).

However, this requires a row-wise linearisation of the matrix. Unfortunately, combining symmetry breaking constraints based on row and column-wise linearisations can, as in our example, eliminate all solutions in a symmetry class.

In fact, we can give an even stronger counter-example to Theorem 6 in [13] which shows that it is incompatible to post together variable and value symmetry breaking constraints *irrespective* of the orderings of variables used by *both* the variable and the value symmetry breaking method.

Theorem 7 *There exist problems on which posting lex-leader constraints to break variable symmetries and applying Puget's method to break value symmetries remove all solutions in a symmetry class irrespective of the orderings on variables used by both methods.*

Proof: Consider variables X_1 to X_4 taking values 1 to 4, an all-different constraint over X_1 to X_4 and a constraint that the neighbouring differences are either all equal or are not an arithmetic sequence. These constraints permit solutions like $X_1, \ldots, X_4 = 1, 2, 3, 4$ (neighbouring differences are all equal) and $X_1, \ldots, X_4 = 2, 1, 4, 3$ (neighbouring differences are not an arithmetic sequence). They rule out assignments like $X_1, \ldots, X_4 = 3, 2, 4, 1$ (neighbouring differences form the arithmetic sequence $1, 2, 3$). This problem has a variable symmetry σ which reflects a solution, swapping X_1 with X_4, and X_2 with X_3, and a value symmetry θ that inverts a solution, swapping 1 with 4, and 2 with 3. Consider $X_1, \ldots, X_4 = 2, 4, 1, 3$ and $X_1, \ldots, X_4 = 3, 1, 4, 2$. These two assignments form a symmetry class of solutions.

Suppose we break variable symmetry with a lex-leader constraint on X_1 to X_4. This will permit the solution $X_1, \ldots, X_4 = 2, 4, 1, 3$ and eliminate the solution $X_1, \ldots, X_4 = 3, 1, 4, 2$. Suppose we break the value symmetry using Puget's method on the same ordering of variables. This will ensure that 1 first occurs before 4. But this will eliminate the solution $X_1, \ldots, X_4 = 2, 4, 1, 3$. Hence, all solutions in this symmetry class are eliminated. In this case, both variable and value symmetry breaking use the same order on variables. However, we can show that all solutions in at least one symmetry class are eliminated whatever the orders used by both the variable and value symmetry breaking.

The proof is by case analysis. In each case, we consider a set of symmetry classes of solutions, and show that the combination of the lex-leader constraints to break variable symmetries and Puget's method to break value symmetries eliminates all solutions from one symmetry class. In the first case, suppose the variable and value symmetry breaking constraints eliminate $X_1, \ldots, X_4 = 3, 1, 4, 2$ and permit $X_1, \ldots, X_4 = 2, 4, 1, 3$. In the second case, suppose they eliminate $X_1, \ldots, X_4 = 2, 4, 1, 3$ and permit $X_1, \ldots, X_4 = 3, 1, 4, 2$. This case is symmetric to the first except we need to reverse the names of the variables throughout the proof. We therefore consider just the first case. In this case, the lex-leader constraint breaks the variable symmetry by putting either X_1 first in its ordering variables or X_3 first.

Suppose X_1 goes first in the ordering used by the lex-leader constraint. Puget's method ensures that the first occurrence of 1 is before that of 4. Puget's method therefore uses an ordering on variables which puts X_3 before X_2. Consider now the symmetry class of solutions: $X_1, \ldots, X_4 = 2, 1, 4, 3$ and $X_1, \ldots, X_4 = 3, 4, 1, 2$. Puget's method eliminates the first solution as 4 occurs before 1 in any ordering that put X_3 before X_2.

And the lex-leader constraint eliminates the second solution as X_1 is larger than its symmetry X_4. Therefore all solutions in this symmetry class are eliminated.

Suppose, on the other hand, X_3 goes first in the lex-leader constraint. Consider now the symmetry class of solutions: $X_1, \ldots, X_4 = 1, 2, 3, 4$ and $X_1, \ldots, X_4 = 4, 3, 2, 1$. The lex-leader constraint eliminates the first solution as X_3 is greater than its symmetry X_2. Suppose now that the second solution is not eliminated. Puget's method ensures the first occurrence of 1 is before that of 4. Puget's method therefore uses an ordering on variables which puts X_4 before X_1. Consider now the symmetry class of solutions: $X_1, \ldots, X_4 = 1, 3, 2, 4$ and $X_1, \ldots, X_4 = 4, 2, 3, 1$. Puget's method eliminates the first solution as 4 occurs before 1 in any ordering that put X_4 before X_1. And the lex-leader constraint eliminates the second solution as X_3 is larger than its symmetry X_2. Therefore all solutions in this symmetry class are eliminated. □

6.3 Value precedence

We end with a special but common case where variable and value symmetry breaking do not conflict. When values partition into interchangeable sets, Puget's method is equivalent to breaking symmetry by enforcing value precedence [14, 15]. Given any two interchangeable values i and j with $i < j$, a value PRECEDENCE constraint ensures that if i occurs then the first occurrence of i is before that of j. It is safe to break row and column symmetry with ROWWISELEXLEADER and value symmetry with PRECEDENCE when value precedence considers variables either in a row-wise or in a column-wise order. This is a simple consequence of Theorem 1 in [14]. It follows that it is also safe to use PRECEDENCE to break value symmetry when using constraints like DOUBLELEX derivable from the lex-leader method.

7 Snake Lex

A promising alternative to DOUBLELEX for breaking row and column symmetries is SNAKELEX [16]. This is also derived from the lex leader method, but now applied to a snake-wise unfolding of the matrix. To break column symmetry, SNAKELEX ensures that the first column is lexicographically smaller than or equal to both the second and third columns, the reverse of the second column is lexicographically smaller than or equal to the reverse of both the third and fourth columns, and so on up till the penultimate column is compared to the final column. To break row symmetry, SNAKELEX ensures that each neighbouring pair of rows, $X_{1,i}, \ldots, X_{n,i}$ and $X_{1,i+1}, \ldots, X_{n,i+1}$ satisfy the entwined lexicographical ordering:

$$\langle X_{1,i}, X_{2,i+1}, X_{3,i}, X_{4,i+1}, \ldots \rangle \leq_{\text{lex}} \langle X_{1,i+1}, X_{2,i}, X_{3,i+1}, X_{4,i}, \ldots \rangle$$

Like DOUBLELEX, SNAKELEX is an incomplete symmetry breaking method. In fact, like DOUBLELEX, it may leave a large number of symmetric solutions.

Theorem 8 *There exists a class of $2n$ by $2n+1$ 0/1 matrix models on which* SNAKELEX *leaves $O(4^n/\sqrt{n})$ symmetric solutions, for all $n \geq 2$.*

Proof: Consider the following 4 by 4 matrix:

$$\begin{matrix} 0\,1\,0\,0 \\ 0\,0\,0\,1 \\ 0\,0\,1\,0 \\ 1\,0\,0\,0 \end{matrix}$$

This is a permutation matrix as there is a single 1 on each row and column. It satisfies the SNAKELEX constraints. In fact, we can add any 5th column which reading top to bottom is lexicographically larger than or equal to 0010 and reading bottom to top is lexicographically larger than or equal to 0010. We shall add a 4 bit column with 2 bits set. That is, reading top to bottom: 1100, 1010, 0110 or 0011. Note that all 4 of these 4 by 5 matrices are row and column symmetries of each other. For instance, consider the row and column symmetry σ that reflects the matrix in the horizontal axis, and swaps the 1st column with the 2nd, and the 3rd with the 4th:

$$\begin{matrix} 0\,1\,0\,0\,1 \\ 0\,0\,0\,1\,1 \\ 0\,0\,1\,0\,0 \\ 1\,0\,0\,0\,0 \end{matrix} \quad \begin{matrix} \\ \\ \Leftrightarrow \\ \sigma \end{matrix} \quad \begin{matrix} 0\,1\,0\,0\,0 \\ 0\,0\,0\,1\,0 \\ 0\,0\,1\,0\,1 \\ 1\,0\,0\,0\,1 \end{matrix}$$

In general, we consider the $2n$ by $2n$ permutation matrix:

$$\begin{matrix} 0\,1\,0\,0\,0\,0\ldots 0\,0\,0 \\ 0\,0\,0\,1\,0\,0\ldots 0\,0\,0 \\ 0\,0\,0\,0\,0\,0\ldots 0\,0\,0 \\ 0\,0\,0\,0\,0\,1\ldots 0\,0\,0 \\ \vdots\;\vdots\;\vdots\;\vdots\;\vdots\;\vdots\quad\;\vdots\;\vdots\;\vdots \\ 0\,0\,0\,0\,0\,0\ldots 1\,0\,0 \\ 0\,0\,0\,0\,0\,0\ldots 0\,0\,1 \\ 0\,0\,0\,0\,0\,0\ldots 0\,1\,0 \\ \vdots\;\vdots\;\vdots\;\vdots\;\vdots\;\vdots\quad\;\vdots\;\vdots\;\vdots \\ 0\,0\,0\,0\,1\,0\ldots 0\,0\,0 \\ 0\,0\,1\,0\,0\,0\ldots 0\,0\,0 \\ 1\,0\,0\,0\,0\,0\ldots 0\,0\,0 \end{matrix}$$

This satisfies the SNAKELEX constraints. We can add any $2n+1$th column which reading top to bottom is lexicographically larger than or equal to the $2n-1$th column and reading bottom to top is lexicographically larger than or equal to the $2n$th column. In fact, we can add any column with eactly n of the $2n$ bits set. This gives us a set of $2n$ by $2n+1$ matrices that are row and column symmetries of each other. There are $(2n)!/(n!)^2$ bit vectors with exactly n of $2n$ bits set. Hence, we have $(2n)!/(n!)^2$ matrices which satisfy SNAKELEX that are in the same row and column symmetry class. Using Stirling's formula, this grows as $O(4^n/\sqrt{n})$. □

8 Experimental results

The proof of Theorem 1 gives a polynomial method to break all row and column symmetry. This allows us to compare symmetry breaking methods for matrix models like

DOUBLELEX and SNAKELEX, not only with respect to each other but for the first time in absolute terms. Our aim is to evaluate: first, whether the worst-case scenarios identified in theorems 2 and 8 are indicative of what can be expected in practice; second, how effective these methods are with respect to each other; third, in cases where they differ significantly, how much closer the best of them is to the optimal.

To answer these questions, we experimented with different symmetry breaking constraints: DOUBLELEX, the column-wise SNAKELEX (SNAKELEX$_C$) or the row-wise SNAKELEX (SNAKELEX$_R$) [16]. We use NOSB to denote no symmetry breaking constraints. For each problem instance we found the total number of solutions left by symmetry breaking constraints ($\#s$) and computed how many of them were symmetric based on the method outlined in the proof of Theorem 1. The number of *non symmetric* solutions is equal to the number of symmetry classes ($\#ns$) if the search space is exhausted. In all instances at least one model exhausted the search space to compute the of symmetry classes, shown in the column ROWWISELEX. We use '$-$' to indicate that the search is not completed within the time limit. As the NOSB model typically could not exhaust the search space within the time limit, we use '$>$' to indicate a lower bound on the number of solutions. Finally, we used a variable ordering heuristic that follows the corresponding lex-leader variable ordering in each set of symmetry breaking constraints (i.e. row-wise snake ordering with SNAKELEX$_R$). We ran experiments in Gecode 3.3.0 on an Intel XEON X5550, 2.66 GHz, 32 GB RAM with 18000 sec timeout.

Unconstrained problems. We first evaluated the effectiveness of symmetry breaking constraints in the absence of problem constraints. This gives the "pure" effect of these constraints at eliminating row and column symmetry. We considered a problem with a matrix $m_{r\times c}$, $r \leq c = [2,6]$, $D(m_{r,c}) = [0, d-1]$, $d = [2,\ldots,5]$ whose rows and columns are interchangeable. Table 1 summarizes the results. The first part presents typical results for 0/1 matrices whilst the second part presents results for larger domains. The results support the exponential worst case in Theorems 2 and 8, as the ratio of solutions found to symmetry classes increases from 1.25 (3,3,2) to over 6 (6,6,2), approximately doubling with each increase of the matrix size. As we increase the problem size, the number of symmetric solutions left by DOUBLELEX and SNAKELEX grows rapidly. Interestingly, SNAKELEX$_C$ achieves better pruning on 0/1 matrices, while DOUBLELEX performs better with larger domains.

Constrained problems. Our second set of experiments was on three benchmark domains: Equidistant Frequency Permutation Array (EFPA), Balanced Incomplete Block Designs and Covering Array (CA) problems. We used the non-Boolean model of EFPA [4] (Table 2), the Boolean matrix model of BIBD [1] (Table 3) and a simple model of CA [17] (Table 4). We consider the satisfaction version of the CA problem with a given number of vectors b. In all problems instances the DOUBLELEX, SNAKELEX$_R$ and SNAKELEX$_C$ constraints show their effectiveness, leaving only a small fraction of symmetric solutions. Note that SNAKELEX$_C$ often leaves fewer symmetric solutions. However, it is significantly slower compared to DOUBLELEX and SNAKELEX$_R$ because it tends to prune later (thereby exploring larger search trees). For example, the number of failures for the $(5,3,3,4)$ EFPA problem is 21766, 14072 and 1129085 for DOUBLELEX, SNAKELEX$_R$ and SNAKELEX$_C$ respectively. On EFPA problems, SNAKELEX$_R$ is about twice as fast as DOUBLELEX and leaves less solutions. On the

Table 1. Unconstrained problems. Number of solutions found by posting different sets of symmetry breaking constraints. r is the number of rows, c is the number of columns, d is the size of the domains.

(r, c, d)	RowWiseLex	NoSB	DoubleLex	SnakeLex$_R$	SnakeLex$_C$
	#ns	#s	#s / time	#s / time	#s / time
$(3, 3, 2)$	36	512	45 / **0.00**	**44** / **0.00**	**44** / **0.00**
$(4, 4, 2)$	317	65536	650 / **0.00**	**577** / **0.00**	**577** / **0.00**
$(5, 5, 2)$	5624	$3.36{\cdot}10^7$	24520 / **0.05**	**18783** / 0.06	**18783** / 0.06
$(6, 6, 2)$	251610	$> 9.4{\cdot}10^9$	$2.62 \cdot 10^6$ / 22.2	$\mathbf{1.71 \cdot 10^6}$ / 22.2	$\mathbf{1.71 \cdot 10^6}$ / **18.1**
$(3, 3, 3)$	738	19683	**1169** / **0.00**	1232 / **0.00**	1232 / **0.00**
$(3, 3, 4)$	8240	$2.62{\cdot}10^5$	**14178** / 0.03	15172 / **0.02**	15172 / 0.05
$(3, 3, 5)$	57675	$1.95{\cdot}10^6$	$\mathbf{1.02{\cdot}10^5}$ / 0.19	$1.09{\cdot}10^5$ / **0.15**	$1.09{\cdot}10^5$ / 0.21
$(3, 3, 6)$	289716	$1.01{\cdot}10^7$	$\mathbf{5.20{\cdot}10^5}$ / **2.32**	$5.54{\cdot}10^5$ / 3.29	$5.54{\cdot}10^5$ / 2.83

Table 2. Equidistant Frequency Permutation Array problems. Number of solutions found by posting different sets of symmetry breaking constraints. v is the number code words, q is the number of different symbols, λ is the size of the domains.

(q, λ, d, v)	RowWiseLex	NoSB	DoubleLex	SnakeLex$_R$	SnakeLex$_C$
	#ns	#s	#s / time	#s / time	#s / time
$(3, 3, 2, 3)$	6	$1.81{\cdot}10^5$	**6** / **0.00**	**6** / **0.00**	**6** / **0.00**
$(4, 3, 3, 3)$	8	$> 3.88{\cdot}10^7$	**16** / **0.01**	**16** / 0.01	**16** / 0.16
$(4, 4, 2, 3)$	12	$> 5.87{\cdot}10^7$	**12** / **0.00**	**12** / **0.00**	**12** / 0.04
$(3, 4, 6, 4)$	1427	$> 5.57{\cdot}10^7$	11215 / 5.88	10760 / **5.36**	**8997** / 493.87
$(4, 3, 5, 4)$	8600	$> 2.03{\cdot}10^7$	61258 / 69.90	58575 / **51.62**	**54920** / 3474.09
$(4, 4, 5, 4)$	9696	$> 5.45{\cdot}10^6$	72251 / 173.72	66952 / **132.46**	**66168** / 14374.82
$(5, 3, 3, 4)$	5	$> 4.72{\cdot}10^6$	**20** / 0.36	**20** / **0.25**	**20** / 31.61
$(3, 3, 4, 5)$	18	$> 2.47{\cdot}10^7$	71 / 0.17	71 / **0.13**	**63** / 30.08
$(3, 4, 6, 5)$	4978	$> 2.08{\cdot}10^7$	77535 / 167.50	**71186** / **137.88**	–
$(4, 3, 4, 5)$	441	$> 6.55{\cdot}10^6$	2694 / 19.37	2688 / **12.80**	**2302** / 5960.43
$(4, 4, 2, 5)$	12	$> 6.94{\cdot}10^6$	**12** / 0.02	**12** / **0.01**	**12** / 1.60
$(4, 4, 4, 5)$	717	$> 6.27{\cdot}10^6$	4604 / 38.15	**4397** / **24.58**	–
$(4, 6, 4, 5)$	819	$> 4.08{\cdot}10^6$	5048 / 69.83	**4736** / **44.83**	–
$(5, 3, 4, 5)$	3067	$> 2.39{\cdot}10^6$	20831 / 403.97	**20322** / **216.93**	–
$(6, 3, 4, 5)$	15192	$> 2.16{\cdot}10^6$	$1.11{\cdot}10^5$ / 4924.41	$\mathbf{1.06{\cdot}10^5}$ / **2006.19**	–

Table 3. Balanced Incomplete Block Designs. Number of solutions found by posting different sets of symmetry breaking constraints. v is the number of objects, k is the objects in each block, every two distinct objects occur together in exactly λ blocks.

(v, k, λ)	RowWiseLex	NoSB	DoubleLex	SnakeLex$_R$	SnakeLex$_C$
	#ns	#s	#s / time	#s / time	#s / time
$(5, 2, 7)$	1	> 0	**1** / **0.01**	**1** / 0.02	**1** / 73.26
$(5, 3, 6)$	1	$> 1.51{\cdot}10^9$	**1** / **0.00**	**1** / **0.00**	**1** / 0.82
$(6, 3, 4)$	4	$> 1.29{\cdot}10^9$	**21** / 0.01	25 / **0.00**	**21** / 12.62
$(6, 3, 6)$	6	$> 1.21{\cdot}10^9$	**134** / **0.04**	146 / 0.07	**134** / 1685.58
$(7, 3, 4)$	35	$> 1.18{\cdot}10^9$	**3209** / **0.33**	9191 / 1.07	5270 / 7241.92
$(7, 3, 5)$	109	$> 1.09{\cdot}10^9$	33304 / **4.15**	85242 / 11.90	–

Table 4. Covering Arrays. Number of solutions found by posting different sets of symmetry breaking constraints. b is the number of vectors, k is the length of a vector, g is the size of the domains, t is the covering strength.

(t, k, g, b)	RowWiseLex	NoSB	DoubleLex	SnakeLex$_R$	SnakeLex$_C$
	#ns	#s	#s / time	#s / time	#s / time
$(2, 3, 2, 4)$	2	48	**2** / **0.00**	**2** / **0.00**	**2** / **0.00**
$(2, 3, 2, 5)$	8	1440	**15** / **0.00**	**15** / **0.00**	**15** / **0.00**
$(2, 3, 3, 9)$	6	$4.35{\cdot}10^6$	**12** / **0.00**	**12** / **0.00**	**12** / 1.95
$(2, 3, 3, 10)$	104	$> 5.08{\cdot}10^8$	**368** / **0.00**	370 / 0.03	372 / 7.06
$(2, 3, 3, 11)$	1499	$> 5.56{\cdot}10^8$	**6824** / **0.23**	6905 / 0.24	6892 / 26.29
$(2, 3, 4, 16)$	150	> 0	**576** / 0.72	**576** / **0.70**	–
$(2, 3, 4, 17)$	8236	> 0	**43368** / **12.43**	43512 / 12.82	–
$(2, 3, 5, 25)$	27280	> 0	$\mathbf{1.61{\cdot}10^5}$ / **1166.94**	$\mathbf{1.61{\cdot}10^5}$ / 1178.14	–
$(2, 4, 2, 5)$	5	1920	**10** / **0.00**	**10** / **0.00**	**10** / **0.00**
$(2, 4, 2, 7)$	333	$1.60{\cdot}10^7$	2285 / **0.04**	2224 / 0.07	**1850** / **0.04**
$(2, 4, 3, 9)$	5	$2.61{\cdot}10^7$	36 / 0.02	36 / **0.01**	**26** / 1102.30

CA problems DOUBLELEX and SNAKELEX$_R$ show similar results, while DOUBLELEX performs better on BIBD problems in terms of the number of solution left.

Overall, our results show that DOUBLELEX and SNAKELEX prune most of the symmetric solutions. SNAKELEX$_C$ slightly outperforms DOUBLELEX and SNAKELEX$_R$ in terms of the number of solutions left, but it explores larger search trees and is about two orders of magnitude slower. However, there is little difference overall in the amount of symmetry eliminated by the three methods.

9 Other related work

Lubiw proved that any matrix has a row and column permutation in which rows and columns are lexicographically ordered and gave a nearly linear time algorithm to compute such a matrix [18]. Shlyakhter and Flener *et al.* independently proposed eliminating row and column symmetry using DOUBLELEX [10, 11, 1]. To break some of the remaining symmetry, Frisch, Jefferson and Miguel suggested ensuring that the first row is less than or equal to all permutations of all other rows [19]. As an alternative to ordering both rows and columns lexicographically, Frisch *et al.* proposed ordering the rows lexicographically but the columns with a multiset ordering [20]. More recently, Grayland *et al.* have proposed SNAKELEX, an alternative to DOUBLELEX based on linearizing the matrix in a snake-like way [16]. An alternative way to break the symmetry of interchangeable values is to convert it into a variable symmetry by channelling into a dual 0/1 viewpoint in which $Y_{ij} = 1$ iff $X_i = j$, and using lexicographical ordering constraints on the columns of the 0/1 matrix [1]. However, this hinders propagation [15]. Finally, dynamic methods like SBDS have been proposed to remove symmetry from the search tree [21]. Unfortunately, dynamic techniques tend not to work well with row and columns symmetries as the number of symmetries is usually too large.

10 Conclusions

We have provided a number of positive and negative results on dealing with row and column symmetry. To eliminate some (but not all) symmetry we can post static constraints like DOUBLELEX and SNAKELEX. On the positive side, we proposed the first polynomial time method to eliminate *all* row and column symmetry when the number of rows (or columns) is bounded. On the negative side, we argued that DOUBLELEX and SNAKELEX can leave a large number of symmetric solutions. In addition, we proved that propagating DOUBLELEX completely is NP-hard. Finally, we showed that it is not always safe to combine Puget's value symmetry breaking constraints with row and column symmetry breaking constraints, correcting a claim made in the literature.

References

1. Flener, P., Frisch, A., Hnich, B., Kiziltan, Z., Miguel, I., Pearson, J., Walsh, T.: Breaking row and column symmetry in matrix models. In: 8th International Conference on Principles and Practices of Constraint Programming (CP-2002), Springer (2002)

2. Flener, P., Frisch, A., Hnich, B., Kiziltan, Z., Miguel, I., Walsh, T.: Matrix Modelling. Technical Report APES-36-2001, APES group (2001) Presented at Formul'01 (Workshop on Modelling and Problem Formulation), CP2001 post-conference workshop.
3. Crawford, J., Ginsberg, M., Luks, G., Roy, A.: Symmetry breaking predicates for search problems. In: Proceedings of 5th International Conference on Knowledge Representation and Reasoning, (KR '96). (1996) 148–159
4. Huczynska, S., McKay, P., Miguel, I., Nightingale, P.: Modelling equidistant frequency permutation arrays: An application of constraints to mathematics. In Gent, I., ed.: Principles and Practice of Constraint Programming - CP 2009, 15th International Conference, CP 2009, Lisbon, Portugal, September 20-24, (2009) 50–64
5. Bessiere, C., Hebrard, E., Hnich, B., Walsh, T.: The complexity of global constraints. In: Proceedings of the 19th National Conference on AI, AAAI (2004)
6. Bessiere, C., Hebrard, E., Hnich, B., Walsh, T.: The complexity of global constraints. Constraints, **12(2)** (2007) 239-259
7. Carlsson, M., Beldiceanu, N.: Arc-consistency for a chain of lexicographic ordering constraints. Technical report T2002-18, Swedish Institute of Computer Science (2002).
8. Katsirelos, G., Narodytska, N., Walsh, T.: Breaking Generator Symmetry In: Proceedings of SymCon'09 - 9th International Workshop on Symmetry and Constraint Satisfaction Problems, colocated with CP2009.
9. Puget, J.F.: Breaking symmetries in all different problems. In: Proceedings of 19th IJCAI, International Joint Conference on Artificial Intelligence (2005) 272–277
10. Shlyakhter, I.: Generating effective symmetry-breaking predicates for search problems. Electronic Notes in Discrete Mathematics **9** (2001) 19–35
11. Flener, P., Frisch, A., Hnich, B., Kiziltan, Z., Miguel, I., Pearson, J., Walsh, T.: Symmetry in matrix models. Technical Report APES-30-2001, APES group (2001) Presented at SymCon'01 (Symmetry in Constraints), CP2001 post-conference workshop.
12. Walsh, T.: Breaking Value Symmetry. In: Proceedings of 13th International Conference on Principles and Practice of Constraint Programming (CP2007), Springer (2007)
13. Puget, J.F.: Breaking all value symmetries in surjection problems. In van Beek, P., ed.: Proceedings of 11th International Conference on Principles and Practice of Constraint Programming (CP2005), Springer (2005)
14. Law, Y., Lee, J.: Global constraints for integer and set value precedence. In: Proceedings of 10th International Conference on Principles and Practice of Constraint Programming (CP2004), Springer (2004) 362–376
15. Walsh, T.: Symmetry breaking using value precedence. In Brewka, G., Coradeschi, S., Perini, A., Traverso, P., eds.: ECAI 2006, IOS Press (2006) 168–172
16. Grayland, A., Miguel, I., Roney-Dougal, C.: Snake lex: An alternative to double lex. In Gent, I.P., ed.: Proceedings of 15th International Conference on Principles and Practice of Constraint Programming. Springer (2009) 391–399
17. Hnich, B., Prestwich, S., Selensky, E., Smith, B.: Constraint models for the covering test problem. Constraints **11** (2006) 199–219
18. Lubiw, A.: Doubly lexical orderings of matrices. SIAM J. on Computing **16** (1987) 854–879
19. Frisch, A., Jefferson, C., Miguel, I.: Constraints for breaking more row and column symmetries. In Rossi, F., ed.: Proceedings of 9th International Conference on Principles and Practice of Constraint Programming (CP2003), Springer (2003)
20. Frisch, A., Hnich, B., Kiziltan, Z., Miguel, I., Walsh, T.: Multiset ordering constraints. In: Proceedings of 18th IJCAI, International Joint Conference on Artificial Intelligence (2003)
21. Gent, I., Smith, B.: Symmetry breaking in constraint programming. In Horn, W., ed.: Proceedings of ECAI-2000, IOS Press (2000) 599–603