Breaking All the Symmetries in Matrix Models: Results, Conjectures, and Directions

Pierre Flener and Justin Pearson

Department of Information Technology, Uppsala University
Box 337, 751 05 Uppsala, Sweden
PierreF@csd.uu.se, justin@docs.uu.se

1 Introduction: Matrix Models and Symmetry

A *matrix model* is a constraint program that contains one or more matrices of decision variables. For example, a natural model of the round robin tournament scheduling problem (prob026 in CSPlib [4]) has a 2-dimensional (2-d) matrix of variables, each of which is assigned a value corresponding to the match played in a given week and period [15]. In this case, the matrix is obvious in the modelling of the problem: we need a *table* of fixtures. However, many other problems that are less obviously defined in terms of matrices of variables can be effectively represented and efficiently solved using matrix models [7]. In this paper, we focus on matrix models with just one matrix of decision variables.

Symmetry is an important aspect of matrix models. A *symmetry* is a bijection on decision variables that preserves solutions and non-solutions. Two variables are *indistinguishable* if some symmetry interchanges their rôles in all solutions and non-solutions. Symmetry often occurs because groups of objects within a matrix are indistinguishable. For example, in the round robin tournament scheduling problem, weeks and periods are indistinguishable. We can therefore permute any two weeks or any two periods in the schedule. That is, we can permute any two rows or any two columns of the associated matrix, whose index sets are the weeks and periods.

A *row* (*column*) *symmetry* of a 2-d matrix is a bijection between the variables of two of its rows (columns) that preserves solutions and non-solutions. Two rows (columns) are *indistinguishable* if their variables are pairwise indistinguishable due to a row (column) symmetry. The rotational symmetries of a matrix are neither row nor column symmetries. A matrix model *has total row* (*column*) *symmetry* iff all the rows (columns) of its matrix are indistinguishable. These definitions can be extended to matrices with any number of dimensions. A *symmetry class* is an equivalence class of assignments, where two assignments are *equivalent* if there is some symmetry mapping one assignment into the other. In group theory, such equivalence classes are referred to as *orbits*.

2 Breaking Symmetry

There are a number of ways of dealing with symmetry in constraint programming. We here only consider techniques that remove symmetries with *symmetry-*

breaking constraints by adding them to the model *before* search starts, e.g., [14, 3]. Other techniques break symmetries by adding constraints *during* search, e.g., [1], the symmetry-breaking during search framework (SBDS) [11], and the global cut framework (GCF) [8].

A common method to break symmetry is to impose a constraint that orders the symmetric objects. To break all the row (column) symmetries, one can order the rows (columns) lexicographically. The rows (columns) in a 2-d matrix are *lexicographically ordered* if each row (column) is lexicographically smaller (denoted $\leq_{lex}$) than the next, if any. Each orbit has a matrix where the rows and columns are lexicographically ordered [6]. However, lexicographically ordering the rows and columns does *not* break all the compositions of the row and column symmetries. Consider a 3×3 matrix of $0/1$ variables, x_{ij}, such that $\forall i\ x_{i1} + x_{i2} + x_{i3} \geq 1$ and $\sum_{ij} x_{ij} = 4$. This model has total row and column symmetry. The following solutions have lexicographically ordered rows and columns:

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

These solutions are symmetric, as one can move from one to the other by swapping the first two rows and the last two columns. Swapping any rows or columns *individually* breaks the lexicographic ordering.

3 Breaking All the Symmetries

It is always possible to break all the symmetries. We now illustrate a method by an example and discuss its application.

3.1 A Method

A *group*, $G = (X, \circ)$, is a set of elements, X, where $\circ$ is an associative binary operation such that there exists exactly one element $id \in X$ having the property that for all elements $x \in X$, we have that $id \circ x = x \circ id = x$, and further for every element $x \in X$, there exists a unique element x^{-1} such that $x \circ x^{-1} = x^{-1} \circ x = id$. Given a set of elements, Ω, we will be interested in groups whose elements are bijective functions on Ω, that is *permutations*. The operation $\circ$, in such groups, will be function composition. If such a set of permutations is closed under taking inverse functions, then the set is a group. We will write permutations on finite sets as products of cycles. A *cycle* $(x, f(x), f^2(x), \ldots, f^n(x))$ gives the action of the permutation f on x; the last item in the cycle $f^n(x)$ asserts that $f^{n+1}(x) = x$. Modulo the order of the cycles and the starting point in each cycle, a permutation on a finite set has a unique cyclic decomposition. When writing permutations, unit cycles are often omitted. For example, a function f on the integers $1 \ldots 6$ such that $f(1) = 2$, $f(2) = 3$, $f(3) = 1$, $f(4) = 5$, $f(5) = 4$, and $f(6) = 6$ would be denoted $(1, 2, 3)(4, 5)$. Given an element, x, of a group, the *order* of x is the

smallest integer n such that $x^n = id$. For example, the order of $(1,2,3)(4,5)$ is 6. The group $\mathrm{Sym}(q)$ is the set of all permutations on the set $\{1,\ldots,q\}$.

In [3], a method of breaking symmetry by adding constraints is introduced. Essentially, for each symmetry of the problem a lexicographic ordering constraint is added, forcing only one of the symmetric solutions to be picked.

We now illustrate all this on a running example that has a maybe small matrix but is sufficiently illustrative.

Example 1. The group of all the row and column symmetries of a 3×2 matrix

$$\begin{pmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \end{pmatrix}$$

can be generated by the following 3 permutations, permuting the first two columns, the last two columns, and the two rows, respectively:

$$(1,2)(4,5) \qquad (2,3)(5,6) \qquad (1,4)(2,5)(3,6)$$

This group contains the following 12 permutations:

Permutation	Name	Order
()	id	1
(1,2)(4,5)	$P_{c_{12}}$	2
(2,3)(5,6)	$P_{c_{23}}$	2
(1,4)(2,5)(3,6)	$P_{r_{12}}$	2
(1,6,2,4,3,5)	P_δ	6
(1,5,3,4,2,6)	P_σ	6
(1,4)(2,6)(3,5)	P_{α_1}	2
(1,5)(2,4)(3,6)	P_{α_2}	2
(1,6)(2,5)(3,4)	P_{α_3}	2
(1,3)(4,6)	$P_{c_{13}}$	2
(1,2,3)(4,5,6)	$P_{c_{123}}$	3
(1,3,2)(4,6,5)	$P_{c_{132}}$	3

Omitting the identity permutation (), these symmetries can be broken by the following 11 constraints:

$$[x_1,x_2,x_3,x_4,x_5,x_6] \leq_{lex} [x_2,x_1,x_3,x_5,x_4,x_6] \qquad (c_{12})$$
$$[x_1,x_2,x_3,x_4,x_5,x_6] \leq_{lex} [x_1,x_3,x_2,x_4,x_6,x_5] \qquad (c_{23})$$
$$[x_1,x_2,x_3,x_4,x_5,x_6] \leq_{lex} [x_4,x_5,x_6,x_1,x_2,x_3] \qquad (r_{12})$$
$$[x_1,x_2,x_3,x_4,x_5,x_6] \leq_{lex} [x_6,x_4,x_5,x_3,x_1,x_2] \qquad (\delta)$$
$$[x_1,x_2,x_3,x_4,x_5,x_6] \leq_{lex} [x_5,x_6,x_4,x_2,x_3,x_1] \qquad (\sigma)$$
$$[x_1,x_2,x_3,x_4,x_5,x_6] \leq_{lex} [x_4,x_6,x_5,x_1,x_3,x_2] \qquad (\alpha_1)$$
$$[x_1,x_2,x_3,x_4,x_5,x_6] \leq_{lex} [x_5,x_4,x_6,x_2,x_1,x_3] \qquad (\alpha_2)$$
$$[x_1,x_2,x_3,x_4,x_5,x_6] \leq_{lex} [x_6,x_5,x_4,x_3,x_2,x_1] \qquad (\alpha_3)$$
$$[x_1,x_2,x_3,x_4,x_5,x_6] \leq_{lex} [x_3,x_2,x_1,x_6,x_5,x_4] \qquad (c_{13})$$
$$[x_1,x_2,x_3,x_4,x_5,x_6] \leq_{lex} [x_2,x_3,x_1,x_5,x_6,x_4] \qquad (c_{123})$$
$$[x_1,x_2,x_3,x_4,x_5,x_6] \leq_{lex} [x_3,x_1,x_2,x_6,x_4,x_5] \qquad (c_{132})$$

Due to the meaning of the lexicographic ordering where:

$$[x_1, x_2, x_3, \ldots] \leq_{lex} [y_1, y_2, y_3, \ldots]$$

is defined to be:

$$(x_1 \leq y_1) \wedge \ (x_1 = y_1 \rightarrow x_2 \leq y_2) \wedge \ (x_1 = y_1 \wedge x_2 = y_2 \rightarrow x_3 \leq y_3) \wedge \ \ldots \quad (1)$$

and due to the right-hand vector here always being a permutation of the left-hand one, the elements at the positions corresponding to the last indices in each cycle (including the unit cycles) can be deleted in both vectors. By this token, the constraints above can be *internally* simplified to the following:

$$\begin{aligned}
[x_1, x_4] &\leq_{lex} [x_2, x_5] && (c_{12}) \\
[x_2, x_5] &\leq_{lex} [x_3, x_6] && (c_{23}) \\
[x_1, x_2, x_3] &\leq_{lex} [x_4, x_5, x_6] && (r_{12}) \\
[x_1, x_2, x_3, x_4, x_5] &\leq_{lex} [x_6, x_4, x_5, x_3, x_1] && (\delta) \\
[x_1, x_2, x_3, x_4, x_5] &\leq_{lex} [x_5, x_6, x_4, x_2, x_3] && (\sigma) \\
[x_1, x_2, x_3] &\leq_{lex} [x_4, x_6, x_5] && (\alpha_1) \\
[x_1, x_2, x_3] &\leq_{lex} [x_5, x_4, x_6] && (\alpha_2) \\
[x_1, x_2, x_3] &\leq_{lex} [x_6, x_5, x_4] && (\alpha_3) \\
[x_1, x_4] &\leq_{lex} [x_3, x_6] && (c_{13}) \\
[x_1, x_2, x_4, x_5] &\leq_{lex} [x_2, x_3, x_5, x_6] && (c_{123}) \\
[x_1, x_2, x_4, x_5] &\leq_{lex} [x_3, x_1, x_6, x_4] && (c_{132})
\end{aligned}$$

The first two constraints now indeed reflect the indistinguishability of the first two columns and the last two columns, respectively, whereas the third constraint reflects the indistinguishability of the two rows. The last three constraints are actually (logically) implied and can be eliminated. Indeed, the constraint c_{13} arises from the indistinguishability of the first and third columns, and is a logical consequence of the constraints c_{12} and c_{23}, due to the transitivity of $\leq_{lex}$. Also, the constraints c_{123} and c_{132} are logical consequences of the same constraints for the same reason, as the corresponding permutations of all the three columns can also be ruled out by requiring the entire columns to be (lexicographically) ordered from left to right. Furthermore, Frisch and Harvey [9] have manually established that the constraints δ and σ can be further simplified *in the context of the others*. All this leaves us now with the following 8 constraints:

$$\begin{aligned}
[x_1, x_4] &\leq_{lex} [x_2, x_5] && (c_{12}) \\
[x_2, x_5] &\leq_{lex} [x_3, x_6] && (c_{23}) \\
[x_1, x_2, x_3] &\leq_{lex} [x_4, x_5, x_6] && (r_{12}) \\
[x_1, x_2, x_3] &\leq_{lex} [x_6, x_4, x_5] && (\delta)
\end{aligned}$$

$$[x_1, x_2, x_3, x_4] \leq_{lex} [x_5, x_6, x_4, x_2] \qquad (\sigma)$$
$$[x_1, x_2, x_3] \leq_{lex} [x_4, x_6, x_5] \qquad (\alpha_1)$$
$$[x_1, x_2, x_3] \leq_{lex} [x_5, x_4, x_6] \qquad (\alpha_2)$$
$$[x_1, x_2, x_3] \leq_{lex} [x_6, x_5, x_4] \qquad (\alpha_3)$$

The last 6 constraints now (at least) require the first row to be lexicographically smaller than any permutation of the second row. We do not know whether this is a coincidence. The constraint σ cannot be further simplified [9].

In general, an $m \times n$ matrix has $m! \cdot n! - 1$ compositions of row and column symmetries except identity, generating thus a super-exponential number of $\leq_{lex}$ constraints. Hence this approach seems not always practical, even after eliminating the $m! - m + n! - n$ constraints that are implied due to the transitivity of $\leq_{lex}$. Despite encouraging experimental results [6] with just the constraints induced by the generator symmetries and in the presence of actual problem constraints, we are looking for special cases where all or most compositions of the row and column symmetries can be broken by a polynomial (and even linear) number of $\leq_{lex}$ constraints (see [6] for other results).

3.2 Implied Constraints

There may be further implied $\leq_{lex}$ constraints than we can predict so far based on the transitivity of $\leq_{lex}$, but we do not know whether the minimum set of non-implied $\leq_{lex}$ constraints is of polynomial size.

A *support set* of an implied constraint C is a set of constraints $\{c_1, \cdots, c_n\}$ not containing C such that $c_1 \wedge \ldots \wedge c_n \rightarrow C$. We here look for support sets for $\leq_{lex}$ constraints among the other $\leq_{lex}$ constraints. *Detecting* a support set of an implied constraint is not easy. Various methods can be adapted, such as circuit minimisation or integer linear programming (ILP). Indeed, $\leq_{lex}$ constraints can be encoded as linear inequalities. For instance, the constraint σ is equivalent to:

$$125 \cdot x_1 + 25 \cdot x_2 + 5 \cdot x_3 + x_4 \leq 125 \cdot x_5 + 25 \cdot x_6 + 5 \cdot x_4 + x_2 \qquad (2)$$

when the domain size is 5. However, the standard syntactic detection criteria from ILP listed in [12] fail. We have yet to try the new syntactic criteria in [12]. *Eliminating* implied constraints is also difficult, as there are usually many support sets. Furthermore, support sets may overlap.

3.3 Domain-Dependent Implied Constraints

An interesting observation is that the number of implied $\leq_{lex}$ constraints grows as the domain size of the decision variables shrinks. For instance, in Example 1, the five constraints δ, σ, α_1, α_2, and α_3 are implied when the domain size is 2, and up to four of them can be collectively eliminated, say all except δ or α_1. If the domain size is 3, then σ is no longer implied, and up to three of the other four can be collectively eliminated, say all except δ. If the domain size is 4, then none of these constraints is implied.

Experimentally, up to 6×6 matrices, we tested the following conjecture:

Conjecture 1. For a domain of size 2, it suffices to add the $\leq_{lex}$ constraints induced by the order 2 permutations.

For domain size 3, this conjecture is not true since, as just seen, the constraint σ is necessary, but is of order 6.

Unfortunately, we already determined even the number of order 2 permutations to be super-polynomial. Indeed, for an $m \times n$ matrix, it is $f(m) \cdot f(n) - 1$, where f is the sequence:

$$1, 1, 2, 4, 10, 26, 76, ...$$

This is sequence A000085 in the *On-Line Encyclopedia of Integer Sequences* [13] and has the following recurrence relation:

$$\begin{aligned} f(0) &= f(1) = 1 \\ f(n) &= f(n-1) + (n-1) \cdot f(n-2) \qquad \text{for } n > 1 \end{aligned}$$

and the following closed form:

$$f(n) = \sum_{k=0}^{n/2} \frac{n!}{(n - 2 \cdot k)! \cdot 2^k \cdot k!}$$

The value $f(q)$ is the number of order 2 permutations in the group $\mathrm{Sym}(q)$. Since the group of row and column symmetries on an $m \times n$ matrix is isomorphic to the product of $\mathrm{Sym}(m)$ and $\mathrm{Sym}(n)$, this justifies the formula $f(n) \cdot f(m) - 1$, as identity is counted twice.

Even if the conjecture is true, there still remain implied $\leq_{lex}$ constraints induced by the order 2 permutations. For example, c_{13} is induced by an order 2 permutation but is implied, as seen above. It is not known whether eliminating implied $\leq_{lex}$ constraints induced by the order 2 permutations leaves a polynomial number of $\leq_{lex}$ constraints.

It would be interesting to characterise which $\leq_{lex}$ constraints are necessary for which domain sizes.

3.4 Other Directions

As Frisch and Harvey [9] have shown, there are *contextual* simplifications to the $\leq_{lex}$ constraints that we cannot mechanise yet.

Another (by us yet unexplored) direction is to experimentally determine a polynomial number of $\leq_{lex}$ constraints that break "most" of the symmetries, possibly taking into account the effect of the problem constraints. Indeed, our experiments (with small matrices, up to 4×4) show that on the pure problem (enumerating all matrices modulo total row and column symmetry, though in the absence of any actual problem constraints), many of the super-polynomial number of $\leq_{lex}$ constraints just eliminate a handful of solutions, whereas a few of them eliminate hundreds or thousands of solutions. A characterisation of the more effective $\leq_{lex}$ constraints will be a useful achievement.

		with all the 35 constraints	without 15 implied constraints before internal simplifications	after internal simplifications
domain size = 4 (8,240 matrices)	Boolean $\leq_{lex}$	11.0"	5.8"	2.1"
	linear $\leq_{lex}$	8.3"	4.5"	1.6"
domain size = 5 (57,675 matrices)	Boolean $\leq_{lex}$	61.0"	31.8"	12.4"
	linear $\leq_{lex}$	49.6"	26.7"	10.0"
domain size = 6 (289,716 matrices)	Boolean $\leq_{lex}$	269.0"	139.0"	56.1"
	linear $\leq_{lex}$	227.0"	122.6"	46.5"

Table 1. The effects of constraint eliminations and internal simplifications

3.5 Experiments

We experimented, under GNU Prolog on a Sun SPARC Ultra station 10, with two different implementations of $\leq_{lex}$, namely a Boolean one based on (1) and an ILP one using linear inequalities as in (2). The objective was to enumerate all 3×3 matrices modulo total row and column symmetry, though in the absence of any actual problem constraints. Transitivity of $\leq_{lex}$ gives 6 implied constraints among 35 (irrespective of the domain size), and we experimentally detected another 9 implied constraints (for domain sizes from 4 up to at least 6), which can even be eliminated together as they have non-overlapping support sets within the remaining constraints.[1] Table 1 summarises the experiments, giving run times in seconds. Whichever the implementation of $\leq_{lex}$, both the constraint eliminations and then the internal simplifications pay off, together giving a five-fold time reduction for solving, irrespective of the domain size.

4 Conclusions

We have investigated some special cases where symmetry breaking constraints are implied or can be internally simplified. The constraints that can be removed due to implication depend on the domain size of the matrix problem. In some cases, these implied constraints actually slow down the constraint solver.

All approaches would benefit from an efficient means of automatic symmetry detection. However, symmetry detection has been shown to be graph-isomorphism complete in the general case [2]. Therefore, it is often assumed that the symmetries are *declared* by the user. However, symmetries are often *introduced* while formulating CSPs in today's rather low-level constraint programming languages. Indeed, the latter usually lack high-level data structures, such as sets and relations, where element order is irrelevant, but whose lower-level

[1] Surprisingly, even the constraint ordering the first two rows is implied for domain sizes up to at least 6, but we did not eliminate it in our experiments, as 8 of the other 9 implied constraints may well be implied irrespective of the domain size and even look as if more general results could eliminate them.

implementations in terms of lists or matrices make the element order seemingly relevant, so that symmetry-breaking constraints become necessary to compensate for this. The compiler of a relational constraint modelling language, such as the proposal in [5], should thus be aware of the symmetries it introduces.

Also, there is a need for efficient methods for establishing generalised arc consistency on sets of $\leq_{lex}$ constraints operating on the same matrix.

Acknowledgements. This work is partially supported by grant 221-99-369 of VR (the Swedish Research Council) and by institutional grant IG2001-67 of STINT (the Swedish Foundation for International Cooperation in Research and Higher Education). We also thank Alan M. Frisch, Warwick Harvey, the referees, as well as the members of the APES and ASTRA research groups, for their helpful discussions.

References

1. R. Backofen and S. Will. Excluding symmetries in constraint-based search. In *Proc. of CP'99*, J. Jaffar (ed), LNCS 1713, pp. 73–87. Springer-Verlag, 1999.
2. J. Crawford. A theoretical analysis of reasoning by symmetry in first-order logic. In *Proc. of AAAI'92 workshop on tractable reasoning*, 1992.
3. J. Crawford, G. Luks, M. Ginsberg, and A. Roy. Symmetry breaking predicates for search problems. In *Proc. of KR'96*, pp. 148–159, 1996.
4. *CSPlib, a Problem Library for Constraints*, at www.csplib.org.
5. P. Flener. Towards relational modelling of combinatorial optimisation problems. In: Ch. Bessière (ed), *Proc. of the IJCAI'01 Workshop on Modelling and Solving Problems with Constraints*, at www.lirmm.fr/~bessiere/nbc_workshop.htm, 2001.
6. P. Flener, A.M. Frisch, B. Hnich, Z. Kızıltan, I. Miguel, J. Pearson, and T. Walsh. Symmetry in matrix models. In *Proc. of SymCon'01*, at www.csd.uu.se/~pierref/astra/SymCon01/, 2001. Extended paper in P. Van Hentenryck (ed), *Proc. of CP'02*, LNCS 2470, Springer-Verlag, 2002.
7. P. Flener, A.M. Frisch, B. Hnich, Z. Kızıltan, I. Miguel, and T. Walsh. Matrix modelling. In *Proc. of Formul'01*, at www.dcs.gla.ac.uk/~pat/cp2001/, 2001.
8. F. Focacci and M. Milano. Global cut framework for removing symmetries. In *Proc. of CP'01*, T. Walsh (ed), LNCS 2239, pp. 77–92. Springer-Verlag, 2001.
9. A.M. Frisch. Slides for the SymCon'01 presentation of [6], at www.cs.york.ac.uk/aig/projects/implied/docs/SymMxCP01.ppt, 2001.
10. *The GAP Group – Groups, Algorithms, and Programming*, at www.gap-system.org.
11. I.P. Gent and B.M. Smith. Symmetry breaking in constraint programming. In *Proc. of ECAI'00*, W. Horn (ed), pp. 599–603. IOS Press, 2000.
12. J.-L. Imbert and P. Van Hentenryck. Redundancy elimination with a lexicographic solved form. In *Annals of Mathematics and AI*, 17(1–2):85–106, 1996.
13. *On-Line Encyclopedia of Integer Sequences*, at www.research.att.com/~njas/ sequences/.
14. J.-F. Puget. On the satisfiability of symmetrical constrained satisfaction problems. In *Proc. of ISMIS'93*, LNAI 689, pp. 350–361. Springer-Verlag, 1993.
15. P. Van Hentenryck, L. Michel, L. Perron, and J.-C. Régin. Constraint programming in OPL. In *Proc. of PPDP'99*, LNCS 1703, pp. 97–116. Springer-Verlag, 1999.