DoubleLex Revisited and Beyond

Xuming Huang and **Jimmy Lee**
Department of Computer Science and Engineering, The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
{xmhuang,jlee}@cse.cuhk.edu.hk

Abstract

The paper proposes Maximum Residue (MR) as a notion to evaluate the strength of a symmetry breaking method. We give a proof to improve the best known DoubleLex MR upper bound from $m!n! - (m! + n!)$ to $\min(m!, n!)$ for an $m \times n$ matrix model. Our result implies that DoubleLex works well on matrix models where $\min(m, n)$ is relatively small. We further study the MR bounds of SwapNext and SwapAny, which are extensions to DoubleLex breaking further a small number of composition symmetries. Such theoretical comparisons suggest general principles on selecting Lex-based symmetry breaking methods based on the dimensions of the matrix models. Our experiments confirm the theoretical predictions as well as efficiency of these methods.

1 Introduction

Many constraint satisfaction problems can be formulated as a matrix model [Flener *et al.*, 2001], which has the decision variables are organized in the form of a matrix with rows and columns. Some such problems consist of matrix symmetries [Flener *et al.*, 2002], in which rows and columns in a solution matrix can be swapped arbitrarily to form symmetrically equivalent solutions. DoubleLex [Flener *et al.*, 2002] is an efficient method for breaking matrix symmetries by posting a linear number of lexicographical ordering constraints. While Flener et al.[2002] demonstrate that DoubleLex can eliminate all row and column symmetries, its observed good performance [Katsirelos *et al.*, 2010] in practice suggests a stronger theoretical pruning guarantee.

In this paper, we propose the notion of Maximum Residue (MR), which is the maximum number of remaining solutions over all symmetry classes of a symmetry group, as a measure of the strength of a symmetry breaking method. A major result is a factorial improvement of the DoubleLex's MR upper bound from $m!n! - (m! + n!)$ to $\min(m!, n!)$ for an $m \times n$ matrix model. We further show that this bound is tight in general. Our result implies that DoubleLex works well when $\min(m, n)$ is relatively small. When $\min(m, n)$ is large, our analysis suggests the need for more symmetry breaking constraints over those imposed by DoubleLex. Furthermore, we study the MR bounds of SwapNext and SwapAny[Smith, 2014], which are two slight extensions upon DoubleLex by further breaking products (compositions) of a row and a column symmetries.

Obviously, SwapNext and SwapAny subsume DoubleLex in symmetry breaking power but with similar complexity in terms of MR lower bounds. Our theoretical understanding of these three methods suggests guiding principles on the efficiency of the methods when applied on matrix models of varying sizes as defined by the matrices' dimensions. Experimental results confirm (a) the accuracy of our theoretical predictions and (b) the overhead of extra symmetry breaking constraints in SwapNext and SwapAny can be nicely compensated in practice in certain scenarios.

2 Background

A *constraint satisfaction problem* (CSP) is a triple $P = (X, D, C)$ where $X = \{x_1, \ldots, x_n\}$ is a set of *variables*, each of which takes value from its *domain* $D(x_i)$, and C is a set of *constraints* each specifying allowed value combinations over a set of variables. An *assignment* $\{x_i = v_i | i \in \{1, \ldots, n\}\}$ instantiates each variable x_i with $v_i \in D(x_i)$. A *solution* θ of P is an assignment that satisfies all constraints in C. We denote the set of all solutions of P as $sol(P)$.

A *symmetry* is a transformation from assignments to assignments and maps a solution to a solution. A *symmetry group* G_Σ is generated by a set of symmetry generators Σ under composition. We use id to denote the identity in G_Σ. A *symmetry class* S of a group G is a set of solutions that are symmetrically equivalent: a solution $s \in S$ if and only if $g(s) \in S$ for every $g \in G$. Equivalently speaking, if a solution $s \in S$, then $S = \{g(s) | g \in G\}$. The symmetry group of an $m \times n$ matrix model can be generated by a set of row symmetry generators $\{r_i | i \in \{1, \ldots, m-1\}\}$ and a set of column symmetry generators $\{c_j | j \in \{1, \ldots, n-1\}\}$, where r_i switches the i-th and the (i+1)-st rows and c_j switches the j-th and the (j+1)-st columns. When the context is clear, we use G_{row} to denote the group $G_{\{r_1, \ldots, r_{m-1}\}}$, G_{col} for $G_{\{c_1, \ldots, c_{n-1}\}}$, and G_{mat} for the matrix symmetry group.

A symmetry breaking constraint removes symmetries, and therefore also symmetric solutions in a CSP. A set of symmetry breaking constraints C^{sb} is *sound* with respect to G if at least one solution in each symmetry class of G in $sol(P)$ remains in $sol(P, C^{sb}) = sol(P')$, where

	id	...	$\boldsymbol{c_{p_1}}$	...	$\boldsymbol{c_{p_2}}$	...
id	θ	...	...	...	...	...
$\vdots$		$\vdots$		$\vdots$		
r_{p_t}	...	...	$r_{p_t}\boldsymbol{c_{p_1}}(\theta)$	...	$r_{p_t}\ \boldsymbol{c_{p_2}}(\theta)$	...
$\vdots$		$\vdots$		$\vdots$		

Table 1: The symmetry class of θ

$P' = (X, D, C \cup C^{sb})$. A set of symmetry breaking constraints C^{sb} is *complete* with respect to G if *exactly* one solution in each symmetry class of G remains in $sol(P, C^{sb})$.

3 DoubleLex Revisited

DoubleLex [Flener *et al.*, 2002] posts $(m+n-2)$ constraints requiring a solution to be both row-wise and column-wise lexicographically ordered. We concatenate the rows of a matrix from the top to the bottom to form a string for lexicographic comparison. More specifically, the following sets of constraints are posted for an $m \times n$ matrix model:

$$\theta \leq_{lex} r_i(\theta) \qquad \forall i \in \{1, ..., (m-1)\} \quad (1)$$
$$\theta \leq_{lex} c_j(\theta) \qquad \forall j \in \{1, ..., (n-1)\} \quad (2)$$

where $r_i's$ are row symmetry generators that swap the i-th row and $(i+1)$-st row, and $c_j's$ are column symmetry generators that swap the j-th column and $(j+1)$-st column.

3.1 Improved Bound of Remaining Solutions

Flener et al.[2002] prove that all row symmetries (of size $m!$) and column symmetries (of size $n!$) are removed, thus potentially leaving $m!n! - (m! + n!)$ solutions in each symmetry class. However, the method works surprisingly well in practise and leaves much fewer solutions. In the following, we provide further insight to the good performance by improving the best known upper bound to $min(m!, n!)$.

Theorem 1. *At most $min(m!, n!)$ distinct solutions satisfy DoubleLex in any symmetry class of G_{mat} for an $m \times n$ matrix.*

Proof. Given a solution θ of a symmetry class, the symmetry class of θ can be generated by applying each symmetry in G_{mat} upon θ. So the symmetry class of θ is $\{g(\theta)|g \in G\}$. We construct a table, where the rows are indexed by row symmetries from G_{row} and the columns are indexed by column symmetries from G_{col}. Since composition of a row symmetry and a column symmetry is commutative, each symmetry $g \in G_{mat}$ can be uniquely written as the product of a row symmetry and column symmetry: $g = r_{p_i}c_{p_j}$ where $r_{p_i} \in G_{row}$ and $c_{p_j} \in G_{col}$ (note that r_{p_i} is not necessarily a generator, and similarly for c_{p_j}). We place the solution $g(\theta) = r_{p_i}c_{p_j}(\theta)$ at the cell on the row indexed by r_{p_i} and the column indexed by c_{p_j} thus the table encodes exactly the symmetry class of θ (shown in Table 1).

Among the solutions on the same row of the table, at most one distinct solution satisfies DoubleLex. Assume the contrary and consider the following two distinct solutions on the same row indexed by r_{p_t} that both satisfy DoubleLex:

$$g_1(\theta) = r_{p_t}c_{p_1}(\theta) \text{ and } g_2(\theta) = r_{p_t}c_{p_2}(\theta)$$

where $c_{p_1} \neq c_{p_2}$. Observe that by applying a column symmetry $c_{p_2}c_{p_1}^{-1}$ on $g_1(\theta)$, we get $g_2(\theta)$:

$$\begin{aligned} c_{p_2}c_{p_1}^{-1}g_1(\theta) &= c_{p_2}c_{p_1}^{-1}r_{p_t}c_{p_1}(\theta) = c_{p_2}c_{p_1}^{-1}c_{p_1}r_{p_t}(\theta) \\ &= c_{p_2}r_{p_t}(\theta) = r_{p_t}c_{p_2}(\theta) = g_2(\theta) \end{aligned}$$

That means by only permuting the columns of a column-wise lexicographically ordered solution $g_1(\theta)$, the resulting columns are still lexicographically ordered. Lexicographic ordering is a total order and this leads to a contradiction if the two solutions are distinct. So $g_1(\theta)$ and $g_2(\theta)$ must be the same. In other words, there can only be at most one solution placed on each row of the table. Similarly, there can only be one distinct solution on each column of the table. We count the number of remaining solutions in the table either by rows or columns and thus there are at most $min(m!, n!)$ solutions in the table. □

The theorem give a factorial improvement to the best known upper bound of DoubleLex. The proof has no dependence upon problem constraints, and should apply to any problems with matrix symmetries. Moreover, it implies that the strong guarantee maintains as long as the ordering is total. We immediately have the following more general theorem on *any* total ordering used in symmetry breaking:

Theorem 2. *Given an $m \times n$ matrix model and a total order $\prec$ on strings. At most $min(m!, n!)$ distinct solutions are both rows and columns ordered with respect to $\prec$ in any symmetry class of G_{mat}.*

By the above theorem, we can order rows and columns using any total order for symmetry breaking to generate other methods enjoying the same theoretical guarantee as DoubleLex. Concretely, we refer to the constraints that order both rows and columns using *Reflex* [Lee and Zhu, 2016] and *Gray code Ordering* [Narodytska and Walsh, 2013] (for binary strings) as **DoubleReflex** and **DoubleGrayCode**.

Corollary 1. *At most $min(m!, n!)$ distinct solutions satisfy **DoubleReflex** in any symmetry class of G_{mat} for an $m \times n$ matrix.*

Corollary 2. *At most $min(m!, n!)$ distinct solutions satisfy **DoubleGrayCode** in any symmetry class of G_{mat} for an $m \times n$ 0/1 matrix.*

3.2 Maximum Residue

The strength of a symmetry breaking method is commonly evaluated by the number of remaining solutions. It shows an overall behaviour upon all symmetry classes. Less effort is made to study what really happened within each symmetry class. Indeed, the effects of a set of symmetry breaking constraints on different symmetry classes are different. In some symmetry class, some symmetries are removed yet they remains in another symmetry class. Zeynep [2004] pointed out it is difficult to theoretically study how many symmetries are removed. The above analysis gives a possible direction to

progress. We can directly characterize the number of remaining solutions in a symmetry class and consider it as an instructive metric to measure the power of a symmetry breaking method. We define the following notion to capture the power of a set of symmetry breaking constraints $\mathcal{C}$.

Definition 1. *Given an $m \times n$ unconstrained matrix where each variable has a domain $\{0, ..., (d-1)\}$ and a set of symmetry breaking constraints $\mathcal{C}$. We define the Maximum Residue as*

$$\textit{MR}(m, n, d, \mathcal{C}) \triangleq \max_{sc \in SC(m,n,d)} |\{\theta | \theta \in \textit{sc and } \theta \textit{ satisfies } \mathcal{C}\}|$$

where $SC(m, n, d)$ denotes all symmetry classes of an $m \times n$ unconstrained matrix with domains $\{0, ..., (d-1)\}$.

MR$(m, n, d, \mathcal{C})$ gives the maximum number of remaining distinct solutions satisfying $\mathcal{C}$ over all symmetry classes. Though characterizing only the worst solution class, MR gives a manageable analysis and serves as a supplement to existing measures of strengths of symmetry breaking constraints. A symmetry breaking method having a smaller number of maximum residue generally leaves fewer redundant symmetric solutions and results in a shorter search time. The maximum residue notion captures the symmetry breaking power in theory, and we will show how it agrees with practical performance in the experimental section.

We now give a property of MR.

Property 1. *MR*$(m, n, d, \mathcal{C}) \leq$ *MR*$(m, n, d', \mathcal{C})$ *when* $d \leq d'$

Proof. Obviously, $SC(m, n, d) \subseteq SC(m, n, d')$ when $d \leq d'$, And thus *MR*$(m, n, d, \mathcal{C}) \leq$ *MR*$(m, n, d', \mathcal{C})$. □

Without loss of generality, we denote the number of rows as m, the number of columns as n, and assume $m \leq n$ from now on to ease our discussion. We know from Theorem 1 that *MR*$(m, n, d,$ *DoubleLex*$) \leq m!$. We provide a construction showing that the bound is tight. Namely, we show a symmetry class of an $m \times m$ unconstrained matrix where there are exactly $m!$ solutions satisfying DoubleLex.

Construction 1 Construct an $m \times m$ matrix where all elements except those on counter diagonal are zero, and the counter diagonal is filled by any permutation of $\{1, 2, ..., m\}$. For example, we construct the following matrix for $m = 3$:

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 2 & 0 \\ 3 & 0 & 0 \end{pmatrix}$$

The matrix satisfies DoubleLex. In fact, any such matrix whose counter diagonal is filled with a permutation of $\{1, 2, 3\}$ satisfies DoubleLex. Obviously one can transform one such matrix to another by permuting the rows and columns, and thus they are in the same symmetry class. There are exactly $m!$ permutations and therefore we have a symmetry class with $m!$ remaining solutions. Obviously, we construct such matrix with more than m columns by appending zero columns to the construction.

Theorem 3. *MR*$(m, n, d,$ *DoubleLex*$) = m!$ *for* $n \geq m, d \geq m + 1$.

3.3 MR as Performance Indicator

MR$(m, n, d, \mathcal{C})$ captures the worst case behaviour of $\mathcal{C}$ and it can be used to indicate the empirical performance of $\mathcal{C}$. It is observed in [Flener *et al.*, 2002] that enforcing lexicographical ordering only between columns (CLex) greatly outperforms enforcing lexicographical ordering only between rows (RLex) on Balanced Incomplete Block Designs (BIBD) instances. The authors conjectured it to be related to the tight scalar product constraint on pairs of rows. We observe BIBD instances involve skew matrices whose number of columns is much larger than its number of rows. We believe this is also a culprit to the peformance difference since RLex has a significantly larger MR ($n!$) than CLex ($m!$).

A symmetry breaking method with a smaller MR is expected to perform better in practise. However, we also observe the performance difference between DoubleLex and CLex that have the same maximum residue. We believe this advantage comes from the common wisdom in symmetry breaking: Breaking an "appropriate" number of extra symmetry is beneficial as the time saving from avoiding exploration of symmetric solutions and failures compensates the extra propagation cost. McDonald and Smith [2002] mentioned that the advantage disappears when the size of symmetries used reach a certain point beyond which enforcing symmetry breaking is no more worthwhile. DoubleLex is closer to the optimum tradeoff point than CLex, but we believe breaking more symmetries over DoubleLex can further improve the performance in certain circumstances. We study the MR bounds of two extensions upon DoubleLex.

4 Beyond DoubleLex

We study SwapNext and SwapAny[Smith, 2014] which breaks extra composition symmetries that are products of a row and a column generators.

4.1 SwapNext and SwapAny

We use rowwise order to concatenate the rows of the matrix.

Definition 2. *SwapNext posts the following sets of constraints for an $m \times n$ matrix model:*

$$(1) + (2) + \\ \theta \leq_{lex} r_i c_j(\theta), \forall i \in \{1, ..., (m-1)\}, j \in \{1, ..., (n-1)\} \quad (3)$$

The second set of $(m-1)(n-1)$ constraints break a set of composition symmetries that are products of a row symmetry generator and a column symmetry generator.

SwapNext considers only products of a row and a column symmetries that swap adjacent rows/columns. SwapAny further extends the idea to breaking products of row symmetries and column symmetries that swap any pairs of rows and columns.

Definition 3. *SwapAny posts the following sets of constraints for an $m \times n$ matrix model:*

$$(1) + (2) + \\ \theta \leq_{lex} r_{i_1,i_2} c_{j_1,j_2}(\theta), \forall 1 \leq i_1 < i_2 \leq m, 1 \leq j_1 < j_2 \leq n \quad (4)$$

where r_{i_1,i_2} swaps the i_1-th row and i_2-th row, and c_{j_1,j_2} swaps the j_1-th column and j_2-th column.

The $\binom{m}{2}\binom{n}{2}$ constraints in (4) break a set of composition symmetries that are products of a row symmetry that swaps two rows and a column symmetry that swaps two columns.

SwapAny is stronger than SwapNext, and SwapNext is stronger than DoubleLex. We are interested in whether breaking these two sets of extra composition symmetries helps lower the maximum residue. We give bounds of maximum residue of SwapNext and SwapAny in the next section.

4.2 MR Bounds

The two methods naturally inherit the MR bound from DoubleLex, and we have *MR*$(m, n, d, \textit{SwapNext}) \leq m!$ and *MR*$(m, n, d, \textit{SwapAny}) \leq m!$. We are curious if smaller upper bounds exist but we found that $m!$ is in fact a tight upper bound for both methods when $d = m + 1$.

Theorem 4. *MR*$(m, n, m+1, \textit{SwapNext}) = m!$ *for* $n \geq 2m$.

Theorem 5. *MR*$(m, n, m + 1, \textit{SwapAny}) = m!$ *for* $n \geq 2m$.

Proof. We prove both theorems by constructing a symmetry class where we can identify $m!$ distinct solutions satisfying SwapAny (and therefore satisfying SwapNext too).

We construct a matrix by duplicating the columns of Construction 1 and order them lexicographically. For example, we have the following matrix for $m = 3$ (so the number of columns $n = 2m$):

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 2 & 2 & 0 & 0 \\ 3 & 3 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Obviously the matrix satisfies DoubleLex. We can easily check by computers that the matrix also satisfies SwapAny thus also SwapNext. The example shows a way to construct such matrix: The matrix should have multiples copies of columns with one non-zero entry. Columns with different non-zero entry should have their entries appear on different rows. We can create copies of m such distinct columns and pack them into an $m \times 2m$ matrix. There are $m!$ such distinct matrices and they all belong to the same symmetry class. we can construct the desired matrix with more than $2m$ columns by padding zero columns. □

Without restrictions on n and d, the worst cases of *MR*s of SwapNext and SwapAny equal that of DoubleLex. As many matrix problems involves $0/1$ matrix, it is also interesting to see if restricting domain size yields a better upper bound. Unfortunately, the bounds cannot be improved below $m!$.

Theorem 6. *MR*$(m, n, 2, \textit{SwapNext}) = m!$ *for* $n \geq \frac{m(m+3)}{2}$.

Theorem 7. *MR*$(m, n, 2, \textit{SwapAny}) = m!$ *for* $n \geq \frac{m(m+3)}{2}$.

Proof. We use the same idea as in the previous constructions by creating distinct columns. However, the domain is restricted to $\{0, 1\}$. We use sequences of $1's$ of different lengths to "represent" distinct values. We construct such a matrix for $m = 3$.

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

The matrix satisfies SwapAny. And similarly we can prove $m!$ solutions within this symmetry class satisfy SwapAny. The construction requires $2 + \ldots + (m + 1) = \frac{m(m+3)}{2}$ columns. Appending zero columns on the left can produce matrices of more than $\frac{m(m+3)}{2}$ columns. □

The flexibility provided by the two unrestricted parameters n and d enables the existence of worst cases where SwapNext and SwapAny hit the bound. The MRs of all three methods grow more than exponentially. We now study if the asymptotic behaviour can be improved when restricting n and d together. We study the case where $m = n$ and $d = 2$.

For DoubleLex, a construction is given in [Katsirelos *et al.*, 2010] showing a symmetry class of a $2m \times 2m$ binary matrix where $m!$ distinct solutions satisfy DoubleLex. The construction packs two identity matrices *I*, one matrix of zeroes *O* and an arbitrary permutation matrix P (all of size $m \times m$ as the subscripts show) together as follows.

$$\begin{pmatrix} O_m & I_m \\ I_m & P_m \end{pmatrix}$$

Equivalently speaking, there exists a symmetry class of an $m \times m$ binary matrix where $\lfloor \frac{m}{2} \rfloor!$ solutions satisfy DoubleLex. So we have *MR*$(m, m, 2, \textit{DoubleLex}) \geq \lfloor \frac{m}{2} \rfloor!$.

How about SwapNext and SwapAny? In fact, we can show their growth is at least exponential in $\sqrt{m}$.

Theorem 8. *MR*$(m, m, 2, \textit{SwapNext}) \geq (\lfloor \frac{m}{4} \rfloor)!$

Proof. For any m being a multiple of 4, we can construct a matrix by packing some identity matrices, zero matrices and an arbitrary permutation matrix P of appropriate sizes as follows.

$$\begin{pmatrix} O_{m/2} & I_{m/2} \\ I_{m/2} & X_{m/2} \end{pmatrix}$$

where

$$X_{m/2} = \begin{pmatrix} O_{m/4} & I_{m/4} \\ I_{m/4} & P_{m/4} \end{pmatrix}$$

It can be easily verified that the matrix satisfies SwapNext. We obtain other distinct solutions within the same symmetry class for different choice of the permutation matrix $P_{m/4}$ so there are at least $(\frac{m}{4})!$ solutions. When m is not a multiple of 4, we can construct such matrix for $k = (\lfloor \frac{m}{4} \rfloor) * 4$ and further append zero rows and columns to construct the matrix of size $m \times m$. To sum up, we can always construct a symmetry class of an $m \times m$ matrix where at least $(\lfloor \frac{m}{4} \rfloor)!$ solutions satisfy SwapNext. □

Theorem 9. *MR*$(m, m, 2, \textit{SwapAny}) \geq (\lfloor \sqrt{2m} - \frac{3}{2} \rfloor)!$

Proof. We can use the same construction technique in proving Theorem 7. We can construct a $k \times \frac{k(k+3)}{2}$ matrix as a core and append zero rows and columns to obtain an $m \times m$ matrix. As the core matrix resides in an $m \times m$ matrix, we can pick the largest k where $\frac{k(k+3)}{2} \leq m$ holds. Thus $k = \lfloor \frac{\sqrt{8m+9}-3}{2} \rfloor \geq (\lfloor \sqrt{2m} - \frac{3}{2} \rfloor)$. The symmetry class has

$k! = (\lfloor\sqrt{2m}-\frac{3}{2}\rfloor)!$ symmetric solutions satisfying SwapAny. Thus $MR(m, m, 2, SwapAny) \geq (\lfloor\sqrt{2m}-\frac{3}{2}\rfloor)!$. □

4.3 Discussion

We show the maximum residues of SwapNext and SwapAny grow at least exponentially in $\sqrt{m}$. However, the experiment results, which we will show in the next section, suggest that SwapNext and SwapAny potentially have a small upper bound. We conjecture tighter upper bounds of maximum residue.

SwapNext and SwapAny are two useful alternatives for scenarios that DoubleLex cannot handle well. DoubleLex works well when the smaller dimension m is particularly small as maximum residue ($m!$) will be small. When m is relatively larger, maximum residue ($m!$) grows significantly and stronger symmetry breaking is worthwhile. We will show in experiments on different benchmarks. When m becomes larger, SwapNext and SwapAny start to outperform DoubleLex and result in shorter runtime and less solutions. The winner between SwapNext and SwapAny depends on the larger dimension n. When n is moderately large, SwapAny generally performs better. When n is larger than a particular value, the overhead of SwapAny cannot be compensated by the saving from symmetry breaking and SwapNext becomes the best option. The exact cutoff points for selecting these methods are problem-specific, but the behaviours are general.

We suggest the following principles on selecting Lex-based symmetry breaking methods.

- DoubleLex: m is relatively small.
- SwapAny: m is relatively larger and n is moderately large.
- SwapNext: m is relatively larger and n is particularly large.

5 Experiment Results

In this section, we compare the pruning powers and running time efficiencies of DoubleLex, SwapNext, SwapAny and DoubleLex+AllPerm [Frisch *et al.*, 2003]. In general, search and solution count reductions are not necessarily strongly connected, since it is always possible to produce a symmetry breaking method which breaks all symmetries but only causes the solver to fail only on leaf nodes. In practice, however, we use global constraints to break symmetries which do not fail only at leaf nodes. Such global constraints also often have good propagation power, but with an overhead. Our experiments aim at demonstrating how well each method strikes balance between search reduction and overhead.

We compare these methods on problems with matrix models and search for all solutions and first solution in satisfaction problems and an optimization problem. The time limit is one hour unless otherwise specified. In the experiments, we use row-wise canonical ordering in $\leq_{lex}$ constraints. Minimum domain size variable heuristic and minimum value heuristic are used unless otherwise specified. The best results are highlighted in bold. In the tables, we report the number of solutions found within timeout (sol) , runtime in seconds (time(s)), and the number of failures encountered (failure) to demonstrate the improved symmetry breaking power of the different methods. We represent large numbers in millions (M). For instances that cannot be solved within time limit, runtime would be noted by "–" and the number of solutions found and failures will be marked with "≥". All experiments are conducted using Gecode 5.0.0 on Intel(R) Xeon(R) CPU E5-2630 v2 2.60GHz processor with 250G memory.

	DoubleLex			**SwapNext**			**SwapAny**		
(d m n)	time(s)	sol	failure	time(s)	sol	failure	time(s)	sol	failure
3 4 6	110.53	86M	**0**	61.07	48M	39	**41.53**	**30M**	50
3 4 7	1672.59	1235M	**0**	947.88	706M	97	**630.88**	**441M**	147
3 5 5	413.07	318M	**0**	201.36	154M	245	**121.68**	**89M**	249
4 3 6	57.0	53M	**0**	42.07	39M	**0**	**38.81**	**28M**	**0**
4 3 7	667.13	581M	**0**	503.42	438M	**0**	**464.47**	**308M**	**0**
4 4 4	21.08	21M	**0**	**13.8**	13M	14	13.98	**9M**	20
4 4 5	1600.98	1361M	**0**	1013.07	860M	193	**759.65**	**548M**	295

Table 2: Unconstrained matrix

5.1 Finding All Solutions

Unconstrained matrix. We compare the "pure" symmetry breaking effects of three methods on unconstrained $m \times n$ matrices with domain size d and results are reported in Table 2. On average, the solution set size of DoubleLex is 1.66 times larger than that of SwapNext, which runs 1.65 times faster than DoubleLex. The solution set size of DoubleLex is 2.57 times larger than that of SwapAny, which runs 2.21 times faster than DoubleLex. These demonstrate the stronger symmetry breaking power of SwapNext and SwapAny in terms of reduced exploration of symmetric solutions.

Error correcting code - Lee Distance (ECCLD). The problem [Frisch *et al.*, 2003] aims at finding a codebook consisting of m codewords, where each codeword is a n-character string from the alphabet $\{1, 2, 3, 4\}$ such that the Lee-Distance between each two codewords is d. The problem can be naturally modelled using an $m \times n$ matrix where each row $[x_{i,1}, \ldots, x_{i,n}]$ constitutes a codeword. As we could not find available implementation of AllPerm, we simulate AllPerm as in [Lee and Li, 2012] using the global cardinality constraints. Results are shown in Table 3. Experiment shows SwapAny outperforms all other methods. All other three methods have at least left one instance unsolved within time limit. On average, the solution set size of DoubleLex is 2.15 times larger than that of SwapNext, which runs 1.56 times faster than DoubleLex. DoubleLex+AllPerm and SwapNext achieve similar performances both in terms of solution set size and runtime. SwapAny achieves the best performance on almost all instances, which is on average 2.3 times faster than DoubleLex and results in 3.8 times smaller solution set. This agree with our suggestion that SwapAny is the best option when m is relatively larger and n is moderately large.

Balanced incomplete block designs (BIBD). The problem is well-known from design theory that requires a configuration distributing v objects into b blocks. Each object appears in r out of b blocks and there are k objects in each block. In addition, any two objects meet each other in the same block for exactly λ times. The problem can be modelled using a $v \times b$ 0/1 matrix where $x_{i,j} = 1$ indicates that Object i is allocated to Block j. As the occurrences of 0 and 1 on each row

	DoubleLex			SwapNext			SwapAny			DoubleLex+AllPerm		
(m n d)	time(s)	sol	failure	time(s)	sol	failure	time(s)	sol	failure	time(s)	sol	failure
5 6 4	2076.19	29M	73M	1130.03	15M	39M	**643.55**	**8M**	**21M**	1133.57	14M	35M
5 7 4	–	≥40M	≥129M	–	≥38M	≥125M	**3105.98**	**31M**	**103M**	–	≥37M	≥118M
6 5 4	1050.66	5M	31M	554.67	3M	16M	**296.51**	**1M**	**8M**	558.73	2M	15M
7 5 4	3066.26	4M	75M	1569.5	2M	37M	**775.4**	**926468**	**17M**	1549.5	2M	35M
8 5 4	–	≥797553	≥70M	3381.09	677037	64M	**1541.73**	**254972**	**28M**	3279.09	622786	60M
7 7 2	20.08	115296	236767	13.7	46405	149364	**13.06**	**31304**	**116828**	15.12	73943	166390
8 7 2	37.39	137636	347764	24.32	43991	198648	**21.8**	**27964**	**150397**	27.3	85354	237070
7 8 2	48.37	277020	522165	**31.32**	111309	325331	31.92	**75008**	**251072**	35.99	176552	360966
8 8 2	99.75	417083	833223	**56.41**	129726	462256	57.63	**82516**	**343965**	71.5	255728	555759

Table 3: Error correcting code - Lee Distance (ECCLD)

	DoubleLex			SwapNext			SwapAny		
(v b r k λ)	time(s)	sol	failure	time(s)	sol	failure	time(s)	sol	failure
5 40 16 2 4	0.06	**1**	**243**	**0.04**	**1**	**243**	1.85	**1**	**243**
5 50 20 2 5	**0.07**	**1**	**361**	0.08	**1**	**361**	6.49	**1**	**361**
6 30 10 2 2	0.06	**1**	212	**0.03**	**1**	**206**	1.01	**1**	**206**
7 35 15 3 5	15.37	64601	471356	**8.16**	**16744**	**163452**	14.0	**16744**	**163452**
7 42 18 3 6	152.82	432193	4M	**58.94**	**109433**	**1M**	76.65	**109433**	**1M**
7 49 21 3 7	1016.79	2M	28M	**362.0**	**609429**	**8M**	439.87	**609429**	**8M**
7 56 24 3 8	–	≥5M	≥95M	**2115.64**	**2M**	**52M**	2350.41	**2M**	**52M**
8 28 14 4 6	606.96	2M	14M	219.27	711707	4M	**176.9**	**596399**	**3M**

Table 4: Balanced incomplete block designs (BIBD)

are the same, DoubleLex+AllPerm degenerates to DoubleLex so we compare among the other three methods. The results in Table 4 agree with our prediction on the method with best performance: When the number of row v is small (5), DoubleLex performs well and stronger symmetry breaking is not in need. When v grows, SwapNext outperforms DoubleLex. SwapNext also outperfoms SwapAny on instances with large number of columns (b) since SwapNext explores similar numbers of solutions as SwapAny with much less overhead. SwapAny wins over SwapNext on only one instance where b is not too large.

5.2 Optimization

ECCLD (optimization version). We also test the efficiency on an optimization version of ECCLD. The goal is to minimize the average absolute equal occurrence discrepancy over the columns, which is a metric for selecting a good Cover Array [Kim *et al.*, 2017]. The results are shown in Table 5. As SwapNext and DoubleLex+AllPerm achieve similar performance, we skip DoubleLex+AllPerm and compare among the three other methods. SwapAny still outperforms the other two methods. In particular, SwapAny is 2.42 times faster than DoubleLex on average.

5.3 Finding First Solution

We test also if the symmetry breaking methods help in finding first solution. Since symmetry breaking is not worthwhile when there is little search, we experiment on larger ECCLD and BIBD instances. In Tables 6 & 7, we report the runtime and the number of failures of each method. The "NoSB" column reports the result where no symmetry breaking is used.

ECCLD (first solution). We experiment with the rowwise variable heuristic and minimum value heuristic as it greatly outperforms the default heuristic we used in finding all solutions. While we have experimented with quite a number of instances, we report only results of those requiring more than 10s to solve with at least one method (including NoSB). The time limit is 2 hours. The results are reported in Table 6. NoSB is essentially impractical to use as compared to the other four methods. As predicted theoretically, SwapAny has the least failures among all methods. Instances (13 8 6), (13 9 6) and (13 10 6) are hard instances requiring much search, and SwapAny performs the best in terms of runtime with its strongest pruning power. The number of columns of all instances are not too large therefore enforcing strong symmetry breaking for search reduction is worthwhile. DoubleLex and DoubleLex+AllPerm are the worst in runtime among the symmetry breaking methods. The search heuristic always tries assigning 1 to all variables in the first row, making AllPerm constraints trivially satisfied. Thus, DoubleLex and DoubleLex+AllPerm achieve exactly the same performance both in runtime and failures.

	DoubleLex		SwapNext		SwapAny	
(n m d)	failure	time(s)	failure	time(s)	failure	time(s)
6 4 8	487	0.024	498	0.023	**427**	**0.020**
5 4 6	187201	3.808	119649	2.406	**78637**	**1.718**
5 10 2	45516	5.657	27106	3.533	**21262**	**3.464**
4 8 4	1M	61.239	540285	32.000	**300808**	**18.744**
5 5 6	3M	99.927	2M	60.581	**1M**	**38.155**
6 4 4	7M	158.201	4M	99.001	**2M**	**66.319**
5 6 6	15M	479.787	18M	291.820	**5M**	**173.396**
8 4 4	74M	1772.549	45M	1107.320	**27M**	**715.061**
6 5 4	100M	2666.142	53M	1443.238	**28M**	**822.779**

Table 5: ECCLD optimization

BIBD (first solution). The rowwise variable heuristic and minimum value heuritic are equivalent to the default heuristic here since the domains are binary. The results are reported in Table 7. Here, we report results of only instances requiring more than 10s with at least one symmetry breaking method (excluding NoSB). NoSB can solve none of the instances.

	NoSB		**DoubleLex**		**SwapNext**		**SwapAny**		**DoubleLex+AllPerm**	
(m n d)	time(s)	failure	time(s)	failure	time(s)	failure	time(s)	failure	time(s)	failure
13 8 6	-	$\geq$ 96M	3580.28	27M	2806.84	22M	**1942.92**	**13M**	3755.52	27M
9 9 4	1139.22	20M	4.03	53749	**1.74**	20533	2.72	**16629**	4.72	53749
13 9 6	-	$\geq$ 91M	5440.65	37M	4282.18	30M	**3333.47**	**19M**	5595.31	37M
10 10 4	3384.16	36M	6.76	55085	**2.73**	21096	4.34	**17122**	8.22	55085
13 10 6	-	$\geq$ 84M	6584.31	41M	5192.47	33M	**4577.75**	**21M**	6759.68	41M

Table 6: ECCLD (first solution)

	NoSB		**DoubleLex**		**SwapNext**		**SwapAny**	
(v b r k λ)	time(s)	failure	time(s)	failure	time(s)	failure	time(s)	failure
8 56 21 3 6	-	$\geq$ 21M	**13.75**	139470	20.12	**138111**	226.91	**138111**
9 54 24 4 9	-	$\geq$ 16M	**135.0**	970564	166.15	**933302**	843.92	**933302**
9 60 20 3 5	-	$\geq$ 16M	**38.17**	289398	46.37	**258128**	505.03	**258128**
9 72 24 3 6	-	$\geq$ 11M	**18.41**	112582	29.7	**111304**	1366.2	**111304**
9 84 28 3 7	-	$\geq$ 10M	**522.85**	**2M**	734.05	**2M**	-	$\geq$ 17423
9 96 32 3 8	-	$\geq$ 9M	**171.52**	786687	313.33	**783238**	-	$\geq$ 3728
10 45 18 4 6	-	$\geq$ 18M	664.62	11M	**435.07**	**7M**	545.43	**7M**

Table 7: BIBD (first solution)

DoubleLex wins in runtime in all instances except the last since relatively little search is required to find the first solution and DoubleLex is comparable in search reduction to SwapNext and SwapAny. SwapAny timeouts on instances (9 84 28 3 7) and (9 96 32 3 8) as a prohibitively large number of constraints are introduced by the large number of columns. DoubleLex still wins since SwapNext only achieves minor search reduction. The number of columns of the last instance is moderate and some search ($<$ 10M failures) is needed to find the first solution. SwapNext prevails.

6 Conclusion

We give a factorial improvement to the best known upper bound of remaining solutions over all symmetry classes of G_{mat}. From there, we propose Maximum Residue (MR) as a notion to evaluate the strength of a symmetry breaking method. We further study the MR bounds of SwapNext and SwapAny, which are extensions to DoubleLex for stronger symmetry breaking. The theoretical comparisons allow us to suggest general principles on selecting Lex-based symmetry breaking methods based on the dimensions of the matrix models and they are confirmed by our experiments. We believe investigation on tighter bounds of SwapNext, SwapAny and other existing symmetry breaking methods are meaningful future works. Studying MRs of other methods potentially give insights and hints for deriving alternative choices of symmetry breaking constraints.

References

[Flener *et al.*, 2001] Pierre Flener, Alan M. Frisch, Brahim Hnich, Zeynep Kiziltan, Ian Miguel, and Toby Walsh. Matrix modelling. In *CP'01 Workshop on Modelling and Problem Formulation*, page 223, 2001.

[Flener *et al.*, 2002] Pierre Flener, Alan M. Frisch, Brahim Hnich, Zeynep Kiziltan, Ian Miguel, Justin Pearson, and Toby Walsh. Breaking row and column symmetries in matrix models. In *CP'02*, pages 462–477. Springer, 2002.

[Frisch *et al.*, 2003] Alan M. Frisch, Chris Jefferson, and Ian Miguel. Constraints for breaking more row and column symmetries. In *CP'03*, pages 318–332. Springer, 2003.

[Katsirelos *et al.*, 2010] George Katsirelos, Nina Narodytska, and Toby Walsh. On the complexity and completeness of static constraints for breaking row and column symmetry. In *CP'10*, pages 305–320. Springer, 2010.

[Kim *et al.*, 2017] Youngil Kim, Dae-Heung Jang, and Christine M. Anderson-Cook. Selecting the best wild card entries in a covering array. *Quality and Reliability Engineering International*, 33(7):1615–1627, 2017.

[Kiziltan, 2004] Zeynep Kiziltan. *Symmetry breaking ordering constraints*. PhD thesis, Uppsala University, 2004.

[Lee and Li, 2012] Jimmy H.M. Lee and Jingying Li. Increasing symmetry breaking by preserving target symmetries. In *CP'12*, pages 422–438. Springer, 2012.

[Lee and Zhu, 2016] Jimmy H.M. Lee and Zichen Zhu. Static symmetry breaking with the reflex ordering. In *IJCAI'16*, pages 758–765, 2016.

[McDonald and Smith, 2002] Iain McDonald and Barbara Smith. Partial symmetry breaking. In *CP'02*, pages 431–445. Springer, 2002.

[Narodytska and Walsh, 2013] Nina Narodytska and Toby Walsh. Breaking symmetry with different orderings. In *CP'13*, pages 545–561. Springer, 2013.

[Smith, 2014] Barbara Smith. Symmetry breaking constraints in constraint programming. Slides published online. Retrieved June 12, 2019, from https://www.slideserve.com/lore/symmetry-breaking-constraints-in-constraint-programming, 2014.