Symmetry-Breaking Constraints for Directed Graphs

Jussi Rintanen[a] **and Masood Feyzbakhsh Rankooh**[b]

[a]Aalto University, Helsinki, Finland
[b]Tampere University, Tampere, Finland

Abstract. Finding a graph with given properties occurs as a sub-problem of many important problems in A.I. and other areas of computer science. Main approaches to solving such problems include automated reasoning and constraint satisfaction methods. These can often be substantially sped up by considering only a subset of graphs for each equivalence class of isomorphic graphs, motivating the use of symmetry-breaking constraints for graphs.

We present a symmetry-breaking constraint for directed graphs, generalizing earlier works that have presented such constraints for undirected graphs without loops, and experimentally demonstrate their effectiveness.

1 Introduction

Many search problems exhibit symmetries, and eliminating all or some of the symmetry may make the search problem exponentially smaller and easier to solve. Symmetry elimination has been important in solving a broad range of problems in AI [3] and other areas of computer science [5].

A main approach to eliminating symmetry is to partition the states in the search space to equivalence classes with respect to symmetry, and then map each encountered state to a canonical representative of its class [17], thus reducing the number of states to be considered. This requires the search methods to implement general-purpose symmetry eliminations methods. Implementations of automated reasoning such as constraint programming or satisfiability checking typically lack in-built symmetry reduction, and a different approach is followed, by introducing *symmetry-breaking constraints* which eliminate some of the symmetric solutions from consideration [3].

A specific class of problems solved with automated reasoning methods is finding a graph with given properties, with applications for example in network design [13] and automated planning and decision-making [1, 8, 14, 6].

Graph-finding problems often assume a fixed set of nodes, and the objective is to choose a set of edges for the graph so that it satisfies some given constraints. If the *names* of all or most nodes can be permuted arbitrarily, without impacting the given constraints, then each solution graph is only one representative of an equivalence class of isomorphic graphs that are all solutions. This permutability is not recognized by most constraint solvers, and it can lead to unnecessary search when a high number of isomorphic graphs is considered separately. Breaking the symmetries in each equivalence class, by considering only some or only one element of each class, can improve constraint solvers' performance substantially.

Symmetry-breaking constraints have been earlier proposed for directed acyclic graphs [16] and undirected graphs [2]. Many applications involve general (possibly cyclic) directed graphs, and symmetry-breaking for them is not covered by existing methods.

In this work, we address symmetry-breaking for arbitrary graphs, undirected, directed, cyclic and acyclic, and also contribute to the work by Codish et al. [2] with an alternative symmetry-breaking constraint for undirected graphs.

The structure of the paper is as follows. In Section 2 we discuss the basic ideas in earlier works on constraints for breaking symmetries arising from graph isomorphism. Section 3 presents our general symmetry-breaking constraint that covers both directed and undirected graphs. Section 4 extends the constraint to handle the DAG symmetry constraint proposed by Shlyakhter [16]. Section 5 specializes the constraint to undirected graphs, providing an alternative to constraints presented earlier. Section 6 experimentally demonstrates the effectiveness of our constraints on directed graphs, and compares them to earlier symmetry-breaking constraints for undirected graphs and directed acyclic graphs. Section 7 briefly discusses related work, and Section 8 concludes the paper.

2 Background

Any graph can be represented in terms of an *adjacency matrix*, which is an $N \times N$ 0-1 matrix with element (i, j) indicating whether there is a directed edge from node $i \in \{1, \ldots, N\}$ to node $j \in \{1, \ldots, N\}$.

Shlyakhter [16] proposes reducing symmetry in directed acyclic graphs by the restriction to strictly triangular matrices, so that from any node, there are directed edges to other nodes with a higher index only (the matrix is *strictly upper triangular*) or with a lower index only (the matrix is *strictly lower triangular*.) This restriction not only guarantees that the graph is acyclic, but also eliminates any permutation of the node indices that would have the directed edges going into the wrong direction.

Later, Codish et al. [2] formalize, analyze and generalize a class of symmetry-breaking constraints for undirected graphs first used by Miller and Prosser [12]. The basic idea is to constrain the rows of the adjacency matrix to be lexicographically non-decreasing, with each row understood as a sequence of 0s and 1s. For undirected graphs this guarantees that swapping two nodes cannot move the matrix earlier in the lexicographic ordering, in which matrices are understood as sequence of 0s and 1s read from top to bottom and from left to right. While this idea works for undirected graphs, for directed graphs it does not.

Example 1. *Consider the adjacency matrix on the left.*

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

Row 1 lexicographically precedes row 2, which precedes row 3, but swapping rows and columns 2 and 3 leads to the lexicographically better matrix above on the right.

The proof showing that node swaps cannot improve the matrix if the rows are lexicographically increasing [2], relies on the adjacency matrix having diagonal symmetry, a property that does not hold for the more general class of directed graphs.

Further, the idea that nodes could be reordered so that the rows of adjacency matrices are lexicographically non-decreasing does not work for graphs in general.

Example 2. *Consider the following adjacency matrices. The first one represents a graph with three nodes with self-loops, and the other two a graph with three nodes that form a cycle, the first with node ordering* $1, 2, 3$, *and the second with* $1, 3, 2$.

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

In the first matrix, the rows are lexicographically decreasing, and swapping any two nodes produces the same matrix. For the other two matrices, any node swap produces the other matrix, always containing one pair of decreasing rows.

Hence, for some graphs the constraint that the rows have to be non-decreasing would not be satisfied by any of the adjacency matrices obtained by permuting some the nodes, which means that imposing this kind of symmetry-breaking constraint would eliminate some possible solutions, in some cases incorrectly making problems unsolvable. Therefore, for symmetry-breaking for general (directed and undirected) graphs some other property than lexicographically ordered rows has to be employed.

In this work we look at the idea of lexicographically ordered adjacency matrices more thoroughly, and generalize it so that it works also for directed graphs. Our main result is a symmetry-breaking constraint for arbitrary directed graphs, which includes undirected graphs as a special case.

3 Lex-Constraints for Directed Graphs

We make the observation that, in order to assess the impact of swapping two nodes, one cannot focus on the mutual lexicographic ordering of the corresponding rows of the adjacency matrix, but must also observe the corresponding columns. The constraint that we derive checks whether the *result* of the swap will be a lexicographically better adjacency matrix, and then rules out any matrix that can be improved by a single swap. Swapping two nodes corresponds to swapping both the corresponding rows and the columns of the adjacency matrix [2].

Interestingly, we will show that this constraint can be encoded almost as easily as the lexicographic ordering constraint on rows, and that it works equally well for both undirected and directed graphs as a part of the symmetry-breaking constraint. Further, while the constraint by Codish et al. [2] works with increasing lexicographic row orderings only, our constraint can be equally easily defined for both increasing and decreasing orderings.

Two sequences are ordered $x_1 \cdots x_n < y_1 \cdots y_n$ iff there is i such that $x_i < y_i$ and $x_k = y_k$ for all $k < i$. Two sequences are ordered $\sigma \leq \sigma'$ if either $\sigma = \sigma'$ or $\sigma < \sigma'$.

Consider the following matrix, with columns and rows i and j written explicitly. When swapping nodes i and j, only columns i and

j and rows i and j change, and the remaining entries in the matrix remain unchanged.

$$
\begin{pmatrix}
& a_{1,i} & & a_{1,j} & \\
& \vdots & & \vdots & \\
& a_{i-1,i} & & a_{i-1,j} & \\
a_{i,1}\cdots a_{i,i-1} & a_{i,i} & a_{i,i+1}\cdots a_{i,j-1} & a_{i,j} & a_{i,j+1}\cdots a_{i,n} \\
& a_{i+1,i} & & a_{i+1,j} & \\
& \vdots & & \vdots & \\
& a_{j-1,i} & & a_{j-1,j} & \\
a_{j,1}\cdots a_{j,i-1} & a_{j,i} & a_{j,i+1}\cdots a_{j,j-1} & a_{j,j} & a_{j,j+1}\cdots a_{j,n} \\
& a_{j+1,i} & & a_{j+1,j} & \\
& \vdots & & \vdots & \\
& a_{n,i} & & a_{n,j} &
\end{pmatrix}
$$

After the swap, the contents of the matrix are as follows.

$$
\begin{pmatrix}
& a_{1,j} & & a_{1,i} & \\
& \vdots & & \vdots & \\
& a_{i-1,j} & & a_{i-1,i} & \\
a_{j,1}\cdots a_{j,i-1} & a_{j,j} & a_{j,i+1}\cdots a_{j,j-1} & a_{j,i} & a_{j,j+1}\cdots a_{j,n} \\
& a_{i+1,j} & & a_{i+1,i} & \\
& \vdots & & \vdots & \\
& a_{j-1,j} & & a_{j-1,i} & \\
a_{i,1}\cdots a_{i,i-1} & a_{i,j} & a_{i,i+1}\cdots a_{i,j-1} & a_{i,i} & a_{i,j+1}\cdots a_{i,n} \\
& a_{j+1,j} & & a_{j+1,i} & \\
& \vdots & & \vdots & \\
& a_{n,j} & & a_{n,i} &
\end{pmatrix}
$$

To compare the matrices before and after the swap, it suffices to lexicographically compare the entries that may have changed, as vectors obtained by reading them row by row from top to bottom, and each row read from left to right.

There is some redundancy in these sequences, and they can be simplified, as demonstrated by the following lemma.

Lemma 1. *Let* $\sigma_1, \sigma_2, \sigma_3, \sigma_A$ *and* σ_B *be any sequences such that* $|\sigma_A| = |\sigma_B|$. *Then* $\sigma_1 \sigma_A \sigma_2 \sigma_B \sigma_3 < \sigma_1 \sigma_B \sigma_2 \sigma_A \sigma_3$ *iff* $\sigma_1 \sigma_A \sigma_2 \sigma_3 < \sigma_1 \sigma_B \sigma_2 \sigma_3$.

Proof. $\sigma_1 \sigma_A \sigma_2 \sigma_B \sigma_3 < \sigma_1 \sigma_B \sigma_2 \sigma_A \sigma_3$ if and only if $\sigma_A < \sigma_B$ if and only if $\sigma_1 \sigma_A \sigma_2 \sigma_3 < \sigma_1 \sigma_B \sigma_2 \sigma_3$.

In other words, both orderings hold if and only if $\sigma_A < \sigma_B$, as the first possible differences in the sequences are in σ_A and σ_B. If $\sigma_A = \sigma_B$, then the sequences are the same. Hence for the second occurrences of σ_A and σ_B the ordering $\sigma_B < \sigma_A$ cannot impact the overall ordering, and therefore they can be ignored. □

Hence we can ignore the symbols in one position in both sequences, if the same symbols occur earlier in the sequences the other way round. This allows restricting the lexicographic comparison to only a part of the entries that the (i, j) swap may change: column i and row i will be compared to column j and row j.

Now we define the constraint $lex^{\leq}_{i,j}$ by requiring that the adjacency matrix does not become lexicographically smaller by the (i, j) swap. This is by comparing the possibly changing elements of the matrix, as shown below, to the elements in the same positions after the swap.

$$
\begin{aligned}
a_{1,i} \cdots a_{i-1,i} \\
a_{i,1} \cdots a_{i,n} \\
a_{i+1,i} \cdots a_{n,i}
\end{aligned}
$$

After the swap these entries have the following contents.

$$a_{1,j} \cdots a_{i-1,j}$$
$$a_{j,1} \cdots a_{j,i-1} a_{j,j} a_{j,i+1} \cdots a_{j,j-1} a_{j,i} a_{j,j+1} \cdots a_{j,n}$$
$$a_{i+1,j} \cdots a_{j-1,j} a_{i,j} a_{j+1,j} \cdots a_{n,j}$$

So we compare these vectors lexicographically, element by element $a_{1,i} < a_{1,j}, \ldots, a_{n,i} < a_{n,j}$, and require that the first vector is not lexicographically strictly greater than the second. The lengths of both vectors are $2n - 1$, most easily seen from the three components of the first vector, which respectively have lengths $i - 1$, n, and $n - (i + 1) + 1$.

Several encodings of lexicographical comparison of Boolean vectors as formulas exist [4]. In Section 6 we use what is closest to AND decomposition with common sub-expression elimination.

A simple encoding lexicographical comparison of Boolean vectors as a propositional formula is by the following recursive definition, where the b_i represent individual Boolean values 0 and 1, the σ_i are sequences of Boolean values, and ϵ is the empty sequence.

$$\epsilon \leq \epsilon \quad = \quad \top$$
$$b_1 \sigma_1 \leq b_2 \sigma_2 \quad = \quad (\neg b_1 \wedge b_2) \vee ((b_1 \leftrightarrow b_2) \wedge (\sigma_1 \leq \sigma_2))$$

As the vectors to be compared have $2n - 1$ elements, and these formulas have a size that is linear in $2n - 1$ for a graph with n nodes. For the pairwise comparisons of rows used by Codish et al. [2] the vectors have $n - 1$ elements, and the comparisons are encoded similarly. The reason needing only $n - 1$ elements in the comparisons is the diagonal symmetry of the adjacency matrix of undirected graphs. Later we see in Section 5 that our comparison specialized to undirected graphs similarly only has $n - 1$ elements, and hence leads to constraints of essentially the same size as those of Codish et al. [2].

Similarly to Codish et al., we can use this constraint for all pairs of consecutive nodes $i, i + 1$ for $i \in \{1, \ldots, n - 1\}$. Unlike the row-wise lexicographic comparison in the basic form of the Codish et al. constraint, $lex_{i,j}^{\leq} \wedge lex_{j,k}^{\leq}$ does not entail $lex_{i,k}^{\leq}$, and hence the quadratic-size constraint $lex_{i,j}^{\leq}$ for all $1 \leq i < j \leq n$ can be useful.

Theorem 2. *If an adjacency matrix M satisfies the constraint $lex_{i,j}^{\leq}$, then $M \leq M'$ for the matrix M' obtained by doing any column and row swap (i, j) for $1 \leq i < j \leq n$.*

Proof. Sketch: The constraint $lex_{i,j}^{\leq}$ is defined so that either no element in M is changed by the column swap (i, j), or the first differing element in the first row that differs, is 0 in M and 1 in M', and hence $M \leq M'$. □

Limiting node swaps to consecutive nodes $i, i+1$ produces weaker symmetry-breaking than considering all possible node swaps.

Example 3. *Node swaps $(1, 2)$ or $(2, 3)$ do not lexicographically improve the matrix on the left. But swapping nodes 1 and 3 leads to the minimal matrix on the right.*

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

Even though often quite strong, the constraints we have defined above do not in general eliminate all symmetry, so there may be more than one graph satisfying the constraint. Importantly, however, the canonical graph in every equivalence class of isomorphic graphs does satisfy these constraints.

4 Directed Acyclic Graphs

Obviously, the constraints given in the preceding section work also for the subclass of directed acyclic graphs. However, the question arises, whether these constraints are compatible with Shlyakhter's [16] requirement that the adjacency matrix is strictly upper triangular, and whether combining these two will yield performance improvements in constraint solving.

The issue is that the constraints in Section 3 attempt to rule out node numberings by checking whether a node swap could lead to a lexicographically better adjacency matrix, while considering arbitrary such matrices, and not only strictly upper triangular ones. What if the lexicographically best strictly upper triangular adjacency matrix could be improved by a node swap that leads to a matrix that is not strictly upper triangular?

Example 4. *Consider the following adjacency matrix on the left, and the lexicographically increasing order. Note that this is the lexicographically best strictly upper triangular matrix for this graph, as any other permutation of the node names breaks strict upper triangularity.*

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

Now consider the node swap $(2, 3)$, which results in the matrix on the right, which is not strictly upper triangular.

There are similar examples for lexicographically decreasing orders. For the matrix below on the left, the $(2, 3)$ swap produces the lexicographically better matrix, on the right, which is not strictly upper triangular.

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

So the constraints developed in Section 3 sometimes rule out the lexicographically best adjacency matrix that is strictly upper triangular. One way to combine the constraints is to only consider swaps that do not break strict upper triangularity.

To devise a symmetry-breaking constraint for adjacency matrices that must satisfy strict upper triangularity, as proposed by Shlyakhter [16], consider the (i, j) swap.

After the swap, the contents of the matrix are as follows.

$$\begin{pmatrix} & a_{1,j} & & & & a_{1,i} & \\ & \vdots & & & & \vdots & \\ & a_{i-1,j} & & & & a_{i-1,i} & \\ \hline 0 \cdots 0 & 0 & 0 & \cdots & 0 & 0 & a_{j,j+1} \cdots a_{j,n} \\ \hline & a_{i+1,j} & & & & 0 & \\ & \vdots & & & & \vdots & \\ & a_{j-1,j} & & & & 0 & \\ \hline 0 \cdots 0 & a_{i,j} & a_{i,i+1} \cdots a_{i,j-1} & & 0 & a_{i,j+1} \cdots a_{i,n} \\ \hline & 0 & & & & 0 & \\ & \vdots & & & & \vdots & \\ & 0 & & & & 0 & \end{pmatrix}$$

We can see from this matrix that strict upper triangularity is preserved if and only if the elements $(i+1,j), \ldots, (i,j)$ and $(i,i+1), \ldots, (i,j-1)$ are all 0. So it suffices to impose this condition to make it compatible with strict upper triangularity. The resulting symmetry-breaking constraint for directed acyclic graphs is

$$^A lex_{i,j}^{\leq} = \left(\bigwedge_{k=i+1}^{j-1} \neg a_{k,j} \wedge \bigwedge_{k=i+1}^{j} \neg a_{i,k} \right) \to lex_{i,j}^{\leq}.$$

Further, the $lex_{i,j}^{\leq}$ constraint on the right can be improved by observing that all elements on the diagonal and below are zero, and do not need to be compared. As a result, no reference to those need to be made at all, and – as in standard implementations of the Shlyakhter scheme – it is sufficient to represent only those $a_{i,j}$ with $i < j$. The lexicographic comparison simplifies to the following.

$$a_{1,i} \cdots a_{i-1,i} a_{i,j+1} \cdots a_{i,n} \leq a_{1,j} \cdots a_{i-1,j} a_{j,j+1} \cdots a_{j,n}$$

5   Undirected Graphs

For undirected graphs, due to the diagonal symmetry of the adjacency matrix, the general constraint from Section 3 can be simplified, limiting the lexicographic comparison for (i,j) swaps to

$$a_{1,i}, \ldots, a_{i,i}, \ldots, a_{i,n}$$
$$\leq a_{1,j}, \ldots a_{i-1,j}, a_{j,j}, a_{j,i+1}, \ldots, a_{j,j-1}, a_{j,i}, a_{j,j+1}, \ldots, a_{j,n}.$$

We denote the corresponding swap test by $^U lex_{i,j}^{\leq}$. The vectors being compared here have $n-1$ elements, in contrast to the $2n-1$ elements in the general case for directed graphs.

Codish et al. [2] have introduced two types of lexicographic constraints that break symmetry for undirected graphs. Their basic symmetry breaking constraint requires the rows of the matrix to be lexicographically sorted. They also introduce a new relation $\leq_{\{i,j\}}$ that lexicographically compares rows i and j after removing columns i and j from both rows. Their improved symmetry-breaking constraint imposes $\leq_{\{i,j\}}$ on all pairs for rows i and j such that $i < j$. Here, we show that our constraint is strictly stronger than their basic symmetry breaking constraint. We also prove that in the case of simple graphs (graphs without self-loops), our constraints is equivalent to the improved symmetry breaking constraint of Codish et al.

Theorem 3. *If an adjacency matrix M satisfies $^U lex_{i,i+1}^{\leq}$ then row i of M is lexicographically ordered before row $i+1$ of M.*

Proof. Assume the contrary: row i is not lexicographically ordered before row $i+1$. The proof of Theorem 1 of [2] shows that swapping rows i and $i+1$ results in a matrix M' such that $M' < M$, which contradicts our Theorem 2. $\quad\square$

Theorem 3 together with Example 5 below show that our constraint is strictly stronger than the basic symmetry breaking constraint of Codish et al.

Example 5. *Consider the adjacency matrix on the left.*

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

The rows of the matrix are ordered lexicographically, but swapping the first two nodes would lead to the matrix on the right that is lexicographically ordered before. Of these two matrices, only the second one satisfies our constraint $lex_{0,1}^{\leq} \wedge lex_{1,2}^{\leq}$. Notice that the rows of this matrix are not lexicographically ordered.

Theorem 4. *For simple undirected graphs, $^U lex_{i,j}^{\leq}$ is equivalent to $\leq_{\{i,j\}}$ of Codish et al. [2].*

Proof. The constraint $^U lex_{i,j}^{\leq}$ expands to $a_{1,i}, \ldots, a_{i,i}, \ldots, a_{i,n} \leq a_{1,j}, \ldots a_{i-1,j}, a_{j,j}, a_{j,i+1}, \ldots, a_{j,j-1}, a_{j,i}, a_{j,j+1}, \ldots, a_{j,n}$. We can rewrite the left-hand side of this constraint by considering the symmetry of the matrix as

$$a_{i,1}, \ldots, a_{i,i-1}, a_{i,i}, a_{i,i+1}, \ldots, a_{i,j-1}, a_{i,j}, a_{i,j+1}, \ldots, a_{i,n}.$$

Also considering the symmetry, the right-hand side becomes

$$a_{j,1}, \ldots, a_{j,i-1}, a_{j,j}, a_{j,i+1}, \ldots, a_{j,j-1}, a_{j,i}, a_{j,j+1}, \ldots, a_{j,n}.$$

Now, because the graph is simple, we have $a_{i,i} = a_{j,j} = 0$, and because of symmetry $a_{i,j} = a_{j,i}$. Therefore, lexicographically comparing these two sequences is equivalent to lexicographically comparing rows i and j while ignoring columns i and j. $\quad\square$

6   Experiments

We have experimented with all three variants of our symmetry-breaking constraint: for undirected graphs, for directed graphs, and for directed acyclic graphs.

In the first experiment, we compare our constraint for undirected graphs to the constraints by Codish et al. [2] with the extremal graph problems they experimented with. After that, we investigate our constraints for directed graphs (both general and acyclic) with directed variants of the same extremal graph problems.

We used the KisSAT SAT solver (version 1.0.3) for all our experiments, and ran the experiments with an Intel i7-3930K CPU with 32 GB of RAM under Ubuntu Linux.

We first implemented the extremal graph problems considered by Codish et al. [2] which determine the maximum number of edges an undirected graph with v nodes and no cycles of length ≤ 4 can have.

We compared the Codish et al. constraints to ours on these problems, and give the results in Table 1. The columns $lex_{i,i+1}^{\leq}$ are for our constraint applied to pairs of consecutive nodes as $lex_{i,i+1}^{\leq}$ for $1 \leq i < n$, and by $lex_{i,j}^{\leq}$ the constraint applied to all pairs of nodes as $lex_{i,j}^{\leq}$ for $1 \leq i < j \leq n$. The UNSAT column is the sum of the CPU times spent by the SAT solver to determine unsatisfiability of all formulas that represent the problem for $E+1$ edges, where E is the maximum number of edges possible for the given graph, and for different applicable values of δ and Δ, as used by Codish et al. [2]. The SAT column is the CPU time it took to find a satisfiable formula from among the formulas that represent E edges, for different values

of δ and Δ, with all formulas solved in parallel, and solvers stopped as soon as one of the solvers determined satisfiability.

Timeouts (over 3600 seconds) are indicated in the tables with TO, and dashes $-$ are for cases where the degree lower δ and upper bound Δ parameters (as used by Codish et al. [2]) conflicted, trivially showing unsatisfiability without any SAT solving needed.

The results show our constraints to be roughly equally effective in speeding up constraint solving as the constraints by Codish et al. [2]. So our constraints, when specialized to the undirected case, work quite well.[1] As observed by Codish et al., the symmetry-breaking constraints tend to moderately slow down the solution of satisfiable instances, while they speed up unsatisfiable ones substantially.

Then we consider a directed version of the same problem, but without directed counterparts of the δ and Δ parameters that Codish et al. [2] obtained from theoretical results for the class of undirected graphs they considered. Our experiment just compares two versions of our symmetry-breaking constraint, as there are no comparable earlier works on symmetry-breaking for directed graphs.

This experiment, with results shown in Table 2, confirms the effectiveness of our symmetry-breaking constraint for directed graphs, in comparison to the case with no symmetry-breaking at all. Also, the $lex^{\leq}_{i,j}$ constraint systematically beats $lex^{\leq}_{i,i+1}$, despite the quadratic number of lexicographic comparisons.

Finally, we solved the same problem for directed acyclic graphs, for which the existing constraint is Shlyakhter's [16] restriction to strictly upper triangular adjacency matrices, in which $i < j$ for all edges (n_i, n_j). The results from this experiment in Table 3[2] demonstrate that combining Shlyakhter's constraint with our constraint leads to a performance improvement, but the improvement is not quite as dramatic as the one seen for undirected or general directed graphs. So the restriction to strictly upper triangular matrices alone is quite effective, despite its utter simplicity. Interestingly, and unlike in the previous experiments, symmetry constraints are effective in reducing runtimes also in the satisfiable cases, at least for the larger instances, in comparison to the Shlyakhter constraint.

7 Related Work

There is a long history of symmetry-breaking for constraint satisfaction problems [3], addressing the permutability of different aspects of constraints, including variables and their possible values, which leads to symmetry, as well as the detection of symmetries [15, 18]. Lexicographic orderings have been used as part of symmetry-breaking methods, and hence symmetry has been one of the motivations for investigating constraints for lexicographic orderings [7].

Closest related work is that of Codish et al. [2] who break symmetries by imposing an increasing lexicographic ordering on rows of adjacency matrices of undirected graphs. Our work presents a substantially generalized symmetry-breaking, also by lexicographic order-

ings, but covering all graphs, also directed ones, and both increasing and decreasing orderings. We have further shown how our constraint can be specialized to undirected graphs as well as to acyclic graphs, where acyclicity is maintained by Shlyakhter's [16] constraint on the adjacency matrix being strictly upper-triangular.

Breaking all symmetry in undirected graphs has been investigated by Itzhakov and Codish [10] and Heule [9]. These constraints are practical for small graphs with 10 nodes or less.

Kirchweger and Szeider [11] have implemented a method for symmetry-breaking for graphs inside a SAT solver. Specialized constraints and propagators may have a far better performance than symmetry-breaking constraints expressed in terms of general-purpose primitive constraints, but they require implementation work for every solver in which they are to be used. Constraints such as ours are applicable with any solver framework that can express basic Boolean constraints.

8 Conclusion

We have presented a symmetry-breaking constraint for both directed and undirected graphs, generalizing earlier works that limited to directed acyclic graphs or undirected graphs. Our constraint is based on ordering adjacency matrices of graphs lexicographically, and accurately representing the impact of node swaps on the ordering.

Unlike earlier constraints for undirected graphs, our constraint can enforce equally well both increasing and decreasing lexicographic orderings of adjacency matrices. Our experiments demonstrate that our constraint breaks symmetries equally effectively as the constraints by Codish et al. [2], which cover undirected graphs only.

Our constraint works with an earlier constraint for directed acyclic graphs that requires edges to go from lower indexed nodes to higher indexed nodes, which is weaker in breaking symmetries than constraints based on lexicographic orderings, and that our constraints substantially speed up constraint solving also in this case.

None of these constraints break all symmetry, as typically multiple elements in a symmetry class satisfy the constraints, not only the canonical element. Future work includes tightening the constraints further, and finding more specialized and more effective constraints for specific sub-classes of graphs.

References

[1] K. Chatterjee, M. Chmelik, and J. Davies. A symbolic SAT-based algorithm for almost-sure reachability with small strategies in POMDPs. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*, pages 3225–3232. AAAI Press, 2016.

[2] M. Codish, A. Miller, P. Prosser, and P. J. Stuckey. Constraints for symmetry breaking in graph representation. *Constraints*, 24:1–24, 2019.

[3] J. Crawford, M. Ginsberg, E. Luks, and A. Roy. Symmetry-breaking predicates for search problems. In L. C. Aiello, J. Doyle, and S. Shapiro, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR '96)*, pages 148–159. Morgan Kaufmann Publishers, 1996.

[4] H. A. Elgabou and A. M. Frisch. Encoding the lexicographic ordering constraint in SAT modulo theories. In *Thirteenth International Workshop on Constraint Modelling and Reformulation (ModRef 2014)*, pages 85–96, 2014. unpublished.

[5] E. A. Emerson and A. P. Sistla. Symmetry and model-checking. *Formal Methods in System Design: An International Journal*, 9(1/2):105–131, 1996.

[6] S. Fadnis and J. Rintanen. Planning with partial observability by SAT. In *Logics in Artificial Intelligence, 18th European Conference, JELIA 2023, September 2023, Proceedings*, volume 14281 of *Lecture Notes in Computer Science*, pages 605–620. Springer-Verlag, 2023.

[7] A. Frisch, B. Hnich, Z. Kızıltan, I. Miguel, and T. Walsh. Global constraints for lexicographic orderings. In *Principles and Practice of Constraint Programming-CP 2002: 8th International Conference, CP 2002*

---

[1] Note that some of the formulas which Codish et al. say are solved without search (in 0.0 seconds) by their solver and encodings, require substantial time with our implementation of all of the constraints. We have not determined the cause of this. Codish et al. leave some details of their implementation unexplained, including how the basic lexicographic comparison is implemented. Also, we have not implemented *equi-propagation* and *partial evaluation* which are part of the BEE solver used by them. Except for these a couple of cases, our runtimes are similar, and in some cases better than those reported by them.

[2] The maximum edge numbers for a given number of vertices for digraphs no length ≤ 4 undirected cycles seems to follow the integer sequence A002620 https://oeis.org/A002620 $a(n) = floor(\frac{n}{2}) \times ceiling(\frac{n}{2})$. Interestingly, this is also the maximum number of edges for undirected graphs without triangles (cycles of length 3).

$|V|$	none		sb_l [2]		sb_l^* [2]		$^U lex^{\leq}_{i,i+1}$		$^U lex^{\leq}_{i,j}$	
	SAT	UNSAT	SAT	UNSAT	SAT	UNSAT	SAT	UNSAT	SAT	UNSAT
12	0.0	-	0.0	-	0.0	-	0.0	-	0.0	-
13	0.1	-	0.0	-	0.0	-	0.0	-	0.0	-
14	0.1	TO	0.2	24.2	0.1	16.8	0.1	22.1	0.2	20.3
15	0.1	-	0.1	-	0.4	-	1.2	-	0.3	-
16	0.2	TO	0.3	1903.3	0.5	522.9	0.2	1053.6	0.5	648.7
17	0.4	TO	0.4	0.6	2.8	0.3	1.8	0.3	1.5	0.8
18	0.4	TO	2.2	4.9	5.2	2.4	2.5	2.2	20.4	3.0
19	0.1	-	0.1	-	0.1	-	0.1	-	0.2	-
20	37.9	-	54.8	-	30.3	-	48.9	-	68.9	-
21	1.1	TO	7.4	TO	0.8	TO	5.6	TO	4.6	TO
22	9.2	TO	16.2	TO	27.3	TO	50.8	TO	14.5	TO
23	9.2	TO	63.4	TO	125.3	TO	17.0	TO	70.8	TO
24	3.7	-	106.5	-	103.2	-	47.2	-	83.0	-
25	235.8	TO	229.0	TO	264.7	TO	440.6	TO	492.5	TO
26	987.4	TO	614.4	157.9	548.7	114.3	286.2	78.5	539.2	86.5
27	TO	TO	634.1	TO	TO	503.6	1000.1	932.6	2644.0	545.0
28	TO	TO	3193.3	TO	2077.0	TO	4746.6	TO	3301.8	TO
29	TO	-	TO	-	TO	-	TO	-	TO	-
30	TO	-	TO	-	TO	-	TO	-	TO	-
31	392.1	TO	849.0	TO	1506.5	TO	TO	TO	TO	TO
32	49.7	-	TO	-	TO	-	591.7	-	TO	-

Table 1.   Runtimes for undirected graphs with symmetry-breaking

$|V|$	none		$lex^{\leq}_{i,i+1}$		$lex^{\leq}_{i,j}$	
	SAT	UNSAT	SAT	UNSAT	SAT	UNSAT
6	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	0.5	0.0	0.1	0.0	0.0
8	0.0	7.6	0.0	0.5	0.0	0.3
9	0.0	154.6	0.0	2.9	0.0	1.6
10	0.0	TO	0.0	25.2	0.0	7.4
11	0.0	TO	0.0	382.2	0.0	72.4
12	0.0	TO	0.0	TO	0.0	TO

Table 2.   Runtimes for directed graphs

$|V|$	Shlyakhter		$^A lex^{\leq}_{i,i+1}$		$^A lex^{\leq}_{i,j}$	
	SAT	UNSAT	SAT	UNSAT	SAT	UNSAT
10	0.1	0.1	0.1	0.1	0.1	0.1
11	0.3	0.3	0.1	0.2	0.1	0.2
12	0.5	0.9	0.3	0.6	0.3	0.8
13	0.6	3.6	0.9	0.9	1.1	1.3
14	2.2	8.8	1.6	3.1	2.8	4.4
15	6.7	19.1	3.0	10.9	9.2	14.8
16	2.8	75.7	5.7	25.9	6.8	28.3
17	8.9	229.8	25.2	90.8	14.8	83.4
18	69.9	543.0	41.6	125.8	30.0	176.6
19	220.7	1024.3	56.3	373.1	9.4	323.3
20	724.3	3308.6	91.5	788.3	19.1	742.0
21	203.0	TO	395.3	2089.3	244.2	2120.8
22	TO	TO	184.2	TO	91.9	3327.5
23	531.0	TO	370.5	TO	438.8	TO
24	2477.5	TO	563.5	TO	590.6	TO

Table 3.   Runtimes for directed acyclic graphs

Ithaca, NY, USA, September 9–13, 2002 Proceedings, volume 2470 of *Lecture Notes in Computer Science*, pages 93–108. Springer-Verlag, 2002.

[8] T. Geffner and H. Geffner. Compact policies for non-deterministic fully observable planning as SAT. In *ICAPS 2018. Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling*, pages 88–96. AAAI Press, 2018.

[9] M. J. H. Heule. Optimal symmetry breaking for graph problems. *Mathematics in Computer Science*, 13:533–548, 2019.

[10] A. Itzhakov and M. Codish. Breaking symmetries in graph search with canonizing sets. *Constraints*, 21:357–374, 2016.

[11] M. Kirchweger and S. Szeider. SAT modulo symmetries for graph generation. In *27th International Conference on Principles and Practice of Constraint Programming, CP*, volume 210 of *LIPIcs*, pages 34:1–34:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[12] A. Miller and P. Prosser. Diamond-free degree sequences. *arXiv preprint arXiv:1208.0460*, 2012.

[13] M. Nakao, H. Murai, and M. Sato. A method for order/degree problem based on graph symmetry and simulated annealing with MPI/OpenMP parallelization. In *Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region*, HPC Asia 2019, pages 128–137. Association for Computing Machinery, 2019.

[14] B. Pandey and J. Rintanen. Planning for partial observability by SAT and graph constraints. In *ICAPS 2018. Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling*, pages 190–198. AAAI Press, 2018.

[15] J.-F. Puget. Automatic detection of variable and value symmetries. In *Principles and Practice of Constraint Programming - CP 2005 11th International Conference, CP 2006, Sitges, Spain, October 2005, Proceedings*, volume 3709 of *Lecture Notes in Computer Science*, pages 475–489. Springer-Verlag, 2005.

[16] I. Shlyakhter. Generating effective symmetry-breaking predicates for search problems. *Discrete Applied Mathematics*, 155(12):1539–1548, 2007.

[17] P. H. Starke. Reachability analysis of Petri nets using symmetries. *Journal of Mathematical Modelling and Simulation in Systems Analysis*, 8 (4/5):293–303, 1991.

[18] T. Walsh. General symmetry breaking constraints. In *Principle and Practice of Constraint Programming - CP 2006 12th International Conference, CP 2006, Nantes, France, September 25-29, 2006, Proceedings*, volume 4204 of *Lecture Notes in Computer Science*, pages 650–664. Springer-Verlag, 2006.