Constraints (2016) 21:357–374
DOI 10.1007/s10601-016-9244-z

Breaking symmetries in graph search with canonizing sets

Avraham Itzhakov[1] · **Michael Codish**[1]

Published online: 2 April 2016

Abstract There are many complex combinatorial problems which involve searching for an undirected graph satisfying given constraints. Such problems are often highly challenging because of the large number of isomorphic representations of their solutions. This paper introduces effective and compact, complete symmetry breaking constraints for small graph search. Enumerating with these symmetry breaks generates all and only non-isomorphic solutions. For small search problems, with up to 10 vertices, we compute instance independent symmetry breaking constraints. For small search problems with a larger number of vertices we demonstrate the computation of instance dependent constraints which are complete. We illustrate the application of complete symmetry breaking constraints to extend two known sequences from the OEIS related to graph enumeration. We also demonstrate the application of a generalization of our approach to fully-interchangeable matrix search problems.

Keywords Graph search problems · Symmetry breaking · SAT solving

1 Introduction

Graph search problems are about the search for a graph which satisfies a given set of constraints, or to determine that no such graph exists. Often graph search problems are about the search for the set of all graphs, modulo graph isomorphism, that satisfy the given

Supported by the Israel Science Foundation, grant 182/13.

✉ Michael Codish
mcodish@cs.bgu.ac.il

Avraham Itzhakov
itzhakoa@cs.bgu.ac.il

[1] Department of Computer Science, Ben-Gurion University of the Negev, Beer Sheva, Israel

constraints. Graph search problems are typically invariant under graph isomorphism. Namely, if G is a solution, then any graph obtained by permuting the vertices of G is also a solution. When seeking solutions, the size of the search space is significantly reduced if symmetries are eliminated. The search space can be explored more efficiently when avoiding paths that lead to symmetric solutions and avoiding also those that lead to symmetric non-solutions.

One common approach to eliminate symmetries is to introduce symmetry breaking constraints [11, 29, 33, 34] which rule out isomorphic solutions thus reducing the size of the search space while preserving the set of solutions. Ideally, a symmetry breaking constraint is satisfied by a single member of each equivalence class of solutions, thus drastically restricting the search space. However, computing such symmetry breaking constraints is, most likely, intractable in general [11]. In practice, symmetry breaking constraints typically rule out some, but not all of the symmetries in the search and, as noted in the survey by Walsh [35], often a few simple constraints rule out most of the symmetries.

Shlyakhter [33] notes that the core difficulty is to identify a symmetry-breaking predicate which is both *effective* (rules out a large portion of the search space) and *compact* (so that checking the additional constraints does not slow down the search). In [10], Codish et al. introduce a symmetry breaking constraint for graph search problems. Their constraint is compact, with size polynomial in the number of graph vertices, and shown to be effective but it does not eliminate all of the symmetries in the search.

There is a large body of research that concerns identifying symmetries in a given graph. In this setting, finding symmetries is about detecting graph automorphisms. A typical application is in the context of SAT solving as described for example in [2, 4, 16–18]. In this paper the setting is different as the graph is not given but rather is the subject of the search problem.

In this paper we adopt the following terminology. Symmetry breaking constraints that break all of the symmetries, or more precisely, that are satisfied by exactly one solution in each symmetry class, are called *complete*. Symmetry breaking constraints which are sound i.e, satisfied by at least one solution in each symmetry class, but not complete are called *partial*. If a symmetry breaking constraint is satisfied exactly by the canonical representatives of the symmetry classes, it is called *canonizing*. Note that canonizing symmetry breaking constraints are also complete.

Computing all solutions to a graph search problem with partial symmetry breaking constraints is a two step process. First one generates the set S of solutions to the constraints, and then one applies a graph isomorphism tool, such as `nauty` [22] to reduce S modulo isomorphism. Often, the number of solutions in the first step is very large and then this method may fail to generate the initial set of solutions.

This paper presents a methodology to compute small sets of static canonizing symmetry breaking constraints for "small" graph search problems. Consider for example the search for a graph with $n = 10$ vertices. The search space consists of 2^{45} graphs, whereas, there are only 12 005 168 such graphs modulo isomorphism (see sequence `A000088` of the OEIS [28]). In theory, to break all symmetries one could construct a symmetry breaking constraint that considers all $10! = 3{,}628{,}800$ permutations of the vertices. We will show how to construct a compact canonizing symmetry breaking constraint for graph search problems on 10 vertices using only 7853 permutations.

Our approach can be applied, in the terminology of [3], both in an "instance independent" fashion and "instance dependent". When "instance independent", it generates canonizing symmetry breaking constraints for any graph search problem and in this setting

it applies to break all symmetries in graph search problems on up to 10 vertices. When "instance dependent", it generates canonizing symmetry breaking constraints which apply to break symmetries in larger graphs which are solutions of a given graph search problem. These symmetry breaking constraints are typically smaller and easier to compute than the corresponding "instance independent" constraints. We illustrate the application of complete symmetry breaking constraints, both instance independent and instance dependent, to extend two known sequences from the OEIS related to graph enumeration.

We also observe that the derived symmetry constraints are "solver independent". They can be applied in conjunction with any constraint solver to restrict the search to canonical solutions of a given search problem.

The rest of this paper is structured as follows. Section 2 provides a motivating example. Section 3 presents preliminary definitions and notation. Section 4 describes how we compute complete and canonizing symmetry breaking constraints. First, in Section 4.2, for instance independent graph search problems. Then, in Section 4.5, for a given graph search problem. Section 5 demonstrates a generalization of our approach to matrix search problems and illustrates its impact when solving the Equi-distant Frequency Permutation Array problem (EFPA). Section 7 concludes.

2 A motivating example

A classic example of a graph search problem relates to the search for Ramsey graphs [30]. The graph $R(s, t; n)$ is a simple graph with n vertices, no clique of size s, and no independent set of size t. Figure 1 illustrates a $R(3, 3; 5)$ graph. The graph contains no 3-clique and no 3-independent set. A Ramsey (s, t)-graph is a $R(s, t; n)$ graph for some n. The set of all $R(s, t; n)$ graphs, modulo graph isomorphism, is denoted $\mathcal{R}(s, t; n)$. Ramsey Theory tells us that there are only a finite number of Ramsey (s, t)-graphs for each s and t, but finding all such graphs, or even determining the largest n for which they exist, is a famously difficult problem. It is unknown, for example, if there exists a $R(5, 5; 43)$ graph and the set $\mathcal{R}(4, 5; 24)$ has yet to be been fully determined, although 350,904 non-isomorphic graphs are known to belong to $\mathcal{R}(4, 5; 24)$.

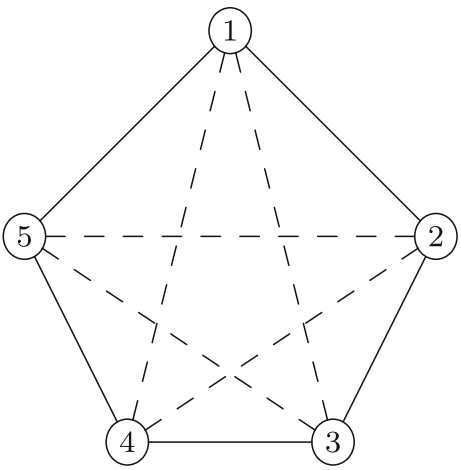

Fig. 1 A $R(3, 3; 5)$ Ramsey graph: edges denoted by solid lines and non-edges by *dashed*

Solving the graph search problem to find all $R(3,4;8)$ graphs without any symmetry breaking constraint results in a set of 17,640 graphs. Applying `nauty` [22] to these solutions identifies precisely 3 solutions modulo graph isomorphism. Introducing a partial symmetry breaking constraint as described in [9] in the search to enumerate all $R(3,4;8)$ graphs computes only 11 graphs in a fraction of the time required to compute the full set of solutions. These too can then be reduced applying `nauty` to obtain the 3 solutions. Application of a complete symmetry breaking constraint as proposed in this paper results in the exact set of 3 non-isomorphic solutions.

3 Preliminaries

Throughout this paper we consider finite and simple graphs (undirected with no self loops). The set of simple graphs on n nodes is denoted $\mathcal{G}_n$. We assume that the vertex set of a graph, $G = (V, E)$, is $V = \{1, \ldots, n\}$ and represent G by its $n \times n$ adjacency matrix A defined by

$$A_{i,j} = \begin{cases} 1 \text{ if } (i,j) \in E \\ 0 \text{ otherwise} \end{cases}$$

An n-vertices graph search problem is a predicate φ on an $n \times n$ matrix A of Boolean variables $A_{i,j}$; and a solution to a graph search problem φ is a satisfying assignment (to the variables in A) of the conjunction $\varphi(A) \wedge adj^n(A)$ where $adj^n(A)$ states that A is an $n \times n$ adjacency matrix:

$$adj^n(A) = \underbrace{\bigwedge_{1 \le i \le n} (\neg A_{i,i})}_{(a)} \quad \wedge \quad \underbrace{\bigwedge_{1 \le i < j \le n} (A_{i,j} \leftrightarrow A_{j,i})}_{(b)} \tag{1}$$

In Constraint (1), the left part (a) states that there are no self loops and the right part (b) states that the edges are undirected. The set of solutions of a graph search problem is denoted $sol(\varphi)$ and when we wish to make the variables explicit we write $sol(\varphi(A))$. The set $sol(\varphi)$ is typically viewed as a set of graphs. Note that $sol(true) = \mathcal{G}_n$. The following presents two examples of graph search problems which we will refer to in rest of the paper.

Example 1 The Ramsey graph $R(s,t;n)$ is a simple graph with n vertices, no clique of size s, and no independent set of size t. The set of all $R(s,t;n)$ graphs, modulo graph isomorphism, is denoted $\mathcal{R}(s,t;n)$. The search for a Ramsey graph is a graph search problem where we take the following $\varphi_{R(s,t;n)}$ as the predicate φ. Here we denote by $\wp_s[n]$ (respectively $\wp_t[n]$) the set of subsets of size s (respectively t) of $\{1, \ldots, n\}$. The left conjunct (a) states that there is no clique of size s in the graph, and the right conjunct (b) that there is no independent set of size t.

$$\varphi_{(s,t;n)}(A) = \underbrace{\bigwedge_{I \in \wp_s[n]} \bigvee \left\{ \neg A_{i,j} \,\middle|\, \begin{matrix} i,j \in I, \\ i < j \end{matrix} \right\}}_{(a)} \quad \wedge \quad \underbrace{\bigwedge_{I \in \wp_t[n]} \bigvee \left\{ A_{i,j} \,\middle|\, \begin{matrix} i,j \in I, \\ i < j \end{matrix} \right\}}_{(b)} \tag{2}$$

Example 2 A graph is claw-free if it does not contain the complete bipartite graph $K_{1,3}$ (sometimes called a "claw") as a subgraph. The claw free graph search problem is

formalized by taking the following $\varphi_{cf(n)}$ as the predicate φ. Each clause in the conjunction expresses for i, j, k, ℓ that there is no subgraph $K_{1,3}$ between $\{i\}$ and $\{j, k, \ell\}$.

$$\varphi_{cf(n)}(A) = \bigwedge \left\{ \begin{array}{l} \neg A_{i,j} \vee \neg A_{i,k} \vee \neg A_{i,\ell} \\ \vee A_{j,k} \vee A_{k,\ell} \vee \vee A_{j,\ell} \end{array} \middle| \begin{array}{l} 1 \leq i \leq n,\ i \neq j,\ i \neq k, \\ i \neq \ell,\ 1 \leq j < k < \ell \leq n \end{array} \right\} \tag{3}$$

The set of permutations $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ is denoted S_n. For $G = (V, E) \in \mathcal{G}_n$ and $\pi \in S_n$, we define $\pi(G) = \{V, \{(\pi(u), \pi(v)) | (u, v) \in E)\}$. Permutations act on adjacency matrices in the natural way: If A is the adjacency matrix of a graph G, then $\pi(A)$ is the adjacency matrix of $\pi(G)$ obtained by simultaneously permuting with π the rows and columns of A. We adopt the tuple notation $[\pi(1), \ldots, \pi(n)]$ for a permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$.

Two graphs $G_1, G_2 \in \mathcal{G}_n$ are isomorphic, denoted $G_1 \approx G_2$, if there exists a permutation $\pi \in S_n$ such that $G_1 = \pi(G_2)$. Sometimes we write $G_1 \approx_\pi G_2$ to emphasize that π is the permutation such that $G_1 = \pi(G_2)$. For sets of graphs H_1, H_2, we say that $H_1 \approx H_2$ if for every $G_1 \in H_1$ (likewise in H_2) there exists $G_2 \in H_2$ (likewise in H_1) such that $G_1 \approx G_2$.

We consider an ordering on graphs, defined viewing their adjacency matrices as strings. Because adjacency matrices are symmetric with zeroes on the diagonal, it suffices to focus on the upper triangle parts of the matrices [8].

Definition 1 (**ordering graphs**) Let $G_1, G_2 \in \mathcal{G}_n$ and let s_1, s_2 be the strings obtained by concatenating the rows of the upper triangular parts of their corresponding adjacency matrices A_1, A_2 respectively. Then, $G_1 \preceq G_2$ if and only if $s_1 \preceq_{lex} s_2$. We also write $A_1 \preceq A_2$.

One way to define the canonical representation of a graph is to take the smallest graph in the $\preceq$ order in each equivalence class of isomorphic graphs [31]. In this paper we follow this definition for canonicity.

Definition 2 (**canonicity**) Let $G \in \mathcal{G}_n$ be a graph, $\Pi \subseteq S_n$, and denote the predicate $\min_\Pi(G) = \bigwedge \left\{ G \preceq \pi(G) \,\middle|\, \pi \in \Pi \right\}$. We say that G is canonical if $\min_{S_n}(G)$. We say that Π is canonizing if $\forall_{G \in \mathcal{G}_n}. \min_\Pi(G) \leftrightarrow \min_{S_n}(G)$.

Observe that in Definitions 1 and 2, the order is defined on given graphs. Often, we consider the same relation, but between adjacency matrices that contain propositional variables (representing unknown graphs, as in the case for graph search problems). Then, the expressions $A_1 \preceq A_2$ and $\min_\Pi(A)$ are viewed as a Boolean constraints on the variables in the corresponding matrices.

Example 3 It turns out that $\Pi = \left\{ [2, 1, 3, 4], [1, 3, 2, 4], [1, 2, 4, 3] \right\}$ is canonizing for $\mathcal{G}_4$. Namely, with only three permutations we express the information present in all $4! = 24$ elements of S_4. So for instance, the graph G depicted in Fig. 2a is canonical because it is smaller than its three permutations with respect to Π detailed as Fig. 2b, c, and d. We come back to elaborate on why Π is canonizing in Example 4.

Definition 3 (**symmetry break**) Let $\varphi(A)$ be a n-vertices graph search problem and $\sigma(A)$ a propositional formula on the variables in A. We say that σ is a symmetry break for

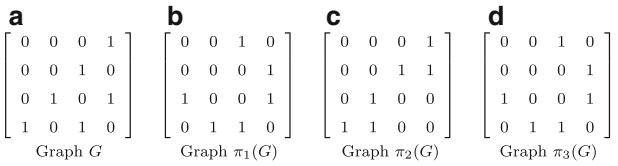

Fig. 2 A graph and its isomorphic representations according to: $\pi_1 = [2, 1, 3, 4]$, $\pi_2 = [1, 3, 2, 4]$, and $\pi_3 = [1, 2, 4, 3]$

φ if $sol(\varphi(A)) \approx sol(\varphi(A) \wedge \sigma(A))$. If the graphs of $sol(\varphi(A) \wedge \sigma(A))$ are mutually non-isomorphic then we say that σ is complete. Otherwise we say that σ is partial. If the graphs of $sol(\varphi(A) \wedge \sigma(A))$ are canonical then we say that σ is canonizing.

Lemma 1 *Let Π be a canonizing set of permutations for graphs of size n. Then* $\min_\Pi$ *is a canonizing symmetry break for any graph search problem on n vertices.*

Proof Let A be a solution to a graph search problem on n vertices and let Π be a canonizing set for graphs with n vertices. In order to prove that min_Π is a canonizing symmetry break it is sufficient to show that only the canonical member in $iso(A) = \{\pi(A) | \forall \pi \in S_n\}$ satisfies min_Π. Π is a canonizing set thus by definition $\forall G \in \mathcal{G}_n : min_\Pi(G) \leftrightarrow min_{S_n}(G)$. Since only the canonical graph in $iso(A)$ satisfies min_{S_n} it follows that it is the only one which satisfies min_Π. □

Corollary 1 $\min_{S_n}(A)$ *is a canonizing symmetry break for any graph search problem on n vertices.*

Example 4 Consider the canonizing set Π from Example 3 and the following 4×4 adjacency matrix A:

$$A = \begin{bmatrix} 0 & a & b & c \\ a & 0 & d & e \\ b & d & 0 & f \\ c & e & f & 0 \end{bmatrix} \qquad \Pi = \left\{ \begin{array}{l} [2, 1, 3, 4], \\ [1, 3, 2, 4], \\ [1, 2, 4, 3] \end{array} \right\}$$

Then, by Definition 2 and Lemma 1,

$$\min_\Pi(A) = (abcdef \preceq_{lex} adebcf) \wedge (abcdef \preceq_{lex} bacdfe) \wedge (abcdef \preceq_{lex} acbdef)$$

and this simplifies using properties of lexicographic orderings to:

$$\min_\Pi(A) = (bc \preceq_{lex} de) \wedge (ae \preceq_{lex} bf) \wedge (bd \preceq_{lex} ce)$$

To verify that Π is indeed canonizing one should consider each of the permutations in $\pi \in S_4 \setminus \Pi$ and prove that $\min_\Pi(A) \Rightarrow A \preceq \pi(A)$ where A is the variable matrix detailed above. For example, when $\pi = [2, 1, 4, 3]$, $A \preceq \pi(A)$ means $abcdef \preceq_{lex} aedcbf$ which simplifies to $bc \preceq_{lex} ed$ and we need to show that $(bc \preceq_{lex} de) \wedge (ae \preceq_{lex} bf) \wedge (bd \preceq_{lex} ce) \Rightarrow bc \preceq_{lex} ed$. This is not difficult to check.

Clearly, for any set of permutations $\Pi \subset S_n$ the predicate $\min_\Pi$ is a partial symmetry break for graph search problems. In [9], Codish et al. introduce the following symmetry

break for graph search problems where A_i denotes the i^{th} row of the adjacency matrix A and $\preceq_{\{i,j\}}$ denotes the lexicographic comparison on strings after removing their i^{th} and j^{th} elements.

Definition 4 (**lexicographic symmetry break**, [9]) Let A be an $n \times n$ adjacency matrix. Then,

$$\mathsf{sb}^*_\ell(A) = \bigwedge_{1 \le i < j \le n} A[i] \preceq_{\{i,j\}} A[j]$$

It is not difficult to observe that sb^*_ℓ is equivalent to the predicate $\min_{\Pi}$ where $var\Pi$ is the set of permutations that swap a single pair (i, j) with $1 \le i < j \le n$.

The experimental setting In this paper all computations are performed using the Glucose 4.0 SAT solver [5]. Encodings to CNF are obtained using the finite-domain constraint compiler BEE [26]. BEE facilitates applications to find a single (first) solution, or to find all solutions for a constraint, modulo a specified set of variables. When solving for all solutions, BEE iterates with the SAT solver, adding so called *blocking clauses* each time another solution is found. This technique, originally due to McMillan [25], is simplistic but suffices for our purposes. All computations were performed on a cluster with a total of 228 Intel E8400 cores clocked at 2 GHz each, able to run a total of 456 parallel threads. Each of the cores in the cluster has computational power comparable to a core on a standard desktop computer. Each SAT instance is run on a single thread.

4 Canonizing symmetry breaks

The observation made in Example 3: that a canonizing set for graphs with n vertices can be much smaller than $n!$, motivates us to seek "small" canonizing sets that might be applied to introduce canonizing symmetry breaking constraints for graph search problems. First, we describe the application of this approach to compute relatively small *instance independent* canonizing sets, which induce general purpose symmetry breaks that can be used for any graph search problem. We compute these sets for graphs with $n \le 10$ vertices. We illustrate their application when breaking all symmetries in the search for Ramsey and claw-free graphs.

Second, we apply our methods to compute *instance dependent* canonizing sets which are computed for a given graph search problem. Namely, these sets promise that only non-isomorphic solutions will be generated when enumerating all solutions for the given graph search problem that satisfy its corresponding canonizing symmetry breaks. However, these sets are not necessarily canonizing for other graph search problem. We show that such canonizing sets can be computed for larger graphs (compare to instance independent canonizing sets) and their usage is illustrated to enumerate all non-isomorphic highly irregular graphs up to 20 vertices.

4.1 Computing canonizing sets

To compute a canonizing set of permutations for graph search problem φ on n vertices we start with some initial set of permutations Π (for simplicity, assume that $\Pi = \emptyset$). Then,

incrementally apply the step specified in lines 2–3 of Algorithm 1, as long as the stated condition holds.

Algorithm 1 Compute Canonizing Set

1: **procedure** COMPUTE-CANONIZING-SET(Π, φ)
2: **while** $\exists G \in sol(\varphi)\ \exists \pi \in S_n$ such that $\min_\Pi(G)$ and $\pi(G) \prec G$ **do**
3: $\Pi = \Pi \cup \{\pi\}$
4: **end while**
5: **return** Π
6: **end procedure**

Lemma 2 *Algorithm 1 terminates and returns a canonizing set Π for the graph search problem φ.*

Proof Each step in the algorithm adds a permutation (at Line 3) and the number of permutations is bound. When the algorithm terminates with Π then for $G \in sol(\varphi)$, if $\min_\Pi(G)$ holds then there is no $\pi \in S_n$ such that $\pi(G) \prec G$. So, $G \preceq \pi(G)$ for all $\pi \in S_n$ and therefore $\min_{S_n}(G)$ holds. □

Drawing on the discussion in [6, 11, 20] we do not expect to find a polynomial time algorithm to compute a canonical (or any other complete) symmetry breaking constraint for graph search problems based on Definition 2. Thus it is also unlikely to find an efficient implementation of Algorithm 1. Our implementation of Algorithm 1 is based on a SAT encoding. We repeatedly apply a SAT solver to find a counter example permutation which shows that Π is not a canonizing set yet and add it to Π, until an UNSAT result is obtained. In the implementation of the algorithm, care is taken to use a single invocation of the SAT solver so that the iterated calls to the solver are incremental. The constraint model used is depicted as Fig. 3 where A, B denote $n \times n$ matrices of propositional variables and π denotes a length n vector of integer variables. Constraint 4 specifies that the parameter π is a permutation on $\{1, \ldots, n\}$. Each element of the vector is a value $1 \leq \pi_i \leq n$ and the elements are all different. Constraint 5 specifies that the parameters A, B represent isomorphic graphs via the parameter π. Constraint 6 specifies the condition of the while loop (line 2) of Algorithm 1: A is restricted to be a solution to the given graph search problem φ, A and B are constrained $B = \pi(A)$ to be isomorphic adjacency matrices (see Constraint (1)) via the permutation π. The constraint $\min_\Pi(A)$ is imposed and also $A \succ B$. If $alg_1^n(\Pi, \varphi)$ is satisfiable, then the permutation π is determined by the satisfying assignment and added to Π as specified in (line 3) of Algorithm 1.

$$perm^n(\pi) = \bigwedge_{1 \leq i \leq n} 1 \leq \pi_i \leq n \ \wedge \ \texttt{allDifferent}(\pi) \tag{4}$$

$$iso^n(A, B, \pi) = \bigwedge_{1 \leq i,j \leq n} \left(B_{i,j} \Leftrightarrow \bigvee_{1 \leq i',j' \leq n} \left(\pi_{i'} = i \wedge \pi_{j'} = j \wedge A_{i',j'}\right)\right) \tag{5}$$

$$\begin{aligned} alg_1^n(\Pi, \varphi) = {} & adj^n(A) \ \wedge \ adj^n(B) \ \wedge \ perm^n(\pi) \ \wedge \ iso^n(A, B, \pi) \ \wedge \\ & \wedge \min_\Pi(A) \ \wedge \ A \succ B \ \wedge \ \varphi(A) \end{aligned} \tag{6}$$

Fig. 3 Constraints for Algorithm 1 where A and B are $n \times n$ Boolean matrices and $\pi = \langle \pi_1, \ldots, \pi_n \rangle$ is a vector of integer variables (with domain $\{1, \ldots, n\}$)

We say that a canonizing set Π of permutations is redundant if for some $\pi \in \Pi$ the set $\Pi \setminus \{\pi\}$ is also canonizing. Algorithm 1 may compute a redundant set. For example, if a permutation added at some point becomes redundant in view of permutations added later. Algorithm 2 iterates on the elements of a canonizing set to remove redundant permutations.

Algorithm 2 Reduce method

```
1: procedure REDUCE(Π, φ)
2:     for each π ∈ Π do
3:         if ∀G ∈ sol(φ): min_{Π\{π}}(G) ⇒ G ⪯ π(G) then
4:             Π = Π \ {π}
5:         end if
6:     end for
7:     return Π
8: end procedure
```

Lemma 3 *If Π is a canonizing set for the graph search problem φ, then so is $Reduce(\Pi, \varphi)$ computed by Algorithm 2.*

Proof Let Π_i be the set obtained after considering the i^{th} permutation in Line 2 of Algorithm 2. The initial set Π_0 is the input to the algorithm. We prove that $\min_{\Pi_i} \leftrightarrow \min_{\Pi_{i+1}}$ and conclude that $\min_{\Pi} \leftrightarrow \min_{Reduce(\Pi)}$. If no permutation was removed in step i then $\Pi_{i+1} = \Pi_i$ and trivially $\min_{\Pi_i} \leftrightarrow \min_{\Pi_{i+1}}$. Otherwise $\Pi_{i+1} = \Pi_i \setminus \{\pi\}$ for a permutation π which satisfies $\forall G \in sol(\varphi)$: $\min_{\Pi_{i+1}}(G) \Rightarrow G \preceq \pi(G)$. Thus π is implied by the permutations in Π_i and can be removed. Therefore $\min_{\Pi_{i+1}}(G) \leftrightarrow \min_{\Pi_i}(G)$. □

Our implementation of Algorithm 2 is based on a SAT encoding. The key is in the encoding for the test in Line 3. Here, for the given Π and $\pi \in \Pi$, we encode the constraint

$$alg_2^n(\Pi, \varphi) = \underbrace{adj^n(A)}_{(a)} \ \wedge \ \underbrace{\varphi(A) \ \wedge \ min_{\Pi\setminus\{\pi\}}(A) \ \wedge \ \pi(A) \succ A}_{(b)} \tag{7}$$

where the left part (a) specifies that A is the $n \times n$ adjacency matrix of some graph (see Constraint (1)), and the right part (b) is the negation of the condition in Line 3. If this constraint is unsatisfiable then π is redundant and removed from Π.

4.2 Instance independent symmetry breaks

Observe that if $\varphi = true$ then $sol(\varphi) = \mathcal{G}_n$. Applying Algorithm 1 to compute `Compute-Canonizing-Set(`Π`, true)` generates canonizing symmetry breaks which apply for any graph search problem on n vertices (i.e instance independent). This is true for any set of permutations Π but for simplicity assume $\Pi = \emptyset$.

Table 1 describes the computation of irredundant instance independent canonizing permutation sets for $n \leq 10$ by application of Algorithms 1 and 2. The corresponding permutation sets can be obtained from http://www.cs.bgu.ac.il/~mcodish/Papers/Tools/canonizingSets. The first 3 columns indicate the number of graph vertices, n, the number of permutations on n, and the number of non-isomorphic graphs on n vertices as specified by sequence `A000088` of the OEIS [28]. The forth and fifth columns indicate the size of

Table 1 Computing irredundant canonizing sets of permutations for $n \leq 10$

			Algorithm 1		Algorithm 2	
n	$n!$	can. graphs	time (sec.)	can. set	time (sec.)	red. set
3	6	4	0.02	3	0.01	**2**
4	24	11	0.02	7	0.01	**3**
5	120	34	0.05	27	0.05	**7**
6	720	156	0.35	79	1.27	**13**
7	5 040	1 044	1.92	223	11.27	**37**
8	40 320	12 346	27.61	713	317.76	**135**
9	362 880	274 668	1 108.13	4 125	7 623.20	**842**
10	3 628 800	12 005 168	9.82 hr.	20 730	84 hr.	**7 853**

the canonical set of permutations computed using Algorithm 1 and the time to perform this computation. Columns six and seven are the size of the reduced canonical set of permutations after application of Algorithm 2 and the corresponding computation time. Column seven is set in boldface. These numbers present the relatively small size of the computed canonizing sets in comparison to the value of $n!$. Using the symmetry breaks derived from these sets we have generated the sets of all non-isomorphic graphs with up to 10 vertices and verified that their numbers correspond to those in column three. These are computed by solving the conjunction of Constraint (1) with the corresponding symmetry breaking predicate $\min_{\Pi}$ the computation of which is described in Table 1.

The numbers in Table 1 also indicate the limitation of complete symmetry breaks which apply to all graphs. We do not expect to succeed to compute a canonizing set of permutations for $n = 11$ and even if we did succeed, we expect that the number of constraints that would then need be added in applications would be too large to be effective.

4.3 Computing Ramsey graphs with canonizing symmetry breaks

Recall Example 1 where we introduce the graph search problem for Ramsey graphs. Table 2 describes the computation of all $R(4, 4; n)$ graphs for $n \leq 10$ using a SAT solver. The table compares two configurations: one using the partial symmetry breaking predicate sb^*_ℓ defined in [9] and the other using a canonizing symmetry break $\min_{\Pi}$ where Π is the canonizing set of permutations, the computation of which is described in Table 1. For each configuration we detail the size of the SAT encoding (clauses and variables), the time in seconds (except where indicated in hours) to find all solutions using a SAT solver, and the number of solutions found. Observe that the encodings using the canonizing symmetry breaks are much larger. However the sat solving time is much smaller. For $n = 10$ the configuration with sb^*_ℓ requires more than 33 hours where as the configuration using $\min_{\Pi}$ completes in under 7 hours. Finally note that the computation with $\min_{\Pi}$ computes the precise number of solutions modulo graph isomorphism as detailed for example in [24]. These are the numbers in the rightmost column set in boldface. The solutions computed using sb^*_ℓ contain many isomorphic solutions which need to be subsequently removed using `nauty` or a similar tool. One might argue that the real cost in applying the complete symmetry breaks should include their computation. To this end we note that these are general symmetry breaking predicates applicable to any graph search problem, and they are precomputed once.

Table 2 Computing $|\mathcal{R}(4, 4; n)|$ with canonizing symmetry breaking and sb^*_ℓ

	Partial sym. break sb^*_ℓ				Canonizing sym. break			
n	clauses	vars	sat (sec.)	sols	clauses	vars	sat (sec.)	sols
4	22	10	0.01	9	17	5	0.00	**9**
5	80	24	0.01	33	235	55	0.01	**24**
6	195	48	0.02	178	315	72	0.01	**84**
7	390	85	0.12	1 478	1 395	286	0.05	**362**
8	690	138	4.91	16 919	10 885	2 177	1.69	**2 079**
9	1 122	210	745.72	227 648	89 877	17 961	144.4	**14 701**
10	1 715	304	33.65 hr.	2 891 024	1 406 100	281 181	6.56 hr.	**103 706**

4.4 Computing claw-free graphs with canonizing symmetry breaks

Recall Example 2 where we introduce the graph search problem for claw-free graphs. The number of claw-free graphs for $n \leq 9$ vertices is detailed as sequence `A086991` on the OEIS [28]. Table 3 describes the search for claw free graphs as a graph search problem. Then we use a SAT solver to compute the set of all claw free graphs on n vertices. We illustrate that using canonizing symmetry breaks, and the results detailed in Table 1, we can compute the set of all claw free graphs modulo graph isomorphism for $n \leq 10$ thus computing a new value for sequence `A086991`. We comment that after this value for $n = 10$ was added to the OEIS, Falk Hüffner added further values for $10 < n \leq 15$. The column descriptions are the same as those for Table 2. For each configuration we detail the size of the SAT encoding (clauses and variables), the time in seconds to find all solutions using a SAT solver, and the number of solutions found. For this example the computation with a complete symmetry break is more costly, but it does return the precise set of canonical graphs. The sequence in the right column are set in boldface. For $n \leq 9$, these are the numbers of claw-free graphs as detailed in sequence `A086991` of the OEIS [28]. It is no coincidence that the number of variables indicated in the columns of Tables 2 and 3 are almost identical. These pertain to the variables in the adjacency graph and those introduced to express the instance independent symmetry breaks.

Table 3 Computing claw-free graphs with canonizing symmetry breaking and sb^*_ℓ

	Partial sym. break sb^*_ℓ				Canonizing sym. break			
n	clauses	vars	sat (sec.)	sols	clauses	vars	sat (sec.)	sols
4	24	10	0.01	10	19	9	0.01	**10**
5	90	24	0.01	32	245	55	0.01	**26**
6	225	48	0.01	143	345	72	0.01	**85**
7	460	85	0.04	819	1 465	286	0.03	**302**
8	830	138	0.86	5 559	11 025	2 177	1.08	**1 285**
9	1 374	210	28.72	42 570	90 129	17 961	75.23	**6 170**
10	2 135	304	2 352.37	368 998	1 406 520	281 181	8797.23	**34 294**

4.5 Instance dependent canonizing symmetry breaks

A canonizing set for a specific graph search problem φ is typically much smaller than a general canonizing set as the constraints in φ restrict the solution structure and hence also the symmetries within the solution space. We call such a set *instance dependent*. In practice we can often compute instance dependent canonizing sets for larger graphs with $n > 10$ vertices. For a given graph search problem φ, let us denote by Π_φ the canonizing set of permutations obtained from `Compute-Canonizing-Set(∅, φ)` of Algorithm 1. Solutions of φ obtained with the induced symmetry break predicate $\min_{\Pi_\varphi}$ are guaranteed to be pairwise non-isomorphic.

In this section we demonstrate the application of instance dependent canonizing sets. Here we consider a search problem for which we seek a graph that has a particular given degree sequence.

A degree sequence is a monotonic non-increasing sequence of the vertex degrees of a graph. Degree sequences are a natural way to break a graph search problem into independent cases (one for each possible degree sequence). Thus the search for a solution or all solutions can be done in parallel.

Since a degree sequence induces a partition on the vertex set, in order to compute an instance dependent canonizing symmetry break with respect to a degree sequence, a constraint stating that B has the same degree sequence as A needs to be added to (6′). The following specifies that an adjacency matrix complies to a given degree sequence. Here each conjunct is a cardinality constraint on a row of A.

$$\varphi_{degSeq}^{\langle d_1,\ldots,d_n\rangle}(A) = \bigwedge_{1\le i\le n}\left(\sum_{j=1}^{n} A_{i,j} = d_i\right) \tag{8}$$

4.6 Computing highly irregular graphs per degree sequence

A connected graph is called highly irregular if each of its vertices is adjacent only to vertices with distinct degrees [1]. The number of highly irregular graphs with $n \le 15$ vertices is detailed as sequence `A217246` in the OEIS [28]. By application of instance dependent canonizing symmetry breaks we extend this sequence with 5 new values. The following constraint specifies that the graph represented by adjacency matrix A with degree sequence $\langle d_1,\ldots,d_n\rangle$ is highly irregular.

$$\varphi_{hi}^{\langle d_1,\ldots,d_n\rangle}(A) = \bigwedge_{1\le i,j<k\le n\ s.t\ d_j=d_k} (\neg A_{i,j} \vee \neg A_{i,k}) \wedge \varphi_{degSeq}^{\langle d_1,\ldots,d_n\rangle}(A) \wedge \varphi_{con}^{n}(A) \tag{9}$$

Here, the formula $\varphi_{con}^{n}(A)$ specifies that the graph represented by adjacency matrix A is connected. The following constraint introduces propositional variables $p_{i,j}^{k}$ to express that vertices i and j are connected by a path that consists of intermediate vertices from the set $\{1,\ldots,k\}$.

$$\varphi_{con}^{n}(A) = \bigwedge_{1\le i,j\le n}\left(p_{i,j}^{0} \leftrightarrow A_{i,j}\right) \wedge \bigwedge_{1\le i,j,k\le n} p_{i,j}^{k} \leftrightarrow \left(p_{i,j}^{k-1} \vee \left(p_{i,k}^{k-1} \wedge p_{k,j}^{k-1}\right)\right) \wedge \wedge \bigwedge_{1\le i,j\le n}\left(p_{i,j}^{n}\right) \tag{10}$$

Our strategy is to compute all highly irregular graphs with n vertices in three steps: (1) We compute the set of degree sequences for all highly irregular graphs with n vertices; (2)

For each degree sequence we compute an instance dependent canonizing symmetry break; (3) We apply per degree sequence, the instance dependent canonizing symmetry break to compute the corresponding set of graphs with the corresponding degree sequence.

To perform the first step we apply a result from [21] which states that any degree sequence of a highly irregular graph is of the form $\langle m^{n_m}, \ldots, i^{n_i}, \ldots, 1^{n_1} \rangle$ where:

1. $n_i \geq n_m$ for $1 \leq i \leq m$; and
2. $\sum_{i=1}^{m} (n_i * i)$ and n_m are positive even numbers.

It is straightforward to enumerate all degree sequences for graphs with up to 20 vertices that satisfy this property. We then apply a SAT solver to determine which of these sequences is the degree sequence of some highly irregular graph. Step (2) is performed using a SAT solver, per degree sequence, by application of the above described adaptation of Algorithm 1 to compute an irredundant instance dependent canonizing set with respect to $\varphi_{hi}^{\langle d_1,\ldots,d_n \rangle}(A)$. In step (3) we enumerate all non-isomorphic highly irregular graphs per degree sequence with respect to the corresponding canonizing symmetry breaking constraints. We compute the graphs with a simple backtrack based (exhaustive search) program written in Java in which the variables of the adjacency matrix are assigned one by one until a solution is found.

Table 4 presents our results. The columns are divided into three pairs corresponding to the three steps described above: the first pair – computing degree sequences, the second pair – computing (instance dependent) canonizing permutation sets, and the third pair – computing solutions (using the derived canonizing symmetry breaks). Each pair presents the computation size and information on the solutions. For the first pair, the number of degree sequences. For the second pair, the average number of permutations in the canonizing permutation sets. In the third pair, the number of connected highly irregular graphs with n vertices (set in boldface). The values for $n \leq 15$ vertices correspond to those detailed as sequence `A217246` in the OEIS [28]. The values for $n > 15$ are new.

When computing solutions, as detailed in the rightmost pair of columns of Table 4, computation is performed in parallel, using a separate thread of the cluster for each degree sequence found in the first step. So for example, when $n = 20$, there are 151 parallel threads running with a total time of 7190.23 hours. This implies an average of about 47 hours. Note that we succeed to compute canonizing symmetry breaks for more than 20 nodes. We have not included the results in Table 4 as the subsequent graph enumeration problems involve a humongous number of graphs.

5 A generalization to matrix models

Graph search problems, as considered in this paper, are a special case of matrix search problems expressed in terms of a matrix of finite domain decision variables [13, 14, 35, 36]. Often, in such problems, both rows and columns can be permuted, possibly by different permutations, while preserving solutions. Matrix search problems with this property are called "*fully-interchangeable*" [36]. A graph search problem can be seen as a fully-interchangeable matrix search problem where the variables are Boolean, the matrix is symmetric and has *false* values on its diagonal, and symmetries involve permuting rows and columns, but only by the same permutation for both.

Similar to Definition 1, it is common to define a lex-order on matrix models. For matrices M_1 and M_2 (of the same dimension) with s_1 and s_2 the strings obtained by concatenating their rows, $M_1 \preceq M_2$ if and only if $s_1 \preceq_{lex} s_2$. Similar to Definition 2, the LEXLEADER

Table 4 Computing highly irregular graphs per degree sequence (time in seconds unless otherwise indicated)

	deg.seqs		can. sets		solutions	
n	time	deg.seqs	time	perms (avg.)	time	sols (total)
11	1.11	2	3.28	6.5	0.29	**21**
12	4.47	7	15.16	7.57	0.87	**110**
13	5.73	7	28.65	9.71	1.42	**474**
14	17.85	16	93.69	10.56	5.62	**2 545**
15	27.15	17	183.49	13.11	28.39	**18 696**
16	59.69	33	487.85	13.57	234.97	**136 749**
17	111.97	38	683.14	13.94	3 312.04	**1 447 003**
18	237.53	68	1 797.16	14.89	14.17 hr.	**18 772 435**
19	468.99	92	3 281.07	16.07	263.90 hr.	**303 050 079**
20	881.53	151	8 450.91	16.73	7190.23 hr.	**6 239 596 472**

method [11] can be applied to a fully-interchangeable matrix search problem to guarantee canonical solutions which are minimal in the lex ordering of matrices. For an $n \times m$ matrix search problem this involves potentially considering all $n! \times m!$ pairs of permutations (per n rows and per m columns). This is not practical as it means introducing $n! \times m!$ lex constraints on strings of size $n \times m$.

To this end, the DOUBLELEX constraint was introduced in [13] to lexicographically order (linearly) both rows and columns. The DOUBLELEX constraint can be viewed as derived by a subset of the constraints imposed in the LEXLEADER method [35]. For a matrix with n rows and m columns this boils down to a total of only $(n-1)+(m-1)$ permutations. The DOUBLELEX constraint has been shown to be very effective at eliminating much of the symmetry in a range of fully-interchangeable matrix search problem. Still, it does not break all of the symmetries broken by LEXLEADER.

To demonstrate the application of our methods to matrix models, we compare the DOUBLELEX symmetry break with canonical symmetry breaking for the application to EFPA (Equi-distant Frequency Permutation Array). An instance of the EFPA problem takes the form (q, λ, d, v). The goal is to find a set of v words of length $q\lambda$ such that each word contains λ copies of the symbols 1 to q, and each pair of words is Hamming distance d apart. The problem is naturally expressed as a $v \times q\lambda$ (fully-interchangeable) matrix search problem.

Table 1 in the survey by Walsh [35] illustrates the power of the DOUBLELEX symmetry break. The table details the number of solutions found with DOUBLELEX constraint for several instances of the EFPA problem in contrast to the total number of non-symmetric solutions. It demonstrates that DOUBLELEX leaves very few redundant solutions.

We have adapted Algorithms 1 and 2 so that they apply to search for pairs of permutations which induce constraints to break all symmetries in fully-interchangeable matrix search problems. With these constraints we obtain only the canonical solutions. We have applied such constraints to the instances of the EFPA problem considered in [35]. For matrix search problems we initialize Algorithm 1 taking Π to include the permutation pairs corresponding to the DOUBLELEX symmetry break.

Table 5 summarizes our results obtained, as all results in this paper, using the finite-domain constraint compiler BEE [26] with the underlying Glucose 4.0 SAT solver [5]. On

Table 5 Number of solutions for EFPA with DOUBLELEX and canonizing symmetry breaks

	DOUBLELEX sym. break			canonizing sym. break			
(q, λ, d, v)	perms	sat (sec.)	sols	perms (Δ)		sat (sec.)	sols
(3, 3, 2, 3)	10	0.01	6	8	(−2)	0.01	**6**
(4, 3, 3, 3)	13	0.06	16	16	(3)	0.06	**8**
(4, 4, 2, 3)	17	0.05	12	15	(−2)	0.03	**12**
(3, 4, 6, 4)	14	19.12	11 215	328	(314)	14.10	**1 427**
(4, 3, 5, 4)	14	145.22	61 258	1537	(1523)	414.56	**8 600**
(4, 4, 5, 4)	18	280.33	72 251	1793	(1775)	748.77	**9 696**
(5, 3, 3, 4)	17	0.29	20	27	(10)	0.22	**5**
(3, 3, 4, 5)	12	0.36	71	36	(24)	0.45	**18**
(3, 4, 6, 5)	15	611.88	77 535	988	(973)	195.17	**4 978**
(4, 3, 4, 5)	15	11.34	2 694	245	(230)	9.51	**441**
(4, 4, 2, 5)	19	0.10	12	15	(−4)	0.11	**12**
(4, 4, 4, 5)	19	22.42	4 604	385	(366)	25.41	**717**
(4, 6, 4, 5)	27	46.88	5 048	441	(414)	75.81	**819**
(5, 3, 4, 5)	18	157.94	20 831	898	(880)	262.02	**3 067**
(6, 3, 4, 5)	21	1230.19	111 082	2348	(2327)	3537.54	**15 192**

the left side the table details statistics for solutions with DOUBLELEX: the number of permutation pairs introduced by DOUBLELEX, the solving time (in seconds) and the number of solutions found. The right side of the table details the same for the canonizing symmetry breaks. Here, the detailed number of permutation pairs are those for the canonizing symmetry breaks, as discovered using the versions of Algorithms 1 and 2 adapted for use with matrix models. Here we also make explicit the number of additional permutations Δ, in addition to those introduced by DOUBLELEX, required to provide a canonizing symmetry break. In several rows of the table, corresponding to instances where DOUBLELEX is in fact complete, this value is negative. In these cases no permutations were added by Algorithm 1 and several were removed by Algorithm 2 when deriving the corresponding canonizing symmetry break. It is interesting to note that, often times, for canonical symmetry breaks, only a few permutations need be added on top of these already introduced by DOUBLELEX. In [19], the authors provide a first experimental study on how much symmetry is left after applying the DOUBLELEX constraint. Table 5 (in the column labeled Δ) illustrates the cost of breaking the symmetries left after applying the DOUBLELEX constraint. The numbers in the rightmost columns (set in boldface) correspond to the number of distinct non-symmetric solutions. The corresponding sets of permutation pairs for the instances in Table 5 can be obtained from http://www.cs.bgu.ac.il/~mcodish/Papers/Tools/canonizingSets.

We note that the numbers presented in the right side of Table 5 do not represent a win in time when compared to the results presented in [35]. However, it is of interest to observe that the DOUBLELEX symmetry break can be expressed in terms of a set of permutations which can then be considered as a starting point to derive a complete symmetry break.

To this end, we also note that other canonical orderings for matrix models have been considered as alternatives to the row-wise LEXLEADER order. For example, the Snake ordering [15] and the Gray ordering [27]. Based on these orderings, partial symmetry breaks

such as SNAKELEX were introduced and shown to be better choices for some types of problems compared to DOUBLELEX. We comment that it is not difficult to adapt Algorithms 1 and 2 to derive canonizing sets of permutations with respect to these alternative orderings. We leave this as a topic for future work.

6 Related work

Isomorphism free generation of combinatorial objects and particularly graphs, is a well studied topic [7, 12, 23, 31, 32]. Methods that generate the canonical representatives of each equivalence class are sometimes classified as "orderly" generation methods. This is a dynamic approach. Typically graphs are constructed by adding edges in iteration until a solution is found and backtracking when failing. In each such iteration the graph is checked to determine whether it can still be further extended: (a) to a solution of the graph search problem, and (b) to a canonical graph. Both of these tests consider only the fixed part of the partial graph. These techniques do not restrict the set of permutations to be canonical but rather consider all permutations relevant to the partially instantiated structure. Still, initially, there are very few permutations that need to be considered for (b) as the partial graph is still small. However, as the partial graph becomes more instantiated this test becomes harder and consumes more time. This approach is also the one applied in [36] for matrix search problems. In contrast our method is static. We aim to compute, before applying search, a small set of permutations that apply to break the symmetries in solutions. Our approach does not rely on which parts of the graph have already been determined during search.

7 Conclusion

We have illustrated the applicability of canonizing symmetry breaking constraints for small graph and matrix search problems. Although any row/column permutation is potentially a symmetry, we compute compact canonizing symmetry breaks, much smaller than those which consider all permutations. Our strategy is two phase. First, symmetry breaking constraints are computed. Second, these constraints are added to the model and then any solver can be applied to find (all) solutions which satisfy the model.

For graph search problems, we have presented methods that generate both instance independent and instance dependent symmetry breaking constraints. While instance dependent symmetry breaks have limited applicability since they grow enormously for graphs with more than 10 vertices, instance independent symmetry breaks have been successfully applied to compute new values in highly irregular graphs OEIS sequence for graphs with up to 20 vertices. For matrix search problems our focus is on instance dependent constraints.

Although, our approach is applicable only to graphs with small numbers of vertices, there are many open small graph search problems. For example the set of all Ramsey $R(4, 5; 24)$ graphs has not been determined yet. We are currently trying to extend our techniques to apply to compute symmetry breaks for this problem which involves only 24 vertices.

We note that our approach can also apply to improve dynamic symmetry breaking techniques. Given a partially instantiated graph, to determine if it is extendable to a canonical graph, one need not consider all of the permutations related to the already instantiated part. This is because some of those permutations are redundant.

Finally, we note the distinction made in [19] between complete symmetry breaking and complete pruning for a set of symmetry breaking constraints. Symmetry breaks remove all

symmetric solutions but not all symmetric branches of the search tree. An interesting future direction is to find a SAT encoding that enforces generalized arc consistency (GAC) on the set of symmetry breaking constraints. Perhaps an encoding that achieves this is not too big given the other constraints.

Acknowledgments We thank the anonymous reviewers of this paper for their constructive suggestions. In particular the addition of Section 5 is in view of the comments of the reviewers.

References

1. Alavi, Y., Chartrand, G., Chung, F.R., Erdös, P., Graham, R.L., & Oellermann, O.R. (1987). Highly irregular graphs. *Journal of Graph Theory*, *11*(2), 235–249.
2. Aloul, F.A. (2010). Symmetry in boolean satisfiability. *Symmetry*, *2*(2), 1121–1134. doi:10.3390/sym2021121.
3. Aloul, F.A., Markov, I.L., Ramani, A., & Sakallah, K.A. (2011). Breaking instance-independent symmetries in exact graph coloring. CoRR abs/1109.2347.
4. Aloul, F.A., Sakallah, K.A., & Markov, I.L. (2006). Efficient symmetry breaking for boolean satisfiability. *IEEE Transactions on Computers*, *55*(5), 549–558. doi:10.1109/TC.2006.75.
5. Audemard, G., & Simon, L. Glucose 4.0 SAT Solver.http://www.labri.fr/perso/lsimon/glucose/.
6. Babai, L., & Luks, E.M. (1983). Canonical labeling of graphs. In *Proceedings of the fifteenth annual ACM symposium on theory of computing* (pp. 171–183). ACM.
7. Brinkmann, G. (1996). Fast generation of cubic graphs. *Journal of Graph Theory*, *23*(2), 139–149.
8. Cameron, R., Colbourn, C., Read, R., & Wormald, N.C. (1985). Cataloguing the graphs on 10 vertices. *Journal of Graph Theory*, *9*(4), 551–562.
9. Codish, M., Miller, A., Prosser, P., & Stuckey, P.J. (2013). Breaking symmetries in graph representation. In Rossi, F. (Ed.), *Proceedings of the 23rd international joint conference on artificial intelligence, Beijing, China, August 3-9, 2013, IJCAI 2013*. IJCAI/AAAI. http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6480.
10. Codish, M., Miller, A., Prosser, P., & Stuckey, P.J. Constraints for symmetry breaking in graph representation (2014). Full version of [9] (in preparation).
11. Crawford, J.M., Ginsberg, M.L., Luks, E.M., & Roy, A. (1996). Symmetry-breaking predicates for search problems. In Aiello, L.C., Doyle, J., & Shapiro, S.C. (Eds.), *Proceedings of the fifth international conference on principles of knowledge representation and reasoning (KR'96), Cambridge, Massachusetts, USA, November 5-8, 1996* (pp. 148–159). Morgan Kaufmann.
12. Faradzev, I. (1978). Constructive enumeration of combinatorial objects. *Problmes Combinatoires et Thorie des Graphes Colloque Internat, Paris* (pp. 131–135).
13. Flener, P., Frisch, A.M., Hnich, B., Kiziltan, Z., Miguel, I., Pearson, J., & Walsh, T. (2002). Breaking row and column symmetries in matrix models. In Hentenryck, P.V. (Ed.), *Principles and practice of constraint programming - CP 2002, 8th international conference, CP 2002, Ithaca, NY, USA, September 9-13, 2002, proceedings, lecture notes in computer science* (Vol. 2470, pp. 462–476). Springer. http://link.springer.de/link/service/series/0558/bibs/2470/24700462.htm.
14. Frisch, A.M., Jefferson, C., & Miguel, I. (2003). Constraints for breaking more row and column symmetries. In Rossi, F. (Ed.), *Principles and practice of constraint programming - CP 2003, 9th international conference, CP 2003, Kinsale, Ireland, September 29 - October 3, 2003, Proceedings, lecture notes in computer science* (Vol. 2833, pp. 318–332). Springer. doi:10.1007/978-3-540-45193-8_22.
15. Grayland, A., Miguel, I., & Roney-Dougal, C.M. (2009). Snake lex: An alternative to double lex. In *Principles and practice of constraint programming - CP 2009, 15th international conference, CP 2009, Lisbon, Portugal, September 20-24, 2009, Proceedings* (pp. 391–399). doi:10.1007/978-3-642-04244-7_32.
16. Katebi, H., Sakallah, K.A., & Markov, I.L. (2010). Symmetry and satisfiability: An update. In Strichman, O., & Szeider, S. (Eds.), *Theory and applications of satisfiability testing - SAT 2010, 13th international conference, SAT 2010, Edinburgh, UK, July 11-14, 2010. Proceedings, lecture notes in computer science* (Vol. 6175, pp. 113–127). Springer. doi:10.1007/978-3-642-14186-7_11.
17. Katebi, H., Sakallah, K.A., & Markov, I.L. (2012). Conflict anticipation in the search for graph automorphisms. In Bjørner, N., & Voronkov, A. (Eds.), *Logic for programming, artificial intelligence, and reasoning - 18th international conference, LPAR-18, Mérida, Venezuela, March 11-15, 2012. Proceedings, lecture notes in computer science* (Vol. 7180, pp. 243–257). Springer. doi:10.1007/978-3-642-28717-6_20.

18. Katebi, H., Sakallah, K.A., & Markov, I.L. (2012). Graph symmetry detection and canonical labeling: Differences and synergies. In Voronkov, A. (Ed.), *Turing-100 - The Alan Turing Centenary, Manchester, UK, June 22-25, 2012, EPiC Series* (Vol. 10, pp. 181–195). EasyChair. http://www.easychair.org/publications/?page=949492057.
19. Katsirelos, G., Narodytska, N., & Walsh, T. (2010). On the complexity and completeness of static constraints for breaking row and column symmetry. In Cohen, D. (Ed.), *Principles and practice of constraint programming - CP 2010 - 16th international conference, CP 2010, St. Andrews, Scotland, UK, September 6-10, 2010. Proceedings, lecture notes in computer science* (Vol. 6308, pp. 305–320). Springer. doi:10.1007/978-3-642-15396-9_26.
20. Luks, E.M., & Roy, A. (2004). The complexity of symmetry-breaking formulas. *Annals of Mathematics and Artificial Intelligence*, *41*(1), 19–45.
21. Majcher, Z., & Michael, J. (1997). Degree sequences of highly irregular graphs. *Discrete Mathematics*, *164*(1), 225–236.
22. McKay, B. (1990). nauty user's guide (version 1.5). Tech. Rep. TR-CS-90-02. Australian National University. Computer Science Department.
23. McKay, B.D. (1998). Isomorph-free exhaustive generation. *Journal of Algorithms*, *26*(2), 306–324.
24. McKay, B.D., & Radziszowski, S.P. (1995). $R(4, 5) = 25$. *Journal of Graph Theory*, *19*(3), 309–322. doi:10.1002/jgt.3190190304.
25. McMillan, K.L. (2002). Applying SAT methods in unbounded symbolic model checking. In Brinksma, E., & Larsen, K.G. (Eds.), *Computer aided verification, 14th international conference, proceedings, lecture notes in computer science* (Vol. 2404, pp. 250–264). Springer. doi:10.1007/3-540-45657-0_19.
26. Metodi, A., Codish, M., & Stuckey, P.J. (2013). Boolean equi-propagation for concise and efficient SAT encodings of combinatorial problems. *Journal of Artificial Intelligence Research (JAIR)*, *46*, 303–341.
27. Narodytska, N., & Walsh, T. (2013). Breaking symmetry with different orderings. In *Principles and practice of constraint programming* (pp. 545–561). Springer.
28. The on-line encyclopedia of integer sequences. published electronically at http://oeis.org (2010).
29. Puget, J. (1993). On the satisfiability of symmetrical constrained satisfaction problems. In Komorowski, H.J., & Ras Z.W. (Eds.), *Methodologies for intelligent systems, 7th international symposium, ISMIS '93, Trondheim, Norway, June 15-18, 1993, Proceedings, lecture notes in computer science* (Vol. 689, pp. 350–361). Springer. doi:10.1007/3-540-56804-2_33.
30. Radziszowski, S.P. (1994). Small Ramsey numbers. *Electronic Journal of Combinatorics*. http://www.combinatorics.org/. Revision #14: January, 2014.
31. Read, R.C. (1978). Every one a winner or how to avoid isomorphism search when cataloguing combinatorial configurations. *Ann. Discrete Math*, *2*, 107–120.
32. Read, R.C. (1981). A survey of graph generation techniques. In *Combinatorial mathematics VIII* (pp. 77–89). Springer.
33. Shlyakhter, I. (2007). Generating effective symmetry-breaking predicates for search problems. *Discrete Applied Mathematics*, *155*(12), 1539–1548. doi:10.1016/j.dam.2005.10.018.
34. Walsh, T. (2006). General symmetry breaking constraints. In Benhamou, F. (Ed.), *Principles and practice of constraint programming - CP 2006, 12th international conference, CP 2006, Nantes, France, September 25-29, 2006, Proceedings, lecture notes in computer science* (Vol. 4204, pp. 650–664). Springer. doi:10.1007/11889205_46.
35. Walsh, T. (2012). Symmetry breaking constraints: Recent results. In Hoffmann, J., & Selman, B. (Eds.), *Proceedings of the twenty-sixth AAAI conference on artificial intelligence, July 22-26, 2012, Toronto, Ontario, Canada*. AAAI Press. http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/4974.
36. Yip, J., & Hentenryck, P.V. (2011). Symmetry breaking via lexleader feasibility checkers. In Walsh, T. (Ed.), *Proceedings of the 22nd international joint conference on artificial intelligence, IJCAI 2011, Barcelona, Catalonia, Spain, July 16-22, 2011* (pp. 687–692). IJCAI/AAAI. http://ijcai.org/papers11/Papers/IJCAI11-121.pdf.