Breaking Symmetries with High Dimensional Graph Invariants and Their Combination

Avraham Itzhakov(✉) and Michael Codish

Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel
{itzhakoa,mcodish}@cs.bgu.ac.il

Abstract. This paper illustrates the application of graph invariants to break symmetries for graph search problems. The paper makes two contributions: (1) the use of higher dimensional graph invariants in symmetry breaking constraints; and (2) a novel technique to obtain symmetry breaking constraints by combining graph invariants. Experimentation demonstrates that the proposed approach applies to provide new results for the generation of a particular class of cubic graphs.

1 Introduction

Graph search problems are about finding simple graphs with desired structural properties. Such problems arise in many real-world applications and are fundamental in graph theory. Solving graph search problems is typically hard due to the enormous search space and the large number of symmetries in graph representation. For graph search problems, any graph obtained by permuting the vertices of a solution (or a non-solution) is also a solution (or a non-solution), which is isomorphic, or "symmetric". When solving graph search problems, the presence of symmetries often causes redundant search effort by revisiting symmetric objects. To optimize the search we aim to restrict it to focus on one "canonical" graph from each isomorphism class.

A standard approach to eliminate symmetries is to add *symmetry breaking constraints* which are satisfied by at least one member of each isomorphism class [8,22,23]. A symmetry breaking constraint is called *complete* if it is satisfied by exactly one member of each isomorphism class and *partial* otherwise. We say that a symmetry breaking constraint is of polynomial size, if it has a representation in propositional logic which is polynomial in size. There is no known polynomial size complete symmetry breaking constraint for graph search problems. Therefore, in practice, one typically applies partial symmetry breaking constraints [5–7] which are polynomial in size.

Over the past decade, there has been little progress in the research of partial symmetry breaking constraints for graph search problems. Codish *et al.* [6,7] introduced a polynomial sized partial symmetry breaking constraint, denoted here as sb_{lex}, which restricts the search space to graphs with lexicographically minimal adjacency matrices with respect to permutations that swap two vertices. This constraint turns out to work well in practice, despite eliminating only a small portion of the symmetries. However, when dealing with hard instances of graph search problems, this constraint does not suffice.

Graph invariants [13] are properties of graphs (typically expressed numerically) which are preserved under isomorphism. Graph invariants have been extensively researched in various disciplines such as, chemistry [1], physics [9], and also in the context of graph isomorphism tools [15]. In this paper we use the following terminology. Graph invariants which relate to individual vertices, such as the degree of a vertex, are one dimensional and called "vertex invariants". Graph invariants which relate to pairs of vertices are two dimensional and called "pair invariants". Graph invariants which relate to sets of vertices (with at least two elements) are called "high dimensional". Previous approaches that consider structural information to improve on sb_{lex} apply one dimensional graph invariants in combination with the lexicographic order. For example, in [5,18], the authors combine lexicographic order with information about vertex degrees.

This paper explores the application of higher dimensional graph invariants to break symmetries. We focus on one and two dimensional invariants. However, all the techniques demonstrated apply to invariants of any dimension. We study two techniques to combine graph invariants. First, we introduce the "chain" symmetry breaking constraint which generalizes the standard approach for breaking symmetries with graph invariants. The chain constraint combines a given series of graph invariants to break symmetries such that each invariant refines its predecessors. We then introduce the "product" symmetry breaking constraint which combines graph invariants by interleaving them. We demonstrate the advantage of this approach over the chain constraint. Finally, we demonstrate the application of high dimensional graph invariants to generate connected claw-free cubic graphs of order $n \leq 36$ vertices where existing symmetry breaking methods do not suffice. The results for 32, 34, and 36 vertices are new.

The computations detailed throughout this paper are performed using the finite-domain constraint compiler BEE [17] which compiles constraints to a CNF and solves it applying an underlying SAT solver. We use Clasp 3.1.3 [12] as the underlying SAT solver. All experiments run on an Intel Xeon E5-2660 with CPU's clocked at 2 GHz, Each instance is run on a single thread.

2 Preliminaries and Notation

Throughout this paper we consider simple graphs, i.e. undirected graphs with no self loops. The vertex set of a graph $G = (V, E)$ of order n, is denoted $V(G)$ and assumed to be $V = \{1, \dots, n\}$. The edge set of G is denoted $E(G) \subseteq V \times V$. The adjacency matrix of G is an $n \times n$ Boolean matrix which, in abuse of notation, is also denoted G. The element at row i and column j is denoted $G_{i,j}$ and is *true* if and only if (i, j) is an edge in G. The set of neighbors of an edge $v \in V$ is denoted $N_G(v)$. The degree of a vertex $v \in V$ is the number of its neighbors, and is denoted $deg_G(v)$. The set of simple graphs on n vertices is denoted $\mathcal{G}_n$. An *unknown graph* of order n is represented as an $n \times n$ adjacency matrix of Boolean variables which is symmetric and has the values *false* (denoted by 0) on the diagonal. A graph H is a *subgraph* of G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. A graph H is called an *induced subgraph* of G if H is a subgraph of G and every edge in G that connects vertices from $V(H)$ also appears in $E(H)$. In other words, the graph H is an induced subgraph of G if H and G have the same edges between the vertices of H.

The group of all permutations on $\{1 \dots n\}$ is denoted S_n. We represent a permutation $\pi \in S_n$ as a sequence of length n where the ith element indicates the value of $\pi(i)$. For example, the permutation $[2, 3, 1] \in S_3$ maps as follows: $\{1 \mapsto 2, 2 \mapsto 3, 3 \mapsto 1\}$. A *transposition* is a permutation which swaps two elements and is the identity for all other elements. The set of all transpositions on $\{1 \dots n\}$ is denoted T_n. The transposition which swaps i and j is denoted $\pi_{i,j}$. For example, the transposition $\pi_{1,3} \in T_4$ maps as follows: $\{1 \mapsto 3, 2 \mapsto 2, 3 \mapsto 1, 4 \mapsto 4\}$. Permutations act on graphs and on unknown graphs in the natural way. For a graph $G \in \mathcal{G}_n$ and also for an unknown graph G, viewing G as an adjacency matrix, given a permutation $\pi \in S_n$, then $\pi(G)$ is the adjacency matrix obtained by mapping each element $G_{i,j}$ to $G_{\pi(i),\pi(j)}$ (for $1 \leq i, j \leq n$). The permutation, $\pi(G)$ of G, can equivalently be described as the adjacency matrix obtained by permuting both rows and columns of G using π. Two graphs $G, H \in \mathcal{G}_n$ are *isomorphic* if there exists a permutation $\pi \in S_n$ such that $G = \pi(H)$.

The standard lexicographic order on strings is denoted $\leq_{\text{lex}}$. We consider also lexicographic orders between integer and Boolean matrices, always comparing matrices of the same type, dimension and order. In our context, matrices of dimension $k > 1$ are always symmetric and have fixed values on the diagonal. We define the lexicographic ordering of two such matrices M_1 and M_2 as follows: $M_1 \leq_{\text{lex}} M_2$ if and only if $vec(M_1) \leq_{\text{lex}} vec(M_2)$ where $vec(M)$ is a string defined by concatenating the rows of M. For a matrix M with dimension $k = 1$, M is a vector and $vec(M)$ is the string of its elements. When M is of dimension $k = 2$, because of symmetry and fixed values on the diagonal, $vec(M)$ can be viewed as the concatenation of the rows of the upper triangle of M [4]. For higher dimensions, $k > 2$, the definition extends in the natural way.

In particular for graphs $G, H \in \mathcal{G}_n$, $G \leq_{\text{lex}} H$ defines a lexicographic ordering on graphs. When G, H are unknown graphs, represented as adjacency matrices of Boolean variables, then the lexicographic ordering, $G \leq_{\text{lex}} H$, can be viewed as specifying a *lexicographic order constraint* over these variables. This constraint is true with respect to an assignment for the variables of G, H if $G \leq_{\text{lex}} H$ under this assignment. We call such a constraint a *"lex-constraint"*. The case for $M_1 \leq_{\text{lex}} M_2$ where M_1, M_2 are matrices of integer variables is similar.

Example 1. Figure 1 depicts an unknown, order 5, graph G and its permutation $\pi(G)$, for $\pi = [2, 1, 3, 5, 4]$, both represented as adjacency matrices of Boolean variables. Note, for example, that the variable x_2 occurs at position $(1, 3)$ in G and at position $(\pi(1), \pi(3)) = (2, 3)$ in $\pi(G)$. The lex-constraint $G \leq_{\text{lex}} \pi(G)$ is

$$[x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}] \leq_{\text{lex}} [x_1, x_5, x_7, x_6, x_2, x_4, x_3, x_9, x_8, x_{10}]$$

$$\mathbf{G} = \begin{bmatrix} 0 & x_1 & x_2 & x_3 & x_4 \\ x_1 & 0 & x_5 & x_6 & x_7 \\ x_2 & x_5 & 0 & x_8 & x_9 \\ x_3 & x_6 & x_8 & 0 & x_{10} \\ x_4 & x_7 & x_9 & x_{10} & 0 \end{bmatrix} \qquad \pi(\mathbf{G}) = \begin{bmatrix} 0 & x_1 & x_5 & x_7 & x_6 \\ x_1 & 0 & x_2 & x_4 & x_3 \\ x_5 & x_2 & 0 & x_9 & x_8 \\ x_7 & x_4 & x_9 & 0 & x_{10} \\ x_6 & x_3 & x_8 & x_{10} & 0 \end{bmatrix}$$

Fig. 1. An unknown graph G and its permutation $\pi(G)$ for $\pi = [2, 1, 3, 5, 4]$.

where the sequences on the left and on the right are obtained by concatenating the rows of the upper triangles of the corresponding graphs. This constraint can be simplified as described by Frisch *et al.* [11] to

$$[x_2, x_3, x_4, x_8] \leq_{\text{lex}} [x_5, x_7, x_6, x_9]$$

□

An order n graph search problem is a predicate, $\varphi(G)$, on an unknown, order n graph G, which is closed under isomorphism. A solution to $\varphi(G)$ is a satisfying assignment for the variables of G. Given a (non-)solution for a graph search problem, each permutation of its vertices yields a symmetric (non-)solution. One common way to break symmetries in graph search problems is to define a symmetry breaking predicate which is satisfied only by the minimal representatives of each isomorphism class with respect to some total order $\preceq$.

Theorem 1. *Let G be an unknown order n graph and let $\preceq$ be a total order on graphs. Then,*

$$\textsc{Can}_{\preceq}(G) = \bigwedge_{\pi \in S_n} G \preceq \pi(G)$$

is a complete symmetry breaking constraint.

Proof. Since $\preceq$ is a total order, every isomorphism class $\mathcal{I}$ of graphs contains a unique minimal member G with respect to $\preceq$. By definition, G satisfies the constraint $\textsc{Can}_{\preceq}(G)$. Suppose that $H \in \mathcal{I}$ also satisfies $\textsc{Can}_{\preceq}$. Because $H \in \mathcal{I}$ it follows that $G = \pi(H)$ for some $\pi \in S_n$. Because H satisfies $\textsc{Can}_{\preceq}$ then $H \preceq G$. Because G is minimal $G \preceq H$. Hence $G = H$. □

The complete symmetry breaking constraint $\textsc{Can}_{\preceq}$ is impractical as it is composed of a super-exponential number of constraints, one for each permutation of the vertices. Hence, in practice, one often applies a partial symmetry breaking constraint defined in terms of a polynomial sized subset of the $\textsc{Can}_{\preceq}$ constraints.

Example 2. A classic example of a total order for graphs is the $\leq_{\text{lex}}$ order. The corresponding complete symmetry breaking constraint $\textsc{Can}_{\leq_{\text{lex}}}$ is often referred to as the lex-leader constraint [20]. Codish *et al.* [6,7] introduced a partial symmetry breaking constraint which is equivalent to taking the subset of the lex-leader constraints corresponding to all transpositions (permutations which swap two values), as specified below.

$$sb_{\text{lex}}(G) = \bigwedge_{\pi \in T_n} G \leq_{\text{lex}} \pi(G) \tag{1}$$

The sb_{lex} constraint is composed of a quadratic number of lex-constraints. It is compact and turns out to be effective when solving a wide range of graph search problems. □

Observation 1. *if $\preceq$ is a weak order on graphs, instead of a total order, then* $\text{CAN}_{\preceq}$ *is a partial symmetry breaking constraint. Moreover, any constraint defined as a subset of the constraints in* $\text{CAN}_{\preceq}$ *is also a partial symmetry breaking constraint.*

Proof. For the first claim, the proof is similar to the proof of Theorem 1. However, there is no guarantee that the minimum is unique. Hence, the corresponding symmetry breaking constraint is partial. For the second claim, weakening a partial symmetry breaking constraint results in a partial symmetry breaking constraint. □

3 Graph Invariants and Their Induced Graph Orderings

In this section, we recall the notion of graph invariants and in particular, high dimensional graph invariants. We propose a constraint-based representation for invariants of unknown graphs which is an essential component when defining symmetry breaking constraints. Finally, we introduce an ordering on graphs based on their corresponding graph invariant values.

A *k-dimensional graph invariant* is a function f which maps a graph G and a set $S = \{ v_1, \ldots, v_k \} \subseteq V(G)$ of k vertices to a value which is invariant under graph isomorphism. Namely, for every permutation π of the vertex set $V(G)$ it holds that $f(G,S) = f(\pi(G), \pi(S))$. When the graph G is fixed, we denote the function $f^G(S) = f(G,S)$. In this paper, we focus primarily on the special cases of 1-dimensional and 2-dimensional graph invariants which we call *vertex invariants* and *pair invariants*, respectively.

Figure 2 introduces several graph invariants that we refer to in the remainder of the paper. Let G be a graph. The degree invariant assigns each vertex to its degree. The common neighbors invariant assigns each pair of vertices to the number of their common neighbors. The min (max) degree invariant assigns each pair of vertices to their minimal (maximal) degree. The triangles invariant assigns each vertex to the number of triangles (cycles of length 3) in which it occurs. In the figure, (u, v) denotes a pair of distinct vertices and the pair invariants are not defined when $u = v$. The *inverse* of a k-dimensional invariant f, denoted $-f$, is also a k-dimensional invariant which maps every input to the minus of the corresponding value of f. For instance, the invariant $-f^G_{\text{common}}$ specifies the minus of the number of common neighbors for each pair of vertices in G. Namely for vertices u and v, $(-f^G_{\text{common}})(u,v) = -(f^G_{\text{common}}(u,v))$.

For a fixed graph G, a k-dimensional graph invariant f^G can be viewed as a k-dimensional matrix. For a set of vertices, $S = \{ v_1, \ldots v_k \}$, the element at

<u>*degree invariant*</u>

$f^G_{\text{deg}} : V \to \mathbb{N}$
$f^G_{\text{deg}}(v) = deg_G(v)$

<u>*min degree invariant*</u>

$f^G_{\min} : V \times V \to \mathbb{N}$
$f^G_{\min}(u, v) = \min(deg_G(u), deg_G(v))$

<u>*common neighbors invariant*</u>

$f^G_{\text{common}} : V \times V \to \mathbb{N}$
$f^G_{\text{common}}(u, v) = |N(u) \cap N(v)|$

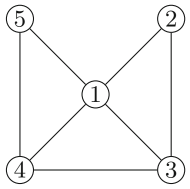
<u>*max degree invariant*</u>

$f^G_{\max} : V \times V \to \mathbb{N}$
$f^G_{\max}(u, v) = \max(deg_G(u), deg_G(v))$

<u>*triangles invariant*</u>

$f^G_{\text{triangles}} : V \to \mathbb{N}$
$f^G_{\text{triangles}}(v) = |\, \{\ (u, w) \in E(G) \,|\, v \in N(u) \cap N(w)\ \} \,|$

Fig. 2. Several example graph invariants.



$$f^G_{\text{deg}} = \begin{bmatrix} 4 \\ 2 \\ 3 \\ 3 \\ 2 \end{bmatrix} \qquad f^G_{\text{common}} = \begin{bmatrix} - & 1 & 2 & 2 & 1 \\ 1 & - & 1 & 2 & 1 \\ 2 & 1 & - & 1 & 2 \\ 2 & 2 & 1 & - & 1 \\ 1 & 1 & 2 & 1 & - \end{bmatrix}$$

Fig. 3. A graph G with matrix representation for f^G_{deg} and f^G_{common}.

position $\langle v_1, \dots v_k \rangle$ in the matrix specifies the integer value $f^G(S)$. The following example illustrates this representation.

Example 3. Figure 3 details the matrix representation for graph invariants f^G_{deg} and f^G_{common} for the graph G depicted on the left. The i-th entry in the vector f^G_{deg} specifies the degree of vertex i in G. For instance, the degree of vertex 1 is 4. The (i, j) entry in the matrix f^G_{common} specifies the number of common neighbors of vertices i and j. For instance, the value in the entry $(1, 3)$ is 2 because vertices 1 and 3 share two neighbors (vertices 2 and 4). Notice that the values on the diagonal are not defined.

When G is an unknown graph, The invariant f^G can be viewed as a k-dimensional matrix of integer variables, together with a constraint μ which links the Boolean variables in G and the integer variables in f^G. Solutions of μ instantiate G to a graph and f^G to corresponding invariant values.

$$\mathbf{G} = \begin{bmatrix} 0 & x_1 & x_2 & x_3 & x_4 \\ x_1 & 0 & x_5 & x_6 & x_7 \\ x_2 & x_5 & 0 & x_8 & x_9 \\ x_3 & x_6 & x_8 & 0 & x_{10} \\ x_4 & x_7 & x_9 & x_{10} & 0 \end{bmatrix} \qquad \mathbf{f^G_{deg}} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{bmatrix} \qquad \mu = \begin{bmatrix} d_1 = x_1 + x_2 + x_3 + x_4 \ \wedge \\ d_2 = x_1 + x_5 + x_6 + x_7 \ \wedge \\ d_3 = x_2 + x_5 + x_8 + x_9 \ \wedge \\ d_4 = x_3 + x_6 + x_8 + x_{10} \ \wedge \\ d_5 = x_4 + x_7 + x_9 + x_{10} \ \wedge \end{bmatrix}$$

Example 4. Figure 3 details an unknown graph G of order 5 and the corresponding matrix representation of f^G_{deg}. The constraints in μ specify the relationship between the Boolean variables in G and the integer variables in f^G. The integer variable d_i in f^G represents the degree of the i^{th} vertex in G.

We observe that a (possibly unknown) graph can also be viewed as a two dimensional graph invariant which specifies the adjacency relation. Let G be a graph. Then,

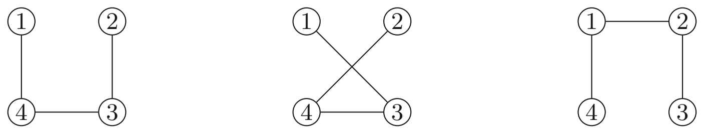
$$f^G_{\text{adj}}(u,v) = \begin{cases} 1 & \text{if } (u,v) \in E(G) \\ 0 & \text{else} \end{cases}$$

The matrix representation of f^G_{adj} is identical to the adjacency matrix of G, except that integer values (one and zero) occur instead of Boolean values (true and false) and the diagonal calls are undefined instead of false.

An essential component to define symmetry breaking constraints based on graph invariants is a notion of graph ordering with respect to an invariant f.

Definition 1 *(invariant induced graph ordering)***.** *Let $G, H \in \mathcal{G}_n$ and let f be a graph invariant. Recall that vec is a flattening of the (upper triangle of the) matrix into a string of values. Then, $G \preceq_f H \quad \Leftrightarrow \quad vec(f^G) \leq_{lex} vec(f^H)$. We write $G =_f H$ if $G \preceq_f H$ and $H \preceq_f G$.*

In general, depending on the specific invariant f, $\preceq_f$ is possibly a weak order as distinct graphs may admit the same values for the invariant f. The following example demonstrates that $\preceq_{f_{\text{deg}}}$ is a weak order.



$$\begin{bmatrix}1\\1\\2\\2\end{bmatrix} \quad \begin{bmatrix}- & 0 & 0 & 1\\0 & - & 1 & 0\\0 & 1 & - & 1\\1 & 0 & 1 & -\end{bmatrix} \qquad \begin{bmatrix}1\\1\\2\\2\end{bmatrix} \quad \begin{bmatrix}- & 0 & 1 & 0\\0 & - & 0 & 1\\1 & 0 & - & 1\\0 & 1 & 1 & -\end{bmatrix} \qquad \begin{bmatrix}2\\2\\1\\1\end{bmatrix} \quad \begin{bmatrix}- & 1 & 0 & 1\\1 & - & 1 & 0\\0 & 1 & - & 0\\1 & 0 & 0 & -\end{bmatrix}$$

Fig. 4. isomorphic representations of P_4 and their $f_{\text{deg}}, f_{\text{adj}}$ values.

Example 5. Consider three isomorphic representations of P_4 (path on four vertices), as depicted in Fig. 4. The minimal graph amongst them with respect to the total order $\preceq_{f_{\text{adj}}}$ is the leftmost graph. The leftmost and the center graphs are both minimal with respect to the weak order $\preceq_{f_{\text{deg}}}$. $\square$

4 Symmetry Breaking Constraints with Graph Invariants

A classic way to refine the partial symmetry breaking constraint sb_{lex} presented in Eq. (1) is to specify a partition of the vertices with respect to a graph invariant, and to post a lex-constraint for the subset of transpositions which preserve the partition. In [5], the authors refine sb_{lex} with respect to a partition based on the degree invariant. This symmetry breaking constraint, which we denote here $sb^{deg}_{\text{lex}}(G)$, is defined as follows where G is an order n unknown graph:

$$\underbrace{\bigwedge_{1\leq i<n} f^G_{\text{deg}}(i) \leq f^G_{\text{deg}}(i+1)}_{(a)} \wedge \underbrace{\bigwedge_{1\leq i<j\leq n} f^G_{\text{deg}}(i) = f^G_{\text{deg}}(j) \implies G \leq_{\text{lex}} \pi_{i,j}(G)}_{(b)} \quad (2)$$

The left conjunct (a) constrains the degrees of the vertices of G to be sorted in non-decreasing order. This induces a vertex partition where vertices with equal degree are in the same part of the partition. The right conjunct (b) enforces G to be minimal with respect to all transpositions which preserve the vertex partition. Equation (2) can be rewritten using the invariant based graph ordering from Definition 1, as follows.

$$\underbrace{\bigwedge_{\pi\in T_n} G \preceq_{f_{\text{deg}}} \pi(G)}_{(a')} \wedge \underbrace{\bigwedge_{\pi\in T_n} G =_{f_{\text{deg}}} \pi(G) \implies G \preceq_{f_{\text{adj}}} \pi(G)}_{(b')} \quad (3)$$

The left and right parts (a') and (b') of Eq. (3) are equivalent respectively to parts (a) and (b) of Eq. (2). The formulation of $sb^{deg}_{\text{lex}}(G)$ as specified in Eq. (3) combines two graph orderings $\preceq_{f_{\text{deg}}}$ and $\preceq_{f_{\text{adj}}}$ to break symmetries. In this combination, graphs are first ordered by $\preceq_{f_{\text{deg}}}$ and then ties are broken according to $\preceq_{f_{\text{adj}}}$. We generalize this "standard" approach to apply a series of graph invariants and term this way of combining graph invariants "chaining". First, we introduce an ordering induced by a sequence of invariants.

Definition 2 *(The chain ordering).* *Let $\langle f_1, \ldots, f_n\rangle$ be a sequence of graph invariants. Then, for any two graphs $G, H \in \mathcal{G}_n$, we define*

$$G \preceq_{\langle f_1,\ldots,f_m\rangle} H = \begin{cases} (G \preceq_{f_1} H) \wedge (G =_{f_1} H \implies G \preceq_{\langle f_2,\ldots,f_m\rangle} H) & \textit{if } m > 0 \\ \textit{true} & \textit{otherwise} \end{cases}$$

The chain ordering induces a chain symmetry breaking constraint, as specified in the following definition.

Definition 3 *(The chain constraint).* *Let G be an unknown graph of order n and let $\langle f_1, \ldots, f_m\rangle$ be a sequence of graph invariants. Then, the chain symmetry breaking constraint induced by $\langle f_1, \ldots, f_m\rangle$ is*

$$sb^{f_1,\ldots,f_m}_{chain}(G) = \bigwedge_{\pi\in T_n} G \preceq_{\langle f_1,\ldots,f_m\rangle} \pi(G)$$

One can check that, in general, the chain ordering is a weak order on graphs. Hence, by Observation 1, the chain symmetry breaking constraint induced by a sequence $\langle f_1, \dots . f_n \rangle$ is a partial symmetry breaking constraint. Observe also that $sb^{deg}_{\text{lex}}(G)$ as expressed in Eq. (3) is a special case of Definition 3 and is equivalent to the chain symmetry breaking constraint induced by $\langle f_{deg}, f_{adj} \rangle$.

As illustrated in the following example, the chain constraint can be alternatively expressed as a conjunction of lex-constraints. Each constraint of the form $G \preceq_{\langle f_1,\dots,f_{\text{m}} \rangle} \pi(G)$ is equivalent to the lex-constraint

$$vec(f^G_1, \dots, f^G_m) \leq_{\text{lex}} vec(f^{\pi(G)}_1, \dots, f^{\pi(G)}_m)$$

where $vec(f^G_1, \dots, f^G_m)$ is obtained by concatenating $vec(f^G_1), \dots, vec(f^G_m)$ and similarly for $vec(f^{\pi(G)}_1, \dots, f^{\pi(G)}_m)$.

Example 6. Consider the unknown graph G of order 4, the invariant f^G_{common} and its constraints μ which are detailed below.

$$G = \begin{bmatrix} 0 & x_1 & x_2 & x_3 \\ x_1 & 0 & x_4 & x_5 \\ x_2 & x_4 & 0 & x_6 \\ x_3 & x_5 & x_6 & 0 \end{bmatrix} \quad f^G_{\text{common}} = \begin{bmatrix} - & y_1 & y_2 & y_3 \\ y_1 & - & y_4 & y_5 \\ y_2 & y_4 & - & y_6 \\ y_3 & y_5 & y_6 & - \end{bmatrix} \quad \mu = \begin{bmatrix} y_1 = x_2 * x_4 + x_3 * x_5 \wedge \\ y_2 = x_1 * x_4 + x_3 * x_6 \wedge \\ y_3 = x_1 * x_5 + x_2 * x_6 \wedge \\ y_4 = x_1 * x_2 + x_5 * x_6 \wedge \\ y_5 = x_1 * x_3 + x_4 * x_6 \wedge \\ y_6 = x_2 * x_3 + x_4 * x_5 \wedge \end{bmatrix}$$

The chain symmetry breaking constraint induced by $\langle f_{\text{common}}, f_{\text{adj}} \rangle$ consists of 6 constraints of the form $G \preceq_{\langle f_1,\dots,f_m \rangle} \pi_{i,j}(G)$, one for each transposition $\pi_{i,j}$. Each of these can be expressed as a lex-constraint. The lex-constraint corresponding to $\pi_{1,2}$ is

$$[y_1, \dots, y_6, x_1, \dots, x_6] \leq_{\text{lex}} [y_1, y_4, y_5, y_2, y_3, y_6, x_1, x_4, x_5, x_2, x_3, x_6]$$

The vector on the left of the constraint consists of the variables from the invariant matrix followed by the variables from the adjacency matrix. The vector on the right, consists of the variables of the corresponding matrices obtained by swapping rows 1 and 2 as well as columns 1 and 2. Both vectors involve y variables first (from the graph invariant), followed by x variables (from the adjacency matrix). This constraint further simplifies to:

$$[y_2, y_3, x_2, x_3] \leq_{\text{lex}} [y_4, y_5, x_4, x_5]$$

□

When combining a sequence, $\langle f_1, ...f_m \rangle$, of graph invariants as a chain constraint, not every sequence "makes sense". Each invariant f_i in the sequence should "refine" those preceding it. We say that f_i refines $f_1, \dots, f_{i-1}$ if the set of graphs which satisfy $sb^{f_1,\dots,f_i}_{\text{chain}}$ is a strict subset of the set of graphs which satisfy $sb^{f_1,\dots,f_{i-1}}_{\text{chain}}$. Adding an invariant which does not refine those preceding it does not make sense as it adds no precision. For example, the order induced by $\langle f_{\text{adj}}, f_{\text{deg}}\rangle$ is equivalent to the order induced by $\langle f_{\text{adj}}\rangle$. This is because if $G =_{f_{\text{adj}}} H$ holds, then also $G =_{f_{\text{deg}}} H$ holds. In practice, if f_{adj} occurs in a sequence combined as a chain constraint, then it should always be the last invariant in the sequence as it is the "most refined".

Table 1. Generating all order n graphs with various symmetry breaking constraints.

method	order						
	5	6	7	8	9	10	11
part 1: base							
exact	34	156	1,044	12,346	274,668	12,005,168	1,018,997,864
sb_{lex}	43 0.00 s	276 0.00 s	3,158 0.01 s	66,595 0.20 s	2,587,488 6.49 s	184,192,329 8.20 m	23,963,012,033 20.91 h
part 2: chain							
$f_{\text{deg}}, f_{\text{adj}}$	34 0.00 s	158 0.00 s	1,143 0.03 s	14,937 0.61 s	363,373 30.69 s	16,773,384 19.66 h	T.O
$f_{\text{common}}, f_{\text{adj}}$	43 0.00 s	231 0.02 s	1,933 0.37 s	28,184 7.32 s	748,727 32.08 m	T.O	T.O
$f_{\text{deg}}, f_{\text{common}}, f_{\text{adj}}$	34 0.00 s	156 0.02 s	1,075 0.28 s	13,223 5.62 s	305,189 54.33 m	T.O	T.O
part 3: product							
$f_{\text{common}}, f_{\text{adj}}$	43 0.00 s	226 0.00 s	1,852 0.11 s	26,030 1.24 s	673,069 47.25 s	32,881,227 12.01 h	T.O
$f_{\text{adj}}, f_{\text{common}}$	42 0.00 s	231 0.00 s	1,949 0.06 s	27,620 0.84 s	715,804 22.38 s	35,060,107 78.43 m	T.O
$f_{\text{adj}}, f_{\text{min}}, f_{\text{max}}$	43 0.00 s	215 0.01 s	1,669 0.07 s	22,464 0.86 s	562,234 24.98 s	26,480,344 95.02 m	T.O
$f_{\text{adj}}, f_{\text{min}}, f_{\text{max}}, f_{\text{common}}$	42 0.00 s	210 0.01 s	1,553 0.19 s	19,209 1.95 s	437,794 63.83 s	19,188,298 16.59 h	T.O

Table 1 illustrates the impact of various symmetry breaking constraints based on combinations of graph invariants. To this end, we compute all order $5 \leq n \leq 11$ graphs using various symmetry breaking constraints. Cells in the table which detail computations performed in this paper consist of two numbers: the number of solutions (above) and the computation time (below). All times are CPU running times specified in an appropriate unit: (s) seconds, (m) minutes, or (h) hours. A timeout (TO) of 24 h is applied. The rows of the table are divided into three parts titled: "base", "chain" and "product".

The first part of Table 1 (titled "base"): provides the base for comparison and consists of two rows. First, the "exact" number of order n graphs (modulo isomorphism) [19] (sequence A000088 of the OEIS). This is the base comparison for precision. For other computations, the closer the number of computed graphs is to these values, the more precise the result. The second row details the number of graphs computed using the sb_{lex} constraint introduced in [6,7]. Applying this constraint, the number of graphs generated is up to 20 times larger than the actual number of graphs modulo isomorphism. This is the least precise configuration described in the table but the only one that can find all solutions for $n = 11$ with the specified timeout.

The second part of Table 1 (titled "chain"): consists of three rows which detail the computation of all graphs using the chain symmetry breaking constraint combining various sequences of graph invariants. Note that the computations are more precise than with sb_{lex} but considerably slower. In particular, when combining three invariants (row three of part two), the computation becomes slightly more precise but considerably slower than when combining two.

A possible explanation for the inefficiency when chaining invariants is that they allow less propagations on the variables of the unknown graph. Generally speaking, for a lex-constraint of the form $a_1, \ldots, a_n \leq_{\text{lex}} b_1, \ldots, b_n$ between strings of variables, propagation on the domain of a variable a_i or b_i, depends on changes to the domains of the variables to the left: $a_1, \ldots, a_{i-1}$ and $b_1, \ldots, b_{i-1}$.

To better understand, we focus in the following example on the comparison of sb_{lex} and $sb_{\text{chain}}^{\langle f_{common}, f_{adj} \rangle}$ viewing both as conjunctions of lex-constraints.

Example 7. Consider the unknown graph G of order 4 and the invariant f_{common}^G detailed in Example 6. The following are the lex-constraints (after simplification) deriving from the transposition $\pi_{1,2}$. The first is from sb_{lex} and the second is from $sb_{\text{chain}}^{\langle f_{common}, f_{adj} \rangle}$:

$$[x_2, x_3] \leq_{\text{lex}} [x_4, x_5] \tag{4}$$

$$[y_2, y_3, x_2, x_3] \leq_{\text{lex}} [y_4, y_5, x_4, x_5] \tag{5}$$

In Eq. (4) all prefixes (of the sequences in the comparison) relate only to x variables, from the unknown adjacency matrix. In Eq. (5) all prefixes involve y variables from the graph invariant. Assignments for the y variables do not necessarily restrict the possible consistent values for the x variables because each y variable is defined in terms of a set of the x variables (see Example 6). □

The third part of Table 1 will be described later in the paper. First, we seek new ways to combine graph invariants that result in symmetry breaking constraints that improve on sb_{lex} in both efficiency and precision.

Let us first clarify notation. Let $\langle f_1, \ldots, f_m \rangle$ be a sequence of k-dimensional graph invariants. Recall that for a given graph G and each invariant f_i, f_i^G is a function which maps sets of k vertices to integer values. The Cartesian product, $f_1^G \times \ldots \times f_m^G$ of these functions maps each set S of k vertices to a tuple of integers, $\langle f_1^G(S), \ldots f_m^G(S) \rangle$. As demonstrated in Example 8, the product, $f_1^G \times \ldots \times f_m^G$, can also be viewed as a k dimensional matrix of tuples.

Definition 4 *(The product ordering).* *Let $\langle f_1, \ldots, f_m \rangle$ be a sequence of graph invariants of dimension k. Then, for any two graphs $G, H \in \mathcal{G}_n$, we say that $G \preceq_{f_1 \times \ldots \times f_m} H$ if and only if $vec(f_1^G \times \ldots \times f_m^G) \leq_{lex} vec(f_1^H \times \ldots \times f_m^H)$.*

Definition 5 *(The product constraint)***.** *Let G be an unknown graph of order n and let $\langle f_1, \dots, f_m \rangle$ be a sequence of k-dimensional graph invariants. Then, the product symmetry breaking constraint induced by $\langle f_1, \dots, f_m \rangle$ is*

$$sb_{prod}^{f_1,\dots,f_m}(G) = \bigwedge_{\pi \in T_n} G \preceq_{f_1 \times \dots \times f_m} \pi(G)$$

One can check that, in general, the product ordering is a weak order on graphs. Hence, by Observation 1, $sb_{prod}^{f_1,\dots,f_m}(G)$ is a partial symmetry breaking constraint.

The following example demonstrates the construction of the product symmetry breaking constraint for the sequence of invariants $\langle f_{\text{adj}}, f_{\text{common}} \rangle$.

Example 8. Consider the unknown graph G of order 4 and the invariant f^G_{common} as detailed in Example 6. Recall that the x variables are from the adjacency matrix, and the y variables are from the graph invariant. Then,

$$f^G_{\text{adj}} \times f^G_{\text{common}} = \begin{bmatrix} - & \langle x_1, y_1 \rangle & \langle x_2, y_2 \rangle & \langle x_3, y_3 \rangle \\ \langle x_1, y_1 \rangle & - & \langle x_4, y_4 \rangle & \langle x_5, y_5 \rangle \\ \langle x_2, y_2 \rangle & \langle x_4, y_4 \rangle & - & \langle x_6, y_6 \rangle \\ \langle x_3, y_3 \rangle & \langle x_5, y_5 \rangle & \langle x_6, y_6 \rangle & - \end{bmatrix}$$

The product symmetry breaking constraint induced from $\langle f_{\text{adj}}, f_{\text{common}} \rangle$ consists of 6 constraints of the form $G \preceq_{f_{\text{adj}} \times f_{\text{common}}} \pi_{i,j}(G)$, one for each transposition $\pi_{i,j}$. Each of these can be expressed as a lex-constraint. The lex-constraint corresponding to $\pi_{1,2}$ is:

$$[x_1, y_1, \dots, x_6, y_6] \leq_{\text{lex}} [x_1, y_1, x_4, y_4, x_5, y_5, x_2, y_2, x_3, y_3, x_6, y_6]$$

The vector on the left of the constraint consists of the variables of the matrix representing the product of the two invariants. The vector on the right consists of the variables from the permuted matrix obtained by swapping rows 1 and 2 as well as columns 1 and 2. This constraint further simplifies to:

$$[x_2, y_2, x_3, y_3] \leq_{\text{lex}} [x_4, y_4, x_5, y_5]$$

Note that the variable order in this constraint interleaves the x variables from the adjacency matrix and the y variables from the invariant whilst in the chain constraint (see Example 7) the adjacency matrix variables occur at the end of the vector. This is a property of the product constraint, that the variables of each invariant gets a "fair" place in the vectors occurring in the lex-constraints. We conjecture that interleaving allows for better propagation. The third part of Table 1 supports this conjecture, at least in the sence that computations are considerably more efficient than with the chain constraint. $\square$

The third part of Table 1 (titled "product"): details the computation of graphs applying symmetry breaking constraints based on the product constraint. This part consists of four rows, each row describes the computation using the specified sequence of invariants as a product. Overall, the product constraints are more efficient than those using the chain constraint. The product symmetry breaking constraint induced from $\langle f_{\text{common}}, f_{\text{adj}} \rangle$ is more precise and faster than the corresponding induced chain symmetry breaking constraint. The product symmetry breaking constraint induced from $\langle f_{\text{adj}}, f_{\text{common}} \rangle$ is slightly less precise than that induced from $\langle f_{\text{common}}, f_{\text{adj}} \rangle$ but it is much faster. For example, when $n = 9$, it is about 5% less precise but is about 80 times faster. Moreover, all of the illustrated product constraints allow to generate the order 10 graphs within the 24 h timeout.

The next section demonstrates the advantage of using the (product) combination of graph invariants when solving graph search problems related to specific classes where knowledge about the structure of the graphs can be exploited to select invariants.

5 An Application: Generation of Cubic Graphs

This section demonstrates the application of symmetry breaking constraints induced from graph invariants to generate a specific class of cubic graphs. Cubic graphs are such that each vertex has degree 3. The class of cubic graphs is well studied, and many papers address the problem of generating small cubic graphs [2,3,16]. Brinkmann *et al.* [3] introduce a generation method which allows to generate all non-isomorphic connected cubic graphs for up to 32 vertices. In [21], the authors compute the number of cubic graphs for graphs of orders $n \leq 40$. However, their technique is non-constructive. That is, it allows to count the graphs but not to generate them. The number of cubic graphs is humongous [19] (sequence A005638 of the OEIS). For example, there are 8,832,736,318,937,756,165 cubic graphs of order 40.

We consider the problem of generating all connected cubic graphs, which are also "claw-free". A graph is called claw-free if it contains no $K_{1,3}$ as an induced subgraph. For cubic graphs the condition for being claw-free is equivalent to the requirement that each vertex participates in a triangle [14]. The number of connected cubic claw-free graphs for order $n \leq 30$ is specified in the OEIS as sequence A084656 [19]. Using our constraint based approach with symmetry breaking constraints based on various combinations of graph invariants, we were able to extend this sequence for $n \leq 36$. It is important to note that one cannot generate the sets of order n connected cubic claw-free graphs simply by testing the corresponding sets of cubic graphs. While the latter have been generated for $n \leq 32$, their sheer number is humongous.

Figure 5 details the constraint model we apply to generate order n cubic claw-free connected graphs. The variables $G_{i,j}$ are the Boolean variables of the unknown order n graph G. Equation (5) constrains the degree of each vertex to be 3. Equation (5) constrains the graph to be claw-free (each vertex must occur in a triangle). Finally, Eq. (5) constrains the graph G to be connected, using an encoding of the Floyd-Warshall shortest paths algorithm [10]. The variables $p^k_{i,j}$ indicate whether there is a path between vertices i and j in which intermediate vertices are from the set $\{1 \dots k\}$. The left conjunct specifies that $p^0_{i,j}$ is true if and only if there is an edge between i and j. The center conjunct specifies the variables $p^k_{i,j}$ for $1 \leq k \leq n$, encoding the recursive part of the Floyd Warshall algorithm. The right conjunct ensures that there is a path in the graph between every two vertices.

$$\bigwedge_{1 \leq i \leq n} \sum_{1 \leq j \leq n} G_{i,j} = 3 \tag{6}$$

$$\bigwedge_{1 \leq i \leq n} \bigvee_{1 \leq j,k \leq n} (G_{i,j} \wedge G_{i,k} \wedge G_{j,k}) \tag{7}$$

$$\bigwedge_{1 \leq i,j \leq n} (p^0_{i,j} \leftrightarrow G_{i,j}) \wedge \bigwedge_{1 \leq i,j,k \leq n} p^k_{i,j} \leftrightarrow (p^{k-1}_{i,j} \vee (p^{k-1}_{i,k} \wedge p^{k-1}_{k,j})) \wedge \bigwedge_{1 \leq i,j \leq n} p^n_{i,j} \tag{8}$$

Fig. 5. The constraint model for connected cubic claw-free graphs.

In our experimentation, we applied chain combinations involving f_{adj} with one additional graph invariant from the set $\{f_{\text{common}}, f_{\text{triangles}}\}$ (and their inverses). We applied product combinations of f_{adj} with f_{common} and its inverse. For each approach (chain and product), we report the results for the symmetry breaking constraint that exhibit the best (time) performance. For comparison, we also apply the state-of-the-art partial symmetry breaking constraint, sb_{lex}.

Table 2 details the computation of claw-free cubic graphs. The first column specifies the order of the graphs. The second column details the number of non-isomorphic solutions. The next three columns detail the solving time and number of solutions for each symmetry breaking method. All times reported are CPU running times and specified in an appropriate unit: (s) seconds, (m) minutes, or (h) hours where we apply a timeout (TO) of 24 h. The numbers of non isomorphic solutions as specified in the second column are obtained by filtering isomorphic representations from the set of solutions using nauty [15]. The numbers below the solid line for $n \geq 32$ are new.

The results in Table 2 clearly show that symmetry breaking based on the product constraint, $f_{\text{adj}} \times -f_{\text{common}}$, is superior in both computation time and precision to the other techniques. This approach allows us to generate all solutions up to order 36, thus extending the OEIS sequence A084656 [19] with three new values.

Table 2. Generating connected cubic claw-free graphs for orders $4 \leq n \leq 36$.

n	graphs	sb_{lex}		$\langle -f_{\text{common}}, f_{\text{adj}} \rangle$		$f_{\text{adj}} \times -f_{\text{common}}$	
		time	sols	time	sols	time	sols
4	1	0.00 s	1	0.00 s	1	0.00 s	1
6	1	0.00 s	1	0.00 s	1	0.00 s	1
8	1	0.00 s	2	0.05 s	4	0.00 s	1
10	1	0.02 s	7	0.40 s	3	0.03 s	4
12	3	0.09 s	24	1.25 s	10	0.07 s	3
14	3	0.52 s	188	3.23 s	17	0.15 s	10
16	5	1.87 s	1,134	12.60 s	58	0.28 s	28
18	11	14.58 s	7,293	26.08 s	100	0.72 s	44
20	15	3.39 m	61,391	37.17 s	280	2.11 s	132
22	27	2.28 h	546,409	2.29 m	716	3.86 s	307
24	54	T.O	–	5.66 m	1,551	10.96 s	660
26	94	T.O	–	5.25 m	4,384	45.37 s	1,835
28	181	T.O	–	50.76 m	10,883	2.57 m	4,372
30	369	T.O	–	46.00 m	26,778	6.60 m	10,567
32	731	T.O	–	1.70 h	75,303	24.28 m	29,069
34	1,502	T.O	–	T.O	–	1.12 h	72,501
36	3,187	T.O	–	T.O	–	8.98 h	188,495

6 Conclusion

This paper explores the application of high dimensional invariants to define symmetry breaking constraints. To the best of our knowledge, this is the first time graph invariants of dimension higher than one have been applied in symmetry breaking constraints. We introduce two techniques to obtain symmetry breaking constraints by combining graph invariants. First, we introduce the chain constraint which generalizes the standard approach for combining several properties when breaking symmetries. Then, after observing the poor performance of this technique, we introduce the product combination and demonstrate its superior performance. We demonstrate the application of the product constraint to extend the computation of cubic claw-free graphs for order $n \leq 36$ vertices.

While we focus on two dimensional invariants in examples and experiments, the same techniques apply for invariants of any dimension.

A Note for the Modeller: When solving a specific graph search problem, selecting which invariants to combine is nontrivial. Two points to consider are: (1) properties of the graphs the problem seeks to find; and (2) the complexity of the invariants when expressed as a CNF. For example, when seeking regular graphs, one would not consider f_{deg} as all vertices have the same degree. Alternatives such as f_{common} and $f_{\text{triangles}}$ encode "similar" information on pairs of vertices. However, their encodings differ in complexity. Each variable of $f_{\text{triangles}}$ encodes the number of triangles that involve a vertex i. This expression is quadratic. In contrast, each f_{common} variable encodes the number of common neighbors of a pair, i and j. This expression is linear in size. Hence, one might prefer the latter.

Acknowledgement. We thank the anonymous reviewers of this paper for their constructive suggestions.

References

1. Balaban, A.T., Balaban, T.S.: New vertex invariants and topological indices of chemical graphs based on information on distances. J. Math. Chem. **8**(1), 383–397 (1991). https://doi.org/10.1007/BF01166951
2. Brinkmann, G.: Fast generation of cubic graphs. J. Graph Theory **23**(2), 139–149 (1996)
3. Brinkmann, G., Goedgebeur, J., McKay, B.D.: Generation of cubic graphs. Discret. Math. Theor. Comput. Sci. **13**, 69–80 (2011)
4. Cameron, R., Colbourn, C., Read, R., Wormald, N.C.: Cataloguing the graphs on 10 vertices. J. Graph Theory **9**(4), 551–562 (1985)
5. Codish, M., Gange, G., Itzhakov, A., Stuckey, P.J.: Breaking symmetries in graphs: the nauty way. In: Rueher, M. (ed.) CP 2016. LNCS, vol. 9892, pp. 157–172. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44953-1_11
6. Codish, M., Miller, A., Prosser, P., Stuckey, P.J.: Breaking symmetries in graph representation. In: Rossi, F. (ed.) Proceedings of the 23rd International Joint Conference on Artificial Intelligence, IJCAI 2013, Beijing, China, 3–9 August 2013, pp. 510–516. IJCAI/AAAI (2013). https://ijcai.org/proceedings/2013
7. Codish, M., Miller, A., Prosser, P., Stuckey, P.J.: Constraints for symmetry breaking in graph representation. Constraints **24**(1), 1–24 (2018). https://doi.org/10.1007/s10601-018-9294-5
8. Crawford, J.M., Ginsberg, M.L., Luks, E.M., Roy, A.: 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 1996), Cambridge, Massachusetts, USA, 5–8 November 1996, pp. 148–159. Morgan Kaufmann (1996)
9. da F. Costa, L., Rodrigues, F.A., Travieso, G., Boas, P.R.V.: Characterization of complex networks: a survey of measurements. Adv. Phys. **56**(1), 167–242 (2007). https://doi.org/10.1080/00018730601170527
10. Floyd, R.W.: Algorithm 97: shortest path. Commun. ACM **5**(6), 345 (1962). https://doi.org/10.1145/367766.368168
11. Frisch, A.M., Harvey, W.: Constraints for breaking all row and column symmetries in a three-by-two matrix. In: Proceedings of SymCon 2003 (2003)
12. Gebser, M., Kaufmann, B., Schaub, T.: Conflict-driven answer set solving: from theory to practice. Artif. Intell. **187**, 52–89 (2012)
13. Harary, F.: Graph Theory. Addison-Wesley, Reading (1969)
14. Hong, Y., Liu, Q., Yu, N.: Edge decomposition of connected claw-free cubic graphs. Discret. Appl. Math. **284**, 246–250 (2020). https://doi.org/10.1016/j.dam.2020.03.040

15. McKay, B.D., Piperno, A.: Practical graph isomorphism, II. J. Symb. Comput. **60**, 94–112 (2014)
16. Meringer, M.: Fast generation of regular graphs and construction of cages. J. Graph Theory **30**(2), 137–146 (1999)
17. Metodi, A., Codish, M., Stuckey, P.J.: Boolean equi-propagation for concise and efficient SAT encodings of combinatorial problems. J. Artif. Intell. Res. (JAIR) **46**, 303–341 (2013)
18. Miller, A., Prosser, P.: Diamond-free degree sequences. CoRR abs/1208.0460 (2012). https://arxiv.org/abs/1208.0460
19. The on-line encyclopedia of integer sequences (OEIS) (2010). Published electronically at https://oeis.org
20. Read, R.C.: Every one a winner or how to avoid isomorphism search when cataloguing combinatorial configurations. Ann. Discret. Math. **2**, 107–120 (1978)
21. Robinson, R.W., Wormald, N.C.: Numbers of cubic graphs. J. Graph Theory **7**(4), 463–467 (1983). https://doi.org/10.1002/jgt.3190070412
22. Shlyakhter, I.: Generating effective symmetry-breaking predicates for search problems. Discret. Appl. Math. **155**(12), 1539–1548 (2007)
23. Walsh, T.: General symmetry breaking constraints. In: Benhamou, F. (ed.) CP 2006. LNCS, vol. 4204, pp. 650–664. Springer, Heidelberg (2006). https://doi.org/10.1007/11889205_46