Incremental Symmetry Breaking Constraints for Graph Search Problems

Avraham Itzhakov, Michael Codish
Department of Computer Science
Ben-Gurion University of the Negev
{itzhakoa, mcodish}@cs.bgu.ac.il

Abstract

This paper introduces incremental symmetry breaking constraints for graph search problems which are complete and compact. We show that these constraints can be computed incrementally: A symmetry breaking constraint for order n graphs can be extended to one for order $n+1$ graphs. Moreover, these constraints induce a special property on their canonical solutions: An order n canonical graph contains a canonical subgraph on the first k vertices for every $1 \leq k \leq n$. This facilitates a "generate and extend" paradigm for parallel graph search problem solving: To solve a graph search problem φ on order n graphs, first generate the canonical graphs of some order $k < n$. Then, compute canonical solutions for φ by extending, in parallel, each canonical order k graph together with suitable symmetry breaking constraints. The contribution is that the proposed symmetry breaking constraints enable us to extend the order k canonical graphs to order n canonical solutions. We demonstrate our approach through its application on two hard graph search problems.

1 Introduction

Graph search problems deal with existence and enumeration of simple graphs with certain properties which are invariant under isomorphism. One of the most famous graph search problems is the search for Ramsey $(s, t; n)$ graphs which seeks order n graphs with no clique of size s and no independent set of size t (Radziszowski 1994). The set of Ramsey $(4, 5; 24)$ graphs was determined only recently (Angeltveit and McKay 2018). Such problems are often challenging due to the large number of symmetries in graph representations, and enormous search space. For graph search problems, each isomorphism class of graphs consists of either symmetric solutions or symmetric non-solutions.

Ultimately, symmetry breaking is about restricting the search to a reduced space which considers a single graph from each isomorphism class. If symmetries are eliminated, the size of the search space is significantly reduced, and it can be explored more efficiently because paths that lead to symmetric (non-)solutions are avoided.

Symmetry breaking in constraint programming and satisfiability solving is often achieved by introducing symmetry breaking constraints (Crawford et al. 1996; Shlyakhter 2001; Walsh 2006; Shlyakhter 2007) which are satisfied by at least one member of each isomorphism class. A symmetry breaking constraint is called *complete* if it is satisfied by exactly one member from each isomorphism class and *partial* otherwise. Ideally, a symmetry breaking constraint should be compact in size, and complete. This enables solvers to avoid symmetries without imposing an overhead due to the size of the constraint.

Computing compact and complete symmetry breaking constraints is, most often, intractable (Crawford et al. 1996). A universal measure for the size of a symmetry breaking constraint is the number of clauses in its CNF representation. For graph search problems, it is unknown if there exists a complete symmetry breaking constraint that is polynomial in the size of the graph. The well-known lex-leader approach (Read 1978), selects the smallest member of each class, with respect to a lexicographic ordering, as a canonical representative. Testing if a given graph is a lex-leader canonical representative is known to be co-NP complete (Luks and Roy 2004). Hence, it is unlikely that there exists a polynomial size symmetry breaking constraint that identifies lex-leader canonical representatives.

In theory, a complete lex-leader symmetry breaking constraint should impose one lexicographic order constraint for every symmetry. As an example, for order 10 graphs this translates to $10! = 3{,}628{,}800$ constraints. In practice many of the these constraints are redundant. Itzhakov and Codish (2016) compute a complete symmetry breaking constraint for order 10 graph search problems consisting of only $7{,}853$ lexicographic order constraints instead of all $10!$ constraints. Codish *et al.* (2018) show that a further reduction is made when expressing the symmetry breaking constraint using the implications derived from the AND-decomposition of the lexicographic order constraints (Frisch et al. 2006). In their approach, symmetry breaking constraints are more compact and faster to compute. They compute, for the first time, a complete and compact symmetry breaking constraint for order 11 graph search problems. Heule (2016; 2019) computes complete symmetry breaking constraints which are optimal in size for graphs up to order 5, and small for graphs up to order 8.

This paper introduces incremental symmetry breaking constraints for graph search problems which are complete, compact and have two special properties: First, the symme-

try breaking constraint for graphs of order n can be extended to one for graphs of order $n+1$. Second, if an order n graph satisfies the symmetry breaking constraint, then so does its subgraph on the first $k \leq n$ vertices. The first property implies that symmetry breaking constraints can be computed incrementally. The second property facilitates a *"generate and extend"* paradigm for parallel graph search problem solving: to solve an order n graph search problem φ, first generate the canonical graphs of some order $k < n$. Then compute canonical solutions for φ by extending, in parallel, each canonical graph of order k, applying a corresponding symmetry breaking constraint for order n graphs. The crucial point is that these symmetry breaking constraints are consistent with the order k canonical subgraphs. We show that this generate and extend paradigm can be effectively applied for order $n \leq 12$ graph search problems.

We demonstrate the application of incremental symmetry breaking constraints on two hard graph search problems: enumeration of "totally magic-" (Exoo et al. 2002; Gallian 2018) and "word-representable-" (Kitaev and Pyatkin 2018; Akgün et al. 2019) graphs. For both, state-of-the-art solutions apply a generate and test approach where each graph is tested for the corresponding property. Solving the order 11 instances involves huge resources and thousands of CPU days. Moreover, this approach cannot be applied for larger graphs. We show that a generate and extend approach with complete symmetry breaking constraints is significantly more efficient.

The computations described in this paper are performed using the finite-domain constraint compiler BEE (Metodi, Codish, and Stuckey 2013) which compiles constraints to a CNF, and solves it applying an underlying SAT solver (we use Glucose 4.0 (Audemard and Simon)). All computations were performed on a cluster of servers, each with 56 Intel Xeon E5-2620 cores and 256GB of RAM memory, clocked at 2 GHz. Each SAT instance is run on a single thread. All running times reported are CPU times and specified in an appropriate unit: (s) seconds, (h) hours, (d) days or (y) years.

2 Preliminaries

Lexicographic Order Constraints: The lexicographic order constraint between two vectors $\bar{x} = \langle x_1, \ldots, x_n \rangle$ and $\bar{y} = \langle y_1, \ldots, y_n \rangle$, each consisting of n finite domain variables, is denoted $\bar{x} \leq_{lex} \bar{y}$. The AND-decomposition of a lexicographic order constraint (Frisch et al. 2006) can be expressed as follows:

$$\bar{x} \leq_{lex} \bar{y} = \bigwedge_{k=1}^{n} imp_k(\bar{x}, \bar{y}) \tag{1}$$

where each of the conjuncts $imp_k(\bar{x}, \bar{y})$ is called a k-length lex-implication and is defined by:

$$imp_k(\bar{x}, \bar{y}) = ((\bigwedge_{i=1}^{k-1} x_i = y_i) \Rightarrow x_k \leq y_k) \tag{2}$$

Permutations: We denote by S_n the group of all permutations on $\{1 \ldots n\}$. We represent a permutation $\pi \in S_n$ as an array of size n where the number $1 \leq i \leq n$ is mapped to $\pi(i)$. For example: the permutation $[2, 3, 1] \in S_3$ maps as follows: $\{1 \mapsto 2, 2 \mapsto 3, 3 \mapsto 1\}$.

Graphs and Graph Orderings: The set of simple graphs on n vertices is denoted $\mathcal{G}_n$. The vertex set of a graph $G = (V, E)$ of order n, is assumed to be $V = \{1, \ldots, n\}$ and in abuse of notation its adjacency matrix representation is also denoted G. We denote by $R(G)$ and by $C(G)$ the strings obtained by respectively concatenating the rows and columns of the upper triangular part of the adjacency matrix of G. We denote by $G^{(k)}$ for $k \leq n$ the induced subgraph of G on the vertex set $\{1, \ldots, k\}$. This is the upper left $k \times k$ corner of the adjacency matrix of G. An unknown graph of order n is represented as an $n \times n$ adjacency matrix of Boolean variables which is symmetric and has values $false$ (denoted by 0) on the diagonal. All of the notations for given graphs, such as $C(G)$, $R(G)$ and $G^{(k)}$, hold also for unknown graphs. For simplicity, unknown graphs are also called graphs. For (possibly unknown) graphs G, H of the same order and $X \in \{R, C\}$, we denote the lexicographic order and the k-length lex-implication constraints with respect to X by:

$$\begin{aligned} G \leq^X_{lex} H &\equiv X(G) \leq_{lex} X(H) \\ imp^X_k(G, H) &\equiv imp_k(X(G), X(H)). \end{aligned}$$

Permutations act on graphs in the natural way: viewing $G \in \mathcal{G}_n$ as an adjacency matrix and 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 $1 \leq i, j \leq n$). Two graphs $G, H \in \mathcal{G}_n$ are called isomorphic if there exists a permutation $\pi \in S_n$ such that $G = \pi(H)$.

Example 1 The following depicts an unknown graph G and its permutation $\pi(G)$, for $\pi = [2, 1, 3, 5, 4]$, both represented as adjacency matrices of Boolean variables.

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

The strings $R(G)$ and $C(G)$ are obtained by respectively concatenating the rows and the columns of the upper triangular part of G.

$$\begin{aligned} R(G) &= [x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}] \\ C(G) &= [x_1, x_2, x_5, x_3, x_6, x_8, x_4, x_7, x_9, x_{10}] \end{aligned}$$

The constraints $G \leq^R_{lex} \pi(G)$ and $G \leq^C_{lex} \pi(G)$ are:

$$\begin{aligned} &[x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}] \leq_{lex} \\ &\quad [x_1, x_5, x_7, x_6, x_2, x_4, x_3, x_9, x_8, x_{10}] \text{ and} \\ &[x_1, x_2, x_5, x_3, x_6, x_8, x_4, x_7, x_9, x_{10}] \leq_{lex} \\ &\quad [x_1, x_5, x_2, x_7, x_4, x_9, x_6, x_3, x_8, x_{10}]. \end{aligned}$$

These can be simplified respectively (per constraint, see for example (Frisch and Harvey 2003)) to:

$$\begin{aligned} [x_2, x_3, x_4, x_8] &\leq_{lex} [x_5, x_7, x_6, x_9] \text{ and} \\ [x_2, x_3, x_6, x_8] &\leq_{lex} [x_5, x_7, x_4, x_9]. \end{aligned}$$

The lex-implication constraint $imp_3^R(G, \pi(G))$ is

$$(x_1 = x_1) \wedge (x_2 = x_5) \quad \Longrightarrow \quad x_3 \leq x_7.$$

Given an isomorphism class of a graphs, a classic way to define the canonical representative of the class is to take the smallest graph with respect to some order. In this paper we consider two specific order relations and define a canonical representative in the following way.

Definition 1 [LEXLEADER] We say that $G \in \mathcal{G}_n$ is row-wise canonical if the following constraint holds for $X = R$, and column-wise canonical if the following constraint holds for $X = C$.

$$\text{LEXLEADER}_X(G) \equiv \bigwedge_{\pi \in S_n} G \leq_{lex}^X \pi(G)$$

The following property of column-wise canonical graphs is stated in (Kvasnička and Pospíchal 1990).

Theorem 1 (column-wise canonical subgraphs) *If $G \in \mathcal{G}_n$ is column-wise canonical then $G^{(k)}$ is also column-wise canonical for any $1 \leq k \leq n$.*

The following example demonstrates that Theorem 1 does not hold for row-wise canonical graphs.

Example 2 The following graph G (on the left) is row-wise canonical, while its subgraph $G^{(6)}$ (bold text) is not; G' (on the right) is the row-wise canonical isomorph of $G^{(6)}$.

$$G = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 1 & 1 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & 0 & 1 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & 1 & 1 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & 1 & 1 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & 1 & 0 \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \qquad G' = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

Graph Search Problems: Graph search problems are about the search for a graph that satisfies certain graph properties which are invariant under isomorphism. If G is a solution to a graph search problem, then so is any G' that is isomorphic to G. More formally, 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.

Symmetry Breaking: We focus in this paper on two particular types of complete symmetry breaking predicates: row-wise and column-wise, that are satisfied exactly by the row-wise and column-wise canonical graphs, respectively.

Consider the two predicates $\text{LEXLEADER}_X(G)$ for $X \in \{R, C\}$ from Definition 1. When G is an unknown graph, expressed in terms of Boolean variables, then Definition 1 can be viewed as specifying a conjunction of lexicographic order constraints over these variables. Each of these two conjunctions specifies, a predicate that is true exactly when its argument graph is respectively row-wise or column-wise canonical. Hence, these predicates are complete symmetry breaking constraints. We can view the constraints in $\text{LEXLEADER}_X(G)$, either as a set of lexicographic order constraints, or as a set of their corresponding lex-implications as specified by Equation 1. We denote these sets, respectively by $\text{LEXLEADER}_X^{lex}(G)$ and by $\text{LEXLEADER}_X^{imp}(G)$. For any symmetry breaking predicate ψ defined as a conjunction of lexicographic order constraints (or lex-implications), we define the size of ψ to be the number of lex-implications in the AND-decomposition of these constraints. This measure is polynomial in the size of the CNF representation of ψ.

Algorithm 1 Compute a compact complete symmetry break for $X \in \{R, C\}$, $Y \in \{lex, imp\}$

procedure $\text{SYMBREAK}_X^Y(G)$
Input: unknown order n graph G
Output: compact complete symmetry breaking constraint
$\psi \leftarrow \{\}$
while $(\exists c \in LexLeader_X^Y(G)$ s.t. $\psi \not\Longrightarrow c)$ **do**
$\psi \leftarrow \psi \cup \{c\}$
return $Reduce(\psi)$

Computing Compact Symmetry Breaking Constraints: For an unknown graph G, for $X \in \{R, C\}$, and for $Y \in \{lex, imp\}$, the set of constraints $\text{LEXLEADER}_X^Y(G)$ (over the variables in G) is of size exponential in the size of G. However, many of these constraints are redundant (implied by the others). One can compute an equivalent set of irredundant constraints which is more concise. From here on, we call a set of irredundant constraints, *compact*. Luks (2004) proves a result from which it follows that unless $\mathtt{P} = \mathtt{NP}$, there is no polynomial-time algorithm that computes a compact lex-leader symmetry breaking constraint for graph search problems. Nevertheless, we aim to compute compact lex-leader symmetry breaking constraints for small graph search problems. Algorithm 1 computes a compact complete symmetry breaking constraint for G given X and Y. The symmetry breaking constraint is computed iteratively by adding a constraint c from $\text{LEXLEADER}_X^Y(G)$, as long as it is not implied by the constraints selected so far. In the implementation, the SAT solver identifies a constraint c which is not implied by the current set of constraints ψ (the condition in the while loop). Possibly, when a new constraint is added, a constraint, already present, becomes redundant. Therefore, the Algorithm applies an operation, $Reduce(\psi)$, to remove redundant constraints. For each constraint the test of redundancy is performed using a SAT-solver. This algorithm generalizes the ones presented in (Itzhakov and Codish 2016) and in (Codish et al. 2018).

Example 3 Let G be the unknown order 5 graph detailed in Example 1. Algorithm 1 (with $X = R, Y = lex$) computes a compact row-wise symmetry breaking constraint consisting of the following 7 lexicographic order constraints (after some simplifications, as in Example 1 instead of the 120 constraints in $\text{LEXLEADER}_R^{lex}(G)$.

$$\begin{array}{l} [x_1,x_6,x_7] \leq [x_2,x_8,x_9] \\ [x_2,x_5,x_9] \leq [x_3,x_6,x_{10}] \\ [x_1,x_2,x_3,\ldots,x_9] \leq [x_5,x_1,x_7,x_6,x_2,x_9,x_8,x_4,x_3] \\ [x_2,x_3,x_4,\ldots,x_{10}] \leq [x_4,x_2,x_3,x_7,x_5,x_6,x_9,x_{10},x_8] \\ [x_1,x_2,x_3,x_5,x_6,x_7,\ldots,x_{10}] \leq [x_9,x_{10},x_7,x_8,x_5,x_2,x_6,x_3,x_1] \\ [x_2,x_3,x_4,x_5,\ldots,x_{10}] \leq [x_7,x_5,x_6,x_4,x_2,x_3,x_9,x_{10},x_8] \\ [x_1,x_2,x_3,\ldots,x_{10}] \leq [x_6,x_1,x_7,x_5,x_3,x_{10},x_8,x_4,x_2,x_9] \end{array}$$

3 Incremental Computation of Symmetry Breaking Constraints

In this section we describe an incremental computation of compact column-wise symmetry breaking constraints for graphs. Our goal is to be able to extend a given compact column-wise symmetry breaking constraint ψ for graphs of order k, with a set of constraints Δ such that $\psi' = \psi \wedge \Delta$ is a compact column-wise symmetry breaking constraint for graphs of order $k+1$. We show that we can achieve this goal when focusing on column-wise symmetry breaking predicates. This ability to extend compact column-wise symmetry breaking constraints facilitates an incremental approach to compute them.

Theorem 2 *Let G be an order k unknown graph and let $\psi \subseteq \text{LEXLEADER}_C^{imp}(G^{(k-1)})$ be a compact column-wise symmetry breaking constraint for order $k-1$ graphs. Then, there exists $\Delta \subseteq \text{LEXLEADER}_C^{imp}(G)$ such that $\psi \wedge \Delta$ is a compact column-wise symmetry breaking constraint for order k graphs.*

Proof: Let G and ψ be as in the premise of the statement, and let $\Delta^* = \left\{x \in \text{LEXLEADER}_C^{imp}(G) \,|\psi \not\Rightarrow x\right\}$. By construction, $\psi \wedge \Delta^*$ is a column-wise symmetry breaking constraint for order k graphs. We show that any redundant implication in $\psi \wedge \Delta^*$ must be from Δ^*. Hence, we can obtain $\Delta \subseteq \Delta^*$ for which $\psi \wedge \Delta$ is a compact column-wise symmetry breaking constraint for order k graphs. Let $c \in \psi$ be redundant in $\psi \wedge \Delta^*$. Since ψ is compact, $\psi - \{c\}$ is not a column-wise symmetry breaking constraint for order $k-1$ graphs. Hence, there exists $H \in \mathcal{G}_{k-1}$ which is not column-wise canonical and for which $(\psi - \{c\})(H)$ is true. By choice of c, $(\psi - \{c\}) \wedge \Delta^*$ is a column-wise symmetry breaking constraint for G and hence, by Theorem 1, also a column-wise symmetry breaking constraint for order $k-1$ graphs. Hence, $((\psi - \{c\}) \wedge \Delta^*)(H)$ is false which implies that $\Delta^*(H)$ is false. So, Δ^* contains an implication x which is false, and hence depends only on variables from $G^{(k-1)}$. $x \in \Delta^*$ implies that $\psi \not\Rightarrow x$. This means that ψ not implies a lex-implication which depends only on variables from $G^{(k-1)}$. The existence of such an implication contradicts the assumption that ψ is a column-wise symmetry breaking constraint for order $k-1$ graphs. □

Algorithm 2 applies an incremental approach to compute a compact set of constraints equivalent to $\text{LEXLEADER}_C^{imp}(G)$. The input is an unknown graph G, the output is a compact column-wise symmetry breaking constraint expressed in terms of the variables in G. When execution enters the outer for-loop at line 5 with a value $1 \leq k \leq n$ we have already computed a compact column-wise symmetry breaking constraint ψ for order $k-1$ graphs. At this stage, the goal of the k-th iteration of the for-loop is to compute a set of constraints Δ such that $\psi \wedge \Delta$ is a column-wise symmetry breaking constraint for order k graphs. The while-loop at lines 7–8 computes a set Δ corresponding to Δ^* in the proof of Theorem 2. The condition in the while-loop at line 7 asks if there exists a constraint c which is a witness to the fact that $\psi \wedge \Delta$ is not yet column-wise for k. Such a witness (if one exists) is to be found in $\text{LEXLEADER}_C^{imp}(G^{(k)})$. In the implementation, to determine if $\psi \wedge \Delta$ is not yet column-wise for k, we seek a value i, an order k graph H, and a permutation $\pi \in S_k$ such that $\psi_i = (\psi \wedge \Delta)(H) \wedge \neg imp_i(C(H), C(\pi(H)))$ holds. If ψ_i holds then the constraint we add to Δ is $c = imp_i(C(G^{(k)}), C(\pi(G^{(k)})))$. Otherwise, $\psi \wedge \Delta$ is column-wise for k and we exit the while-loop. The key step in the implementation is to encode the search for ψ_i as a SAT instance. By Theorem 2 (and its constructive proof), all redundant constraints removed at line 9 are removed from Δ.

Algorithm 2 Incremental computation of compact column-wise symmetry breaking constraints

```
1:  procedure SYMBREAKINCREMENTAL(G)
2:      Input: unknown order n graph G
3:      Output: compact column-wise symmetry breaking
        constraint for G
4:      ψ ← {}
5:      for k := 1 to n do
6:          Δ ← {}
7:          while ( ∃c ∈ LEXLEADER_C^imp(G^(k))
                    such that ψ ∧ Δ ⇏ c ) do
8:              Δ ← Δ ∪ {c}
9:          ψ ← Reduce(ψ ∪ Δ)
10:     return ψ
```

Table 1 details the time to compute compact column-wise symmetry breaking constraints expressed in terms of lex-implications for order n graphs. The table compares the direct computation (using Algorithm 1) and the incremental computation (using Algorithm 2). All of the symmetry breaking constraints computed (direct and incremental) were verified by checking that the constraint for order n graphs renders the exact set of column-wise canonical order n graphs as solutions. The column labeled LEXLEADER_C^{imp} details the size of the LEXLEADER_C^{imp} constraint. The two columns labeled "direct" detail the computation (size and time) of the column-wise symmetry breaking constraint by application of Algorithm 1. The four columns labeled "incremental" detail the computation by application of Algorithm 2. The columns labeled Δ-size and Δ-time detail the size and computation time to extend the symmetry breaking constraint from the previous row. The columns labeled size and time detail aggregated size and time. The column labeled speedup details the ratio between the direct and aggregated incremental computation times.

The table clearly indicates that compact column-wise symmetry breaking constraints are much smaller than their $LexLeader_C^{imp}(G)$ counterparts which are logically equivalent. This is in line with the results of previous works (Itzhakov and Codish 2016; Codish et al. 2018). The table indicates that the incremental computation is more efficient (up to 2.2 times faster) and that the sizes of the constraints (direct and incremental) are similar.

	LEXLEADER_C^{imp}	direct		incremental				
n	size	size	time	Δ-size	Δ-time	size	time	speedup
3	18	3	0.00s	3	0.00s	3	0.00s	1.00
4	144	11	0.04s	7	0.01s	10	0.01s	4.00
5	1,200	23	0.07s	10	0.07s	20	0.08s	0.87
6	10,800	43	0.79s	22	0.72s	42	0.80s	0.98
7	105,840	130	8.61s	83	8.39s	125	9.19s	0.93
8	1,128,960	484	92.73s	342	75.26s	467	84.45s	1.09
9	13,063,680	2,845	799.54s	2,369	511.09s	2,836	595.54s	1.34
10	163,296,000	25,193	3.73h	22,436	1.94h	25,272	2.10h	1.77
11	2,195,424,000	289,698	12.39d	264,845	5.48d	290,117	5.56d	2.22

Table 1: Computation of compact column-wise symmetry breaking constraints for order $3 \leq n \leq 11$ graphs using direct and incremental approach.

	specialized			simplified	
n	time	avg size	max size	avg size	max size
8	0.47h	7.38	19	33.55	222
9	3.51h	26.57	75	225.15	1,792
10	24.06h	107.43	675	2,943.92	18,836
11	6.71d	873.03	15,433	58,561.44	257,121
12	65.68d	10,717.07	294,220	–	–

Table 2: Computation of compact column-wise symmetry breaking constraints for order $8 \leq n \leq 12$ graphs extending order 7 column-wise canonical graphs.

4 Generate and Extend

This sections introduces a "generate and extend" paradigm for graph search problems which derives from the incremental properties of column-wise symmetry breaking constraints. Let G be an order n unknown graph. To solve a graph search problem $\varphi(G)$, one can first generate all order $k < n$ canonical graphs (for a suitable value of k). Then, one can pose, for each order k canonical graph G', the question: does there exist an order n solution for $\varphi(G)$ which extends G'? Basically this means, fixing the variables of the subgraph $G^{(k)}$ to the values of G' before applying a constraint (or SAT) solver on $\varphi(G)$. The graph search problem: extend G' to a solution of $\varphi(G)$ is denoted $\varphi(G/G')$.

For example, there are 1044 order 7 canonical graphs. To solve an order n graph search problem $\varphi(G)$, one can seek solutions, in parallel, for the problems $\varphi(G/G')$ to extend each G' from these 1044 graphs.

The question is: how to break symmetries when solving $\varphi(G/G')$? There are two issues: (1) Given a graph $G' \in \mathcal{G}_k$, how to break symmetries and obtain only non-isomorphic solutions of $\varphi(G/G')$; and (2) Given a pair of graphs $G', G'' \in \mathcal{G}_k$, how to ensure that solutions in $\varphi(G/G')$ are not isomorphic to those in $\varphi(G/G'')$. The beauty of column-wise symmetry breaking constraints is that they address both issues.

If G', of order k, is column-wise canonical, then G' is consistent with a column-wise symmetry breaking constraint ψ of order n ($k < n$). Therefore ψ can be applied when solving $\varphi(G/G')$ and all solutions are column-wise canonical. Given column-wise canonical graphs $G', G'' \in \mathcal{G}_k$, the solutions for $\varphi(G/G')$ and $\varphi(G/G'')$ are column-wise canonical, and hence, by definition, they cannot be isomorphic.

When solving graph search problems of the form $\varphi(G/G')$ for a given column-wise canonical $G' \in \mathcal{G}_k$, we can apply the column-wise symmetry breaking constraints (direct or incremental) described in Table 1. Alternatively, we can compute a specialized column-wise symmetry breaking constraint for each G'. These are considerably smaller and facilitate the computation of compact column-wise symmetry breaking constraints for order 12 graph search problems. This is done by application of Algorithm 2 after fixing the values corresponding to G' in the unknown graph G.

Table 2 details the computation of compact column-wise symmetry breaking constraints for graph search problems of the type $\varphi(G/G')$ where G' is one of the 1044 order 7 column-wise canonical graphs and G is of order $7 < n \leq 12$. The three columns labeled "specialized" detail the computation of specialized column-wise symmetry breaking constraint for all order 7 column-wise canonical graphs. We detail the total computation time (for all 1044 cases) and the average and maximal size of the individual symmetry breaking constraints. The two columns labeled "simplified" detail the size (average and maximal) of the symmetry breaking constraints obtained from the symmetry breaking constraints described in Table 1 by removing implications which become true due to G'.

Specialized symmetry breaking constraints are considerably smaller than the simplifications of the general counterparts described in Table 1. For example when $n = 11$ the general symmetry breaking constraint consists of 289,698 implications while the average (maximal) size of the specialized symmetry breaking constraints is only 873 (15,433). This means that each of the 1044 instances involve much smaller symmetry breaking constraints. The row for $n = 12$ details 1044 specialized symmetry breaking constraints for order 12 graphs. These cannot be computed by simplifying a general symmetry breaking constraint. This is the first time that a complete and compact lex-leader symmetry breaking constraint for graphs with 12 vertices has been computed, albeit distributed over 1044 cases.

The symmetry breaking constraints described in Table 2 apply when extending order 7 canonical graphs to order n

canonical solutions. It follows from Theorem 1 that these same constraints can also be applied when extending order $k > 7$ canonical graphs to canonical solutions of order n. In our experiments (in Section 5), when extending a canonical order $k > 7$ graph G' to an order n solution, we apply the symmetry breaking constraint computed for $G'^{(7)}$.

5 Two Applications of Generate and Extend

In this section we apply a generate and extend approach to solve two hard graph search problems: enumeration of "totally magic-" (Exoo et al. 2002; Gallian 2018) and "word-representable-" (Kitaev and Pyatkin 2018; Akgün et al. 2019) graphs. For both of these problems, state-of-the-art solutions apply a generate and test approach where each non-isomorphic order n graph is tested for the corresponding property. Each such test is not trivial. Determining if a given graph is word-representable is NP-complete (Halldórsson, Kitaev, and Pyatkin 2016). For totally magic graphs, the complexity is unknown. Yet state-of-the-art methods are exponential. Solving the instances for $n = 11$ involves huge resources and thousands of CPU days. Moreover, this approach cannot be applied for larger graphs. We apply the proposed generate and extend paradigm with column-wise symmetry breaking constraints and demonstrate that this approach is significantly more efficient.

Totally Magic Graphs: The n vertex graph search problem $\varphi_{tm(n)}(G)$ is about the search for a totally magic graph G with n vertices (Exoo et al. 2002; Gallian 2018). A graph $G = (V, E)$, with $|V| = n$ and $|E| = m$, is totally magic if there exist a one-to-one labeling $\lambda : V \cup E \rightarrow \{\ 1, \ldots, n+m\ \}$ and two integer values h, k such that:

vertex magic constraint: the sum of the labels of each node and its incident edges is h; and

edge magic constraint: the sum of the labels of each edge and its endpoints is k.

Figure 1 depicts a totally magic graph with 9 vertices. The sum of the labels of each node and its incident edges is 25. The sum of the labels of each edge and its endpoints is 26.

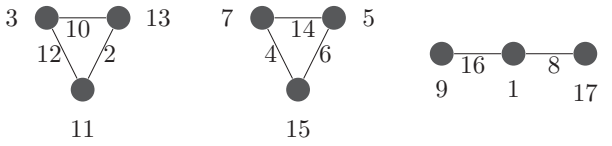

Figure 1: An order 9 totally magic graph.

A relaxation of $\varphi_{tm(n)}$ weakens the definition to consider the vertex and edge magic constraints with arithmetics modulo p and also specifies the number of edges, m, in the solutions. We denote the relaxed problem by $\varphi^{p}_{tm(n,m)}$. Any graph which is totally magic is also totally magic modulo p (Exoo et al. 2002; Jäger and Arnold 2015). So, we can test the solutions of the relaxed problem to identify the totally magic graphs.

	step 1: generate and extend for $\varphi^4_{tm(n)}$		step 2: test the $\varphi^4_{tm(n)}$ solutions	
n	solutions	time	solutions	time
8	1,777	3.44m	0	28.82s
9	37,542	2.16h	2	0.48h
10	1,507,843	4.63d	0	2.58d
11	91,397,498	550.74d	0	425.62d

Table 3: A two step computation of totally magic graphs for $9 \leq n \leq 11$ vertices.

Totally magic graphs are extremely rare. There are only 6 such graphs with 11 or fewer vertices. The only known totally magic graphs, with > 11 vertices, are composed of an odd number of triangles, or of an even number of triangles with a path of length 2. It is unknown if there exist other totally magic graphs with > 11 vertices.

In previous work, Jäger *et al.* (2015) enumerate all totally magic graphs with $n \leq 11$ vertices. Their approach is based on an enumeration of all non-isomorphic graphs with n vertices and testing each graph. These tests are based on, among other criteria, the elimination of graphs which are not totally magic modulo $p \leq 7$. Jäger *et al.* (2015) report a total of 13,595 CPU days to show that there do not exist any order 11 totally magic graphs.

We apply a constraint based approach where for each instance we consider the corresponding constraint model that expresses the totally magic constraints in conjunction with suitable symmetry breaking constraints.

We first applied a direct approach to solve instances of the form $\varphi_{tm(n)}(G)$ and $\varphi^4_{tm(n,m)}(G)$. For both types of instances we found solutions only when $n < 9$ (with a 48 hours time-out). We then applied a generate and extend approach focusing on the relaxed form $\varphi^4_{tm(n,m)}(G/G')$ which enabled us to enumerate all totally magic graphs of order $n < 12$.

Table 3 details a two step computation of totally magic order n graphs. In the first step we apply a generate and extend approach to compute all solutions of $\varphi^4_{tm(n,m)}$ (for all possible values of m). In the second step we test each solution of the relaxed problem to check if it is totally magic.

Each order n instance of the relaxed problem corresponds to a pair (G', m) where G' is one of the order 7 column-wise canonical graphs and m is the number of edges in the solution. We impose a 24 hour timeout on each instance. An instance (G', m) which timed out is further refined to a set of instances of the form (G'', m) where G'' is an order 8 column-wise canonical graph which extends G'. Such time-outs were encountered only for the case $n = 11$ (in 164 of the 36,540 instances).

For the first step, Table 3 details (left side) the total number of $\varphi^4_{tm(n,m)}$ solutions and the aggregated computation time (including the cost of the 24 hour time-outs). For the second step, Table 3 details (right side) the computation time for the tests on the solutions from the first step. To this end, we apply a series of tests similar to those described in (Jäger and Arnold 2015).

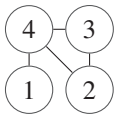

Figure 2: A word-representable graph (represented by the word 1413243).

Table 3 indicates for $n = 10$ and 11 total computation times of 7.21 and 976.36 (days) in contrast to the 21.70 and 13,595 (days) reported in (Jäger and Arnold 2015). One can view the first step (generate and extend) as a filter to the second step. For example, instead of testing all 1,018,997,864 order 11 graphs for the total magic property, as in (Jäger and Arnold 2015), we only have to test 91,397,498 which is less than 9% of them.

Word Representable Graphs: A simple graph $G = (V, E)$ is called word-representable if there exists a word $w \in V^*$ containing each letter of the alphabet V such that for every $i, j \in V$, i and j alternate in w if and only if $(i, j) \in E$ (Kitaev and Pyatkin 2018). Letters i and j alternate in a word w if between every consecutive pair of i there is a j, and between every consecutive pair of j there is an i. In this case we say that G is represented by w. For example, the graph depicted in Figure 2 is word-representable.

Akgün *et al.* (2019) compute the number of connected non-word-representable order $n \leq 11$ graphs. They adopt a generate and test approach, testing each non-isomorphic connected graph of the corresponding order. The test is performed using the constraint solver Minion (Gent, Jefferson, and Miguel 2006). To this end, they specify a constraint model based on the equivalence of word representable graphs and so-called, semi-transitive graphs (Kitaev and Pyatkin 2018). They report 1100 CPU days of computation time to accomplish this task for order 11 graphs.

We denote the graph search problem to find connected order n word-representable graphs by $\varphi_{wr(n)}$. We adopt the same constraint model used by Akgün *et al.* (2019), together with constraints to ensure connectivity, and with column-wise symmetry breaking constraints.

We first applied a direct constraint based approach to solve instances of the form $\varphi_{wr(n)}(G)$. This approach works well to find solutions for $n < 10$. We then apply a generate and extend paradigm to enumerate solutions of $\varphi_{wr(n)}(G/G')$. In this way we succeed to enumerate all connected word-representable graphs for order $n \leq 12$. This is the first time that a solution for $n = 12$ is reported.

Table 4 details the generate and extend approach and compares its computation times with those of the generate and test approach described by Akgün *et al.* (2019). To comply with their results, we present the corresponding numbers of connected and of connected non-word-representable graphs. The first three columns detail the order, n, and the numbers of connected graphs and connected non-word-representable graphs. For the generate and extend paradigm, with $9 \leq n \leq 12$, we extend each order k column-wise canonical graph to the set of its extensions which are canonical connected word-representable graphs. The total computation times are

			g & e		g & t
n	# connected	# non-word-rep.	k	time	time
8	11,117	929	7	5.80s	26s
9	261,080	54,957	7	1.55m	29m
10	11,716,571	4,880,093	8	1.41h	74h
11	1,006,700,565	650,853,916	9	4.04d	1,100d
12	164,059,830,476	135,950,114,622	9	6.40y	—

Table 4: Numbers of connected non-word-representable graphs computed using a generate and extend (g & e) approach, and a generate and test (g & t) approach.

detailed in the table. For the generate and test paradigm we specify (right most column) the computation times detailed in (Akgün et al. 2019). The generate and extend computations are orders of magnitude faster than the corresponding generate and test computations. We note that for order 11 graphs, the results reported in (Akgün et al. 2019) are in error: (1) the correct number of order 11 connected graphs is as specified in Table 4 (see sequence `A001349` in (Sloane, N. J. A., ed.)); and (2) the correct number of connected non-word-representable graphs is as specified in Table 4 (we found 2124 more connected word-representable graphs and verified that they are connected, word-representable, and all non-isomorphic). For $n = 12$, the entries in Table 4 are new. The generate and test approach is not able to handle this case. Due to the huge number of solutions for this case, BEE with Glucose is unable to generate all solutions. Instead, we apply the model counting capability of Clasp 3.1.3 (Gebser, Kaufmann, and Schaub 2012) to count the number of solutions.

6 Conclusion

This paper introduces incremental symmetry breaking constraints for graph search problems. We start from the notion of column-wise canonicity introduced in (Kvasničkа and Pospíchal 1990) where the authors show that the subgraph of an order n canonical graph on the first $k \leq n$ vertices is also canonical. We build on this property in two ways. First we show that column-wise symmetry breaking constraints can be computed incrementally. Then, we introduce a generate and extend paradigm where canonical solutions of an order n graph search problem can be obtained using column-wise symmetry breaking constraints by extending canonical graphs of order $k < n$. We compute, for the first time, a complete and compact lex-leader symmetry breaking constraint for order 12 graphs. We demonstrate the superiority of our generate and extend approach through two hard graph search problems and provide the previously unknown number of order 12 non-word-representable graphs.

There is a large body of work on generation of non-isomorphic combinatorial objects (McKay 1998; Colbourn and Read 1979). Still, there remain many open graph search problems which involve surprisingly small graphs. The results obtained in this paper suggest that a constraint-based approach combined with strong symmetry breaking methods will lead to breakthroughs for many of these problems.

Many graph search problems are hereditary: order n so-

lutions can be obtained by extending order $k < n$ solutions. The generate and extend approach can take advantage of this property by extending order k canonical solutions instead of all order k canonical graphs.

The techniques presented in this paper can be adapted for other combinatorial objects, such as matrix search problems (Flener et al. 2002). The symmetry breaking constraints described in this paper can be obtained from www.cs.bgu.ac.il/~mcodish/Papers/Tools/incrementalSBC. They are "solver independent", and can be applied in conjunction with any constraint solver to restrict the search to canonical solutions of a given graph search problem.

Acknowledgments. We acknowledge the support of the Israel Science Foundation, grant 625/17.

References

Akgün, Ö.; Gent, I.; Kitaev, S.; and Zantema, H. 2019. Solving computational problems in the theory of word-representable graphs. *Journal of Integer Sequences* 22(2):1–18.

Angeltveit, V., and McKay, B. D. 2018. R(5, 5) $\leq$ 48. *Journal of Graph Theory* 89(1):5–13.

Audemard, G., and Simon, L. *Glucose 4.0 SAT Solver*. http://www.labri.fr/perso/lsimon/glucose/.

Codish, M.; Ehlers, T.; Gange, G.; Itzhakov, A.; and Stuckey, P. J. 2018. Breaking symmetries with lex implications. In Gallagher, J. P., and Sulzmann, M., eds., *Functional and Logic Programming - 14th International Symposium, FLOPS 2018, Nagoya, Japan, May 9-11, 2018, Proceedings*, volume 10818 of *Lecture Notes in Computer Science*, 182–197. Springer.

Colbourn, C. J., and Read, R. C. 1979. Orderly algorithms for generating restricted classes of graphs. *Journal of Graph Theory* 3(2):187–195.

Crawford, J. M.; Ginsberg, M. L.; Luks, E. M.; and Roy, A. 1996. Symmetry-breaking predicates for search problems. In Aiello, L. C.; Doyle, J.; and 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.*, 148–159. Morgan Kaufmann.

Exoo, G.; Ling, A. C.; McSorley, J. P.; Phillips, N.; and Wallis, W. 2002. Totally magic graphs. *Discrete Mathematics* 254(1):103 – 113.

Flener, P.; Frisch, A. M.; Hnich, B.; Kiziltan, Z.; Miguel, I.; and Walsh, T. 2002. Matrix modelling: Exploiting common patterns in constraint programming. In *Proceedings of the International Workshop on Reformulating Constraint Satisfaction Problems*, 27–41.

Frisch, A. M., and Harvey, W. 2003. Constraints for breaking all row and column symmetries in a three-by-two matrix.

Frisch, A. M.; Hnich, B.; Kiziltan, Z.; Miguel, I.; and Walsh, T. 2006. Propagation algorithms for lexicographic ordering constraints. *Artif. Intell.* 170(10):803–834.

Gallian, J. A. 2018. A dynamic survey of graph labeling. *The Electronic Journal of Combinatorics* 6. Version 21.

Gebser, M.; Kaufmann, B.; and Schaub, T. 2012. Conflict-driven answer set solving: From theory to practice. *Artif. Intell.* 187:52–89.

Gent, I. P.; Jefferson, C.; and Miguel, I. 2006. Minion: A fast, scalable, constraint solver. In *Proceedings of the 2006 Conference on ECAI 2006: 17th European Conference on Artificial Intelligence August 29 – September 1, 2006, Riva Del Garda, Italy*, 98–102. Amsterdam, The Netherlands, The Netherlands: IOS Press.

Halldórsson, M. M.; Kitaev, S.; and Pyatkin, A. 2016. Semi-transitive orientations and word-representable graphs. *Discrete Appl. Math.* 201(C):164–171.

Heule, M. J. H. 2016. The quest for perfect and compact symmetry breaking for graph problems. In Davenport, J. H.; Negru, V.; Ida, T.; Jebelean, T.; Petcu, D.; Watt, S. M.; and Zaharie, D., eds., *18th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2016, Timisoara, Romania, September 24-27, 2016*, 149–156. IEEE Computer Society.

Heule, M. J. H. 2019. Optimal symmetry breaking for graph problems. *Mathematics in Computer Scienc*.

Itzhakov, A., and Codish, M. 2016. Breaking symmetries in graph search with canonizing sets. *Constraints* 21(3):357–374.

Jäger, G., and Arnold, F. 2015. SAT and IP based algorithms for magic labeling including a complete search for total magic labelings. *J. Discrete Algorithms* 31:87–103.

Kitaev, S. V., and Pyatkin, A. V. 2018. Word-representable graphs: a survey. *Journal of Applied and Industrial Mathematics* 12(2):278–296.

Kvasnička, V., and Pospíchal, J. 1990. Canonical indexing and constructive enumeration of molecular graphs. *J. Chem. Inf. Comput. Sci.* 30(2):99–105.

Luks, E. M., and Roy, A. 2004. The complexity of symmetry-breaking formulas. *Ann. Math. Artif. Intell.* 41(1):19–45.

McKay, B. D. 1998. Isomorph-free exhaustive generation. *Journal of Algorithms* 26(2):306–324.

Metodi, A.; Codish, M.; and Stuckey, P. J. 2013. Boolean equi-propagation for concise and efficient SAT encodings of combinatorial problems. *J. Artif. Intell. Res. (JAIR)* 46:303–341.

Radziszowski, S. P. 1994. Small Ramsey numbers. *Electronic Journal of Combinatorics*. Revision #14: January, 2014.

Read, R. C. 1978. Every one a winner or how to avoid isomorphism search when cataloguing combinatorial configurations. *Ann. Discrete Math.* 2:107–120.

Shlyakhter, I. 2001. Generating effective symmetry-breaking predicates for search problems. *Electronic Notes in Discrete Mathematics* 9:19–35.

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

Sloane, N. J. A., ed. The On-Line Encyclopedia of Integer Sequences. https://oeis.org. Accessed August 2019.

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*, volume 4204 of *LNCS*, 650–664. Springer.