SAT Modulo Symmetries for Graph Generation

Markus Kirchweger
Algorithms and Complexity Group, TU Wien, Austria

Stefan Szeider
Algorithms and Complexity Group, TU Wien, Austria

Abstract

We propose a novel constraint-based approach to graph generation. Our approach utilizes the interaction between a CDCL SAT solver and a special symmetry propagator where the SAT solver runs on an encoding of the desired graph property. The symmetry propagator checks partially generated graphs for minimality w.r.t. a lexicographic ordering during the solving process. This approach has several advantages over a static symmetry breaking: (i) symmetries are detected early in the generation process, (ii) symmetry breaking is seamlessly integrated into the CDCL procedure, and (iii) the propagator can perform a complete symmetry breaking without causing a prohibitively large initial encoding. We instantiate our approach by generating extremal graphs with certain restrictions in terms of girth and diameter. With our approach, we could confirm the Simon-Murty Conjecture (1979) on diameter-2-critical graphs for graphs up to 18 vertices.

2012 ACM Subject Classification Theory of computation → Constraint and logic programming; Mathematics of computing → Extremal graph theory

Keywords and phrases symmetry breaking, SAT encodings, graph generation, combinatorial search, extremal graphs, CDCL

Digital Object Identifier 10.4230/LIPIcs.CP.2021.34

Supplementary Material *Software (Source Code)*: `https://doi.org/10.5281/zenodo.5170575`

Funding The authors acknowledge the support from the Austrian Science Fund (FWF), project P32441, and from the Vienna Science and Technology Fund (WWTF), project ICT19-065.

1 Introduction

Many challenging problems in Combinatorics can be stated as the question of whether a graph with a particular property exists. A common approach to such problems is to use a tool like Nauty [29] to generate all connected graphs up to isomorphism with a given number n of vertices and to check each of them for the desired property. However, up to isomorphism, already for $n = 11$ there are over a billion connected graphs, so this method quickly approaches its limit.

As demonstrated by Codish et al. [10], constraint-based graph generation offers a compelling alternative approach. The graph property is expressed in terms of constraints, and further constraints expressing a *static symmetry break* are added. The latter constraints are based on a lexicographic ordering of solution graphs and exclude some graphs that are not minimal for this ordering. This approach has the advantage that the graph property is taken into account already during the generation process. However, a complete symmetry breaking requires a prohibitively large encoding size. Therefore, one needs to confine only to a partial check, e.g., that swapping two vertices doesn't yield a lexicographically smaller graph.

We propose the novel approach *SAT modulo Symmetries* (SMS) to constraint-based graph generation. SMS utilizes the interaction between a CDCL[1] SAT solver and a special propagator excluding lexicographically non-minimal graphs during the search. Thus, in contrast to static symmetry breaking, the minimality check is not added to the encoding but is carried out dynamically by the symmetry propagator.

[1] Conflict-Driven Clause Learning is the predominantly leading algorithmic paradigm for state-of-the-art SAT solvers [28].

The symmetry propagator is called from within a CDCL-based SAT solver already when only a few of the graph's edges are determined. For this purpose, we introduce *partially defined graphs*, the corresponding lexicographic ordering, and the algorithm MinCheck that checks their minimality. If the symmetry propagator detects the current partially defined graph is not minimal, clauses are learned and added to the solver's collection of clauses. On partially defined graphs, the minimality check is not guaranteed to be complete, but eventually, when all edges are determined, it performs a complete minimality check. Consequently, we can guarantee that all generated graphs are minimal and unique up to isomorphism. MinCheck also detects, in some cases, whether the current partially defined graph implies under minimality the existence or non-existence of further edges. This is done without increasing the encoding size. When the minimality check reveals that the current partially defined graph isn't minimal, this conflict is analyzed, and a suitable clause is added to the solver; also when an implied edge is detected, a unit clause is added.

We implemented a prototype version of SMS and tested it on two prominent problems from Extremal Graph Theory [6]. The first asks for graphs with a prescribed minimum *girth* (length of a shortest cycle) and the largest number of edges. The second asks for graphs whose *diameter* (largest distance between any two vertices) is 2 but decreases when any edge is deleted. Both are fundamental problems that have been studied for many decades [6, 14, 27]. We could verify some extremal numbers for the girth problem, and confirm the Simon-Murty Conjecture [7] on diameter-2-critical graphs with up to 18 vertices, improving upon the known bound of 11.

Our experimental results show that SMS exhibits an encouraging performance, particularly on unsatisfiable instances. We developed the testing algorithm MinCheck from scratch, since existing methods based on various other forms of graph canonization cannot handle partially defined graphs [29]. The check often takes a significant amount of the solving time; here, we see ample room for improvement. However, the time which SMS spends on the SAT solving itself is significantly reduced in comparison to a static symmetry breaking.

Related Work

Dynamic symmetry breaking in the broader sense, where symmetry breaking constraints or clauses are added during the search, has a long history, see, e.g., [3, 13, 15, 21, 32]. More recently, this has also been combined with nogood (or clause) learning [9, 12, 34], using the fact that if a new clause/nogood is learned, and the symmetries of the problem are known, then one can propagate and learn further clauses/nogoods. Metin et al. [30] explored another way of dynamically utilizing symmetries in the context of SAT by considering the lexicographic order on the assignments themselves; the symmetries are computed by external tools or provided by the user before the SAT-solving and are then taken into account by the solver. Equipping a SAT solver with a special-purpose propagator has been explored before, e.g., by Liffiton and Maglalang [25] for cardinality constraints and by Gebser et al. [18] for digraph acyclicity. The SAT Modulo Theory (SMT) framework uses a similar approach, where the SAT solver interacts with a theory solver, which provides propagators for a first-order logic theory [5].

2 Preliminaries

Graphs. All considered graphs are undirected and simple (i.e., without parallel edges or self-loops). A *graph* G consists of set $V(G)$ of vertices and a set $E(G)$ of edges; we denote the edge between vertices $u, v \in V(G)$ by uv or equivalently vu. We write $G - e$ for the graph

obtained from G by deleting the edge e and $G - v$ for the graph obtained from G by deleting the vertex v. $\mathcal{G}_n$ denotes the class of all graphs G with $V(G) = \{1, \ldots, n\}$. A_G denotes the *adjacency matrix* of a graph $G \in \mathcal{G}_n$ where the element at row v and column u, denoted by $A_G[v][u]$, is 1 if $vu \in E$ and 0 otherwise. $A_G[v]$ denotes the v-th row of A_G. $\mathcal{S}_n$ denotes the set of all permutations over $\{1, \ldots, n\}$.

Graphs $G_1, G_2 \in \mathcal{G}_n$ are *isomorphic* if there is a permutation $\pi \in \mathcal{S}_n$ such that for all $1 \leq u < v \leq n$ we have $uv \in E(G_1)$ if and only if $\pi(u)\pi(v) \in E(G_2)$. $\pi(G)$ denotes the graph obtained from $G \in \mathcal{G}_n$ by the permutation $\pi \in \mathcal{S}_n$, where $E(\pi(G)) = \{\, \pi(u)\pi(v) : uv \in E(G) \,\}$. The total order $\preceq$ is defined for $G, H \in \mathcal{G}_n$ by setting $G \preceq H$ if and only if $A_G[1]A_G[2] \ldots A_G[n]$ is lexicographically smaller or equal to $A_H[1]A_H[2] \ldots A_H[n]$. G is *lexicographically smaller* than H (in symbols $G \prec H$) if $G \preceq H$ and $G \neq H$. $G \in \mathcal{G}_n$ is *lexicographically minimal* or $\preceq$*-minimal* if $G \preceq \pi(G)$ for every $\pi \in \mathcal{S}_n$.

We will also consider the lexicographic ordering of pairs of vertices from $\{1, \ldots, n\}$ where $(v_1, v_2) < (u_1, u_2)$ if and only if either (i) $v_1 < u_1$ or (ii) $v_1 = u_1$ and $v_2 < u_2$. We observe that $A[v_1][v_2]$ occurs before $A[u_1][u_2]$ in an adjacency matrix A if and only if $(v_1, v_2) < (u_1, u_2)$. We will say, that a vertex pair is *more important* than another if the vertex pair is smaller by this order. Similarly, we will say that an entry $A[v_1][v_2]$ of the adjacency matrix A is more important than $A[u_1][u_2]$ if $(v_1, v_2) < (u_1, u_2)$.

▶ **Observation 1.** *Let $G, H \in \mathcal{G}_n$. Then $G \prec H$ if and only if there are $1 \leq i, j \leq n$ such that $A_G[i][j] = 0$, $A_H[i][j] = 1$, and for all more important pairs of vertices (i', j') the values of the adjacency matrices are equal, i.e., $A_G[i'][j'] = A_H[i'][j']$.*

Formulas and Satisfiability. A *literal* is a propositional variable or negated propositional variable. A *clause* is a disjunction of literals. A formula in Conjunctive Normal Form (CNF) is a conjunction of clauses. A *(partial) assignment* is a function $f : X \to \{\text{true}, \text{false}\}$ defined on a set X of propositional variables. For a variable $x \notin X$ we say that f is *undefined*. Assignments extend to literals in an obvious way. A *model* of a CNF formula F is an assignment f defined on the variables of F such that each clause of F contains a literal that is set to true by f. A clause containing a single literal is *unity clause*.

3 Dynamic Symmetry Breaking in SMS

This section presents our method for dramatically reducing the search space for finding all graphs in $\mathcal{G}_n$ modulo isomorphism, that satisfy a given property. As we deal with graphs from $\mathcal{G}_n$ for some fixed n, we will use CNF formulas that contain all the propositional variables $e_{v,u}$, for $v < u$, which are true if the edge vu is present in the implicitly represented graph. Hence, we can extract a graph from a model of the formula.

Our aim is to decide during the CDCL SAT solver's run whether the current partial assignment can be extended to a model, such that the represented graph is $\preceq$-minimal. For this purpose we consider *partially defined graphs*, since during solving we don't know the final graph yet. A partially defined graph is a graph G where $E(G)$ is split into two disjoint sets $D(G)$ and $U(G)$. $D(G)$ contains the *defined* edges, $U(G)$ contains the *undefined* edges. A (fully defined) graph is a partially defined graph G with $U(G) = \emptyset$. Similarly to $\mathcal{G}_n$, let $\mathcal{P}_n$ denote the class of all partially defined graphs G with $V(G) = \{1, \ldots, n\}$. Analogously to the adjacency matrix of a fully defined graph, we define the adjacency matrix A_G of a partially defined graph $G \in \mathcal{P}_n$ as follows: $A_G[v_1][v_2] = 1$ if $v_1v_2 \in D(G)$, $A_G[v_1][v_2] = \star$ if $v_1v_2 \in U(G)$, and $A_G[v_1][v_2] = 0$ otherwise. From a partial assignment $f : X \to \{\text{true}, \text{false}\}$ we can extract the partially defined graph G with $V(G) = \{1, \ldots, n\}$, $D(G) = \{\, ij : e_{i,j} \in X,\ f(e_{i,j}) = \text{true} \,\}$ and $U(G) = \{\, ij : e_{i,j} \notin X \,\}$.

A partially defined graph $G \in \mathcal{P}_n$ can be *extended* to a graph $H \in \mathcal{G}_n$ if $D(G) \subseteq E(H) \subseteq D(G) \cup U(G)$. We write $\mathcal{X}(G)$ for the set of all graphs to which G can be extended. A partially defined graph $G \in \mathcal{P}_n$ is $\preceq$-*minimal* if $\mathcal{X}(G)$ contains a $\preceq$-*minimal* graph.

A permutation $\pi \in \mathcal{S}_n$ is a *witness* of the non-$\preceq$-minimality of $G \in \mathcal{P}_n$ if $\pi(H) \prec H$ for all $H \in \mathcal{X}(G)$; observe that in that case G cannot be $\preceq$-minimal.

Let $G \in \mathcal{P}_n$, $\pi \in \mathcal{S}_n$, and (i,j) a vertex pair. We say (i,j) is (G,π)-*equal* if $A_H[i][j] = A_{\pi(H)}[i][j]$ for all $H \in \mathcal{X}(G)$ (i.e., $(i,j) \in \{(\pi(i),\pi(j)),(\pi(j),\pi(i))\}$, or $A_G[i][j] = A_{\pi(G)}[i][j] \neq \star$), and (i,j) is (G,π)-*critical* if $(A_G[i][j], A_{\pi(G)}[i][j]) \in \{(1,0),(\star,0),(1,\star)\}$.

Next we will introduce indicator pairs, which will be of crucial importance for checking whether a partially defined graph is $\preceq$-minimal. For $G \in \mathcal{P}_n$ and $\pi \in \mathcal{S}_n$ the vertex pair (i,j) is a (G,π)-*indicator pair* if (i,j) is (G,π)-critical and for every $(i',j') < (i,j)$ with $i' < j'$ at least one of the three cases holds: (i) $(i',j') \in \{(\pi(i'),\pi(j')),(\pi(j'),\pi(i'))\}$, or (ii) $A_G[i'][j'] = 1$, or (iii) $A_{\pi(G)}[i'][j'] = 0$. A (G,π)-indicator pair (i,j) is a *strict* (G,π)-*indicator pair* if $A_G[i][j] = 1$ and $A_{\pi(G)}[i][j] = 0$. A partially defined graph $G \in \mathcal{P}_n$ is *constraining* if there is a (G,π)-indicator pair for some $\pi \in \mathcal{S}_n$.

▶ **Proposition 2.** *Let $G \in \mathcal{P}_n$. If there is a strict (G,π)-indicator pair for some $\pi \in \mathcal{S}_n$ then G is not $\preceq$-minimal. Furthermore, if G is fully defined and not $\preceq$-minimal, then there is a strict (G,π)-indicator pair for some $\pi \in \mathcal{S}_n$.*

Proof. Let $G \in \mathcal{P}_n$ and (i,j) be a strict (G,π)-indicator pair. For the sake of a contradiction, assume that G is $\preceq$-minimal. By definition, there is a fully defined graph $H \in \mathcal{X}(G)$ which is $\preceq$-minimal. First, we show by induction over all more important vertex pairs (i',j') than (i,j) (including $i' > j'$) that $A_H[i'][j'] = A_{\pi(H)}[i'][j']$. We distinguish five cases.

1. If $i' > j'$ then $(j',i') < (i',j')$ hence $A_H[i'][j'] = A_H[j'][i'] = A_{\pi(H)}[j'][i'] = A_{\pi(H)}[i'][j']$ holds by induction hypothesis.
2. If $(i',j') = (\pi(i'),\pi(j'))$ then $A_{\pi(H)}[i'][j'] = A_H[\pi^{-1}(i')][\pi^{-1}(j')] = A_H[i'][j']$ holds.
3. If $(i',j') = (\pi(j'),\pi(i'))$ then $A_{\pi(H)}[i'][j'] = A_H[\pi^{-1}(i')][\pi^{-1}(j')] = A_H[j'][i'] = A_H[i'][j']$ holds.
4. If $A_G[i'][j'] = 1$ then $A_H[i'][j'] = 1$. By induction hypothesis, $A_{\pi(H)}[i''][j''] = A_H[i''][j'']$ holds for all more import vertex pairs (i'',j'') than (i',j'). Hence also $A_{\pi(H)}[i'][j'] = 1$ must hold, otherwise Observation 1 would be violated, so $A_H[i'][j'] = A_{\pi(H)}[i'][j']$.
5. If $A_{\pi(G)}[i'][j'] = 0$ then $A_{\pi(H)}[i'][j'] = 0$. Again, by induction hypothesis, $A_{\pi(H)}[i''][j''] = A_H[i''][j'']$ holds for all more import vertex pairs (i'',j'') than (i',j'). Hence also $A_H[i'][j'] = 0$ must hold, otherwise Observation 1 would be violated, so $A_H[i'][j'] = A_{\pi(H)}[i'][j']$.

So, we know that $A_{\pi(H)}[i'][j'] = A_H[i'][j']$ for all more important vertex pairs (i',j') than (i,j) and therefore, by Observation 1, H cannot be $\preceq$-minimal, consequently G cannot be $\preceq$-minimal in contradiction to our assumption.

For the second part of the proposition, let G be an arbitrary, non-$\preceq$-minimal, fully defined graph. Then, by definition of $\preceq$-minimality, there is a $\pi \in \mathcal{S}_n$ such that $\pi(G) < G$. Due to Observation 1, we know that there is a vertex pair (i,j) such that $A_G[i][j] = 1$, $A_{\pi(G)}[i][j] = 0$, and $A_{\pi(G)}[i'][j'] = A_G[i'][j']$ for all more important pairs (i',j') than (i,j), hence either $A_G[i][j] = 1$ or $A_{\pi(G)}[i][j] = 0$, so (i,j) is a strict (G,π)-indicator pair. ◀

▶ **Observation 3.** *Let $G \in \mathcal{P}_n$, $H \in \mathcal{X}(G)$ a $\preceq$-minimal graph, and (i,j) a (G,π)-indicator pair for some $\pi \in \mathcal{S}_n$. Then $A_H[i][j] = A_{\pi(H)}[i][j]$.*

Observation 3 states that if (i,j) is a (G,π)-indicator pair and $A_G[i][j] = 1$ then we can imply that $A_{\pi(H)}[i][j] = 1$ for every $\preceq$-minimal graph $H \in \mathcal{X}(G)$, and if $A_{\pi(G)}[i][j] = 0$ then we can imply that $A_H[i][j] = 0$.

A key component for the symmetry propagator that we use within SMS is an algorithm that tests whether the partially defined graph $G \in \mathcal{P}_n$ as represented by the current partial assignment is $\preceq$-minimal, i.e., whether it can be extended to a $\preceq$-minimal fully defined graph. Performing this test is computationally hard in the worst case. For example, finding the lexicographically smallest isomorphic graph is known to be NP-hard [2].

We restrict our search for permutations π with a (G, π)-indicator pair. In the worst case, this test still requires considering all the $n!$ permutations of the vertices. However, we use a more sophisticated approach that exploits the partially defined graph's structure which often allows us to avoid the consideration of most of the permutations. Whenever during search, SMS has a new partial assignment which updates the partially defined graph G, it calls MinCheck(G) which searches for an indicator pair. If a (G, π)-indicator pair is found, it extracts a clause representing the reason why G isn't $\preceq$-minimal or why a certain edge must be present or cannot be present in any $\preceq$-minimal extension of G.

We integrate this procedure into the conflict-driven clause learning algorithm (CDCL). Whenever the CDCL algorithm assigns an edge variable a truth value, we check the $\preceq$-minimality of the current partially defined graph G and add a clause if necessary. For efficiency, some minimality checks may be skipped. If the added clause is invalidated by the current partial assignment, then the current partial assignment must be discarded and the solver backtracks. Otherwise, the clause is a unit clause, so a literal can be propagated by Boolean constraint propagation. All the created clauses are added as learned clauses to the solver, so the solver's clause-deletion policy can discard them if they aren't needed anymore.

Below we present the MinCheck algorithm in detail.

3.1 Minimality Check

Our minimality test is based on the concept of a *generalized ordered partition* (GOP) of $V = \{1, \dots, n\}$, which is a list of triples $P = [(V_1, l_1, u_1), \dots, (V_k, l_k, u_k)]$ such that $V_1 \cup \cdots \cup V_k = V$, V_i, V_j are disjoint for $1 \leq i < j \leq n$, $u_i, l_i \in \{1, \dots, n\}$, $u_i + 1 = l_{i+1}$ for $1 \leq i \leq k-1$, and $|V_i| = u_i - l_i + 1$. Let f be the mapping that indicates to which set each vertex belongs to, i.e., $f(v) = i$ if and only if $v \in V_i$. Then we associate with a GOP P the set of permutations $\mathrm{Perm}(P) = \{\, \pi \in \mathcal{S}_n : l_{f(v)} \leq \pi(v) \leq u_{f(v)} \text{ for all } v \in V \,\}$; i.e., the GOP gives a range for each vertex to which it can potentially be mapped.

Next we describe the algorithm MinCheck. The input for the initial call of MinCheck is a partially defined graph $G \in \mathcal{P}_n$. The idea is to start with the GOP $P = [(V, 1, n)]$ and refine it until we have found a (G, π)-indicator pair with some π represented by the GOP or can conclude that no such indicator pair exists. Therefore, we recursively assign a vertex v to r, i.e., $\pi(v) = r$ for all $\pi \in \mathrm{Perm}(P)$, starting with $r = 1$ and adapt the GOP correspondingly by splitting up the triples (V_i, l_i, u_i) into multiple triples if necessary.

We use a recursive procedure MinCheck with the following input: a partially defined graph $G \in \mathcal{P}_n$, a GOP $P = [(V_1, l_1, u_1), \dots, (V_k, l_k, u_k)]$, and a row $r \in \{1, \dots, n\}$. For all $j < r$ we have $|V_j| = 1$, in other words, all the vertices which are mapped to the first $r-1$ vertices are already fixed. Furthermore, we will see that all pairs up to $(r-1, n)$ are (G, π)-equal for all $\pi \in \mathrm{Perm}(P)$. If $r = n$ we return nil.

Now we choose a $v \in V_r$: we adapt the GOP P to a GOP P_v such that $\pi(r) = v$ for all permutations represented by P_v and split up all triples (V_i, l_i, u_i) with $i \geq r$ (in the order they occur in the list) such that also the current row, (i.e., all pairs (r, j) with $j > r$) are (G, π)-equal for all permutations π represented by the GOP P_v or until we have found an indicator pair. In the former case, we call MinCheck$(G, P_v, r+1)$ with the adapted GOP P_v and return an indicator pair if found; otherwise we backtrack. If every choice of $v \in V_r$ returned no indicator pair, then we return nil.

Before we stipulate how to split the triples in more detail, we mention some preconditions for MinCheck which will allow us to argue that the preconditions are also satisfied at recursive calls. The preconditions are as follows:

P1 The vertices which are mapped to the first $r-1$ vertices are already determined, i.e., $|V_i| = 1$ for every $i < r$.

P2 For every permutation $\pi \in \mathrm{Perm}(P)$ every pair (i', j') with $i' < j', i' < r$ is equal under π.

P3 For every $\pi \in \mathcal{S}_n \setminus \mathrm{Perm}(P)$ with $\pi(k) \in V_k$ for $k < r$, there is some pair (i', j') with $i' < j', i' < r$ which is not (G, π)-equal and is more important than any (G, π)-critical vertex pair, hence there is no (G, π)-indicator pair.

First, we split (V_r, l_r, u_r) into $(\{v\}, l_r, l_r)$ and $(V_r \setminus \{v\}, l_r + 1, u_r)$, so $\pi(v) = r$ for all $\pi \in \mathrm{Perm}(P_v)$. For each $(V_i, l_i, u_i), i \in \{r, \dots, n\}$, we apply the following steps, where $V_i^0 := \{\, u \in V_i : A_G[v][u] = 0 \,\}$, $V_i^\star := \{\, u \in V_i : A_G[v][u] = \star \,\}$, and $V_i^1 := \{\, u \in V_i : A_G[v][u] = 1 \,\}$. (Some special care is needed for $i = r$ since it was already split, so we use $(V_r \setminus \{v\}, l_r + 1, u_r)$ instead of (V_r, l_r, u_r)):

1. We split (V_i, l_i, u_i) into the triples $(V_i^0, l_i, l_i + |V_i^0| - 1)$ and $(V_i^\star \cup V_i^1, l_i + |V_i^0|, u_i)$; this ensures that the vertices not adjacent to v are mapped to the smallest vertices possible without violating the previous GOP.
2. If the set $J = \{\, u : A_G[r][u] \neq 0, l_i \leq u < l_i + |V_i^0| \,\} \neq \emptyset$, then we put $j = \min J$ and (r, j) is a (G, π)-indicator pair for every $\pi \in \mathrm{Perm}(P_v)$. Otherwise $A_G[r][u] = A_{\pi(G)}[r][u] = 0$ for every $\pi \in \mathrm{Perm}(P_v)$, hence Preconditions 2 and 3 hold up to all more important pairs than $(r, l_i + |V_i^0|)$.
3. Now we iterate over the remaining indexes $p \in [l_i + |V_i^0|, \dots, u_i]$. We distinguish between three cases:
   - **a.** $A_G[r][p] = \star$: Because all vertices from V_i^0 are mapped to smaller vertices than p, the only way to guarantee that the pair is equal (or critical) is to ensure that $(r, p) \in \{(\pi(r), \pi(p)), (\pi(r), \pi(p))\}$. In all other cases condition 2 does not hold up to all pairs including (r, p) . This can only be the case if
     - $v = r$ and $p \in V_i^\star$, then we must split $(V_i^\star \cup V_i^1, p, u_i)$ into the triples $(\{p\}, p, p)$ and $(V_i^\star \setminus \{p\} \cup V_i^1, p + 1, u_i)$ and remove p from $V_i^\star$, or
     - $v = p$ and $r \in V_i^\star$, then we must split $(V_i^\star \cup V_i^1, p, u_i)$ into the triples $(\{r\}, p, p)$ and $(V_i^\star \setminus \{r\} \cup V_i^1, p + 1, u_i)$ and remove r from $V_i^\star$

     In all other cases, we backtrack.
   - **b.** If $A_G[r][p] = 0$ we backtrack, since $A_{\pi(G)}[r][p] \neq 0$ for every $\pi \in \mathrm{Perm}(P_v)$.
   - **c.** If $A_G[r][p] = 1$, we again distinguish three cases:
     - If $V_i^\star \neq \emptyset$ then we split $(V_i^\star \cup V_i^1, p, u_i)$ into the triples $(V_i^\star, p, p + |V_i^\star| - 1)$ and $(V_i^1, p + |V_i^\star|, u_i)$ so (r, p) is a (G, π)-indicator pair for all $\pi \in \mathrm{Perm}(P_v)$.
     - If $V_i^\star = \emptyset$ and $A_G[r][p'] \neq 1$ for some $p' \in \{p, \dots, u_i\}$, then we backtrack.
     - If $V_i^\star = \emptyset$ and $A_G[r][p'] = 1$ for all $p' \in \{p, \dots, u_i\}$ then $A_G[r][p'] = A_{\pi(G)}[r][p']$ for all $\pi \in \mathrm{Perm}(P_v)$. So, all more important pairs up to (r, u_i) are (G, π)-equal for all $\pi \in \mathrm{Perm}(P_v)$.

One of the steps 1–3 will either produce an indicator pair or cause a backtrack to another $v \in V_r$. If all choices $v \in V_r$ have been exhausted unsuccessfully, we return nil.

▶ **Lemma 4.** *Let $G \in \mathcal{P}_n$ and $r \in \{1, \dots, n\}$. If the Preconditions P1–P3 are satisfied, MinCheck$(G, [(V_1, l_1, u_1), \dots, (V_k, l_k, u_k)], r)$ will return a permutation π with $\pi(u) \in V_u$ for $u < r$ and a (G, π)-indicator pair if such a permutation and indicator pair exists; otherwise the procedure returns nil.*

Proof. In MinCheck, we only backtrack if we can ensure, that (i) there is no indicator pair with $\pi(u) \in V_u$ for $u < r$ and $\pi(v) = r$ and (ii) whenever we find an indicator pair, we return it immediately. Precondition 2 ensures in all cases that it is indeed an indicator pair and Precondition 3 that we do not loose any potential candidates. Since the preconditions are satisfied for recursive calls, the lemma follows by induction. ◀

We can now put the above auxiliary results together and establish the MinCheck algorithm's correctness and the fact that it performs a complete minimality check on fully defined graphs.

▶ **Theorem 5.** *Let $G \in \mathcal{P}_n$. If G is constraining, then MinCheck(G) returns a permutation π and a (G, π)-indicator pair; otherwise MinCheck(G) returns nil.*

Proof. MinCheck$(G, [(V, 1, n)], 1)$ is called at the beginning, hence the proposition is true due to Lemma 4, since $\mathcal{S}_n = \text{Perm}([(V, 1, n)])$ and all preconditions are satisfied. ◀

Finally, we present how to extract a suitable clause from a partially defined graph $G \in \mathcal{P}_n$ and a (G, π)-indicator pair (i, j).

Let $S = \{\,(i', j') : (i', j') \in \{(\pi(i'), \pi(j')), (\pi(j'), \pi(i'))\}\,\}$, $S_1 = \{\,(i', j') : (i', j') < (i, j), i' < j', A_G[i][j] = 1, (i', j') \notin S\,\}$ and $S_2 = \{\,(i', j') : (i', j') < (i, j), i' < j', A_{\pi(G)}[i][j] = 0, (i', j') \notin S\,\}$. Then

$$\bigvee_{(i',j')\in S_1} \neg e_{i',j'} \bigvee_{(i',j')\in S_2} e_{\pi^{-1}(i'),\pi^{-1}(j')} \vee \neg e_{i,j} \vee e_{\pi^{-1}(i),\pi^{-1}(j)}$$

is the resulting clause, which we add as learned clause. By Proposition 2, this clause must be satisfied by every assignment which represents a $\preceq$-minimal graph. We would like to stress that the created clause is satisfied by every partial assignment representing a $\preceq$-minimal graph; hence, the correctness of the symmetry breaking is independent of the found permutations and indicator pairs.

4 Static Symmetry Breaking

For comparison, we describe the static symmetry breaking approach, which is essentially the "improved lexicographic break" proposed by Codish et al. [10]. The SAT solver runs on a CNF formula $F \wedge M$, where F encodes the properties of the graph sought for, and M encodes a minimality property. A complete symmetry breaking would require a prohibitively large encoding size. Therefore, M just ensures that swapping any two vertices does not lead to a lexicographically smaller graph (if it would, the current graph can't be $\preceq$-minimal).

Swapping two vertices i, j leads to swapping in the adjacency matrix the elements $A[i][k]$ with $A[j][k]$ and $A[k][i]$ with $A[k][j]$, respectively, for $k \in \{1, \ldots, n\} \setminus \{i, j\}$. All the other entries of the matrix will not change, because the diagonal only contains zeros and we have $A[i][j] = A[j][i]$. So we have to check whether swapping the entries does not lead to a $\preceq$-smaller graph. Following [10], we define an order on the rows $A[i]$ and $A[j]$, by setting $A[i] <_{i,j} A[j]$ if and only if $A[i]$ is lexicographically smaller than $A[j]$ while ignoring the i-th and j-th elements in both rows.

▶ **Proposition 6** ([10]). *If $G \in \mathcal{G}_n$ is $\preceq$-minimal, then $A_G[i] \leq_{i,j} A_G[j]$ for all $1 \leq i < j \leq n$.*

We can now define the formula M for the static symmetry break as

$$\bigwedge_{1\leq i<j\leq n} \bigwedge_{k\in\{1,\ldots,n\}\setminus\{i,j\}} \Big(\bigwedge_{l\in\{1,\ldots,k\}\setminus\{i,j\}} (e_{i,l} \leftrightarrow e_{j,l}) \rightarrow (e_{i,k} \rightarrow e_{j,k}) \Big).$$

Instead of this direct SAT encoding, Codish et al. [10] used the solver BEE to encode the property stated in Proposition 6. BEE [31] compiles finite domain constraints to SAT while additionally applying transformations to simplify the encoding and optimize it.

5 Prototype Implementation and Experimental Setup

In this section, we will describe our experimental setup and some implementation details. As the SAT solver, we use Clingo [19, 20], an ASP solver containing a complete state-of-the-art CDCL SAT solver. Clingo comes with a C interface that supports rapid prototyping for developing custom propagators. We used Clingo's C-interface to integrate our implementation of MinCheck into the solver. Our implementation is available at Zenodo [24].

The parameter *frequency* allows us to balance the time spent on the minimality check and the time spent by the SAT solver itself. If *frequency* has the value $1/q$, then MinCheck is called only every q-th time an edge variable has been assigned.

Our experiments are run on a computer with Intel Xeon E5540 at 2.53 GHz, 24 GB RAM, under Ubuntu 18.04. We use Clingo 5.5.0 and all tests are executed with a single thread.

6 Extremal Graphs with Required Girth

A prominent research topic in Extremal Graph Theory [6] is the study of extremal graphs (i.e., graphs with the largest possible number of edges) on n vertices that exclude a given family $\mathcal{F}$ of graphs as subgraphs. $\mathrm{EX}(n, \mathcal{F})$ denotes the class of extremal graphs with that property, and $\mathrm{ex}(n, \mathcal{F})$ denotes the number of edges of the graphs in $\mathrm{EX}(n, \mathcal{F})$. The special case, where $\mathcal{F} = \mathcal{C}_k$, the family of cycles up to length k, has received much attention; for convenience, we write $f_k(n) = \mathrm{ex}(n, \mathcal{C}_k)$. The *girth* of a graph G is the length of a shortest cycle in G (or ∞ if G is acyclic). Hence $\mathrm{EX}(n, \mathcal{C}_k)$ contains precisely the edge-maximal graphs of girth $> k$. The base case of $k = 3$ has been settled over a century ago by Mantel's Theorem [27]: $f_3(n) = \mathrm{ex}(n, \mathcal{C}_3) = \lfloor n^2/4 \rfloor$, where $\mathrm{EX}(n, \mathcal{F})$ contains precisely the complete bipartite graph $K_{\lceil n/2 \rceil, \lfloor n/2 \rfloor}$. For the general case $k > 3$, however, no closed formula is known, and researchers have tried to compute $f_k(n)$ for small values of k [1, 10, 17, 35, 36], or at least provide lower and upper bounds.

Next, we describe a SAT encoding that produces for given integers n, m, k a propositional CNF formula $F(n, m, k)$. The formula is satisfiable if and only if there is a graph $G \in \mathrm{EX}(n, \mathcal{C}_k)$ with m edges, and where we can construct G from the satisfying assignment. We will then evaluate the formula with our SMS-solver and report the experimental results.

6.1 Encoding

We state a useful result before we present the encoding for $F(n, m, k)$ where δ_G and Δ_G denote the minimum and maximum degree of a graph G, respectively.

▶ **Lemma 7** ([17]). *If G is a graph of girth ≥ 5 with n vertices and m edges, then $n \geq 1 + \Delta_G \cdot \delta_G \geq 1 + \delta_G^2$, $\delta_G \geq m - f_4(n-1)$, and $\Delta_G \cdot n \geq 2m$.*

In particular, this applies to all graphs in $\mathrm{EX}(n, \mathcal{C}_k)$ for $k \geq 4$. We also use the obvious inequality $\delta_G n \leq 2m$, which follows from the Handshaking Lemma, to discard some cases.

According to Lemma 7, we can compute for each pair n, m the set $I_{n,m}$ of possible intervals $[a, b]$ such that for each graph G with n vertices and m edges, we have $a \leq \delta_G \leq \Delta_G \leq b$ for some $[a, b] \in I$. We can add to $F(n, m, k)$ suitable cardinality constraints that ensure that vertex degrees belong to one of the intervals.

To guarantee that the resulting graph has girth $> k$, we use two methods: a basic one and an improved one. The *basic method* explicitly forbids that any subset of up to k vertices forms a cycle. The set of all possible cycles of length k can be described with some basic symmetry breaking as follows:

$$C_k = \{\, (v_1, \ldots, v_k) \in \{1, \ldots, n\}^k : i \neq j \to v_i \neq v_j, v_1 = \min\{v_1, \ldots, v_k\}, v_2 < v_k \,\}.$$

Taking v_1 as the minimum fixes a particular rotation of the cycle, requiring $v_2 < v_k$ fixes an orientation of the cycle. Now we add for each element of C_k the constraint that one edge of the corresponding cycle must not be present:

$$\bigwedge_{(v_1,\ldots,v_k)\in C_k} (\neg e_{v_1,v_2} \vee \neg e_{v_2,v_3} \vee \cdots \vee \neg e_{v_{k-1},v_k} \vee \neg e_{v_k,v_1}).$$

For $k \leq 4$ this is a workable solution, but the *improved method* scales better for larger k. It is based on the following observation where $dist_G(u, v)$ denotes the length of a shortest path between vertices u and v in graph G.

▶ **Observation 8.** *A shortest cycle in a graph G containing the edge $uv \in E(G)$ has length $dist_{G-uv}(u, v) + 1$.*

Hence, we can enforce that for every edge uv, $dist_{G-uv}(u, v) + 1 \geq g$ for a required girth g. Therefore, we start at vertex i and mark all vertices adjacent to i in $G - ij$. In the next step, we additionally mark all vertices which are adjacent to already marked vertices. This will be repeated $g - 2$ times. If at the end the vertex j is marked, the girth is smaller than desired.

For the concrete encoding, we introduce propositional variables $reached_{i,j,k,s}$ for representing that vertex k can be reached in s steps from vertex i in the graph $G - ij$. Consequently

$$\begin{aligned} reached_{i,j,k,1} &= e_{i,k} \text{ for } k \in \{1, \ldots, n\} \setminus \{i, j\} \text{ and} \\ reached_{i,j,k,s} &= \bigvee_{l\in V(G)\setminus\{k\}} (e_{l,k} \wedge reached_{i,j,l,s-1}) \text{ for } s \in \{2, \ldots, g-2\},\ k \in \{1, \ldots, n\} \setminus \{i\}. \end{aligned}$$

If at any point vertex j is reached, the girth restriction is invalidated. Hence we can use the following encoding:

$$girth = \bigwedge_{i<j} \bigwedge_{s=2}^{g-2} (\neg reached_{i,j,j,s} \vee \neg e_{i,j}).$$

We further improve this encoding. If we start checking whether a vertex v is part of a cycle smaller than the given girth, i.e, we check whether $dist_{G-uv} + 1 \geq g$ for all $vu \in E$, then v cannot be on a cycle which is shorter than the girth g. So for all subsequent vertices $v' > v$, we only consider the graph $G - v$. This yields the following final encoding:

$$\begin{aligned} reached_{i,j,k,1} &= e_{i,k} \text{ for } k \in \{i+1, \ldots, n\} \setminus \{j\} \text{ and} \\ reached_{i,j,k,s} &= \bigvee_{l\in V(G)\setminus\{k\}} (e_{l,k} \wedge reached_{i,j,l,s-1}) \text{ for } s \in \{2, \ldots, g-2\},\ k \in \{i+1, \ldots, n\}. \end{aligned}$$

6.2 Results

We computed $f_k(n)$ for $k \in \{4, 5, 6\}$, and thereby verified known results [1, 10]. For fixed k and n we run SMS on the formulas $F(n, f_k(n), k)$ and $F(n, f_k(n) + 1, k)$; we performed separate runs for all the intervals in $I_{n,m}$ where m donates the number of edges. The first formula must be satisfiable for at least one interval in $I_{n,f_k(n)}$ while the second must be

unsatisfiable for every interval in $I_{n,f_k(n)+1}$. In some cases, we didn't need to compute $F(n, f_k(n)+1, k)$, since already the bounds from Lemma 7 show the non-existence of a graph with $f_k(n)+1$ edges.

In general, for fixed k and n we run SMS on $F(n, m, k)$ for different values of m, starting from a lower bound obtained by Lemma 7. As long as $F(n, m, k)$ is satisfiable, we increment m by one and repeat until we arrive at a value for which $F(n, m, k)$ is unsatisfiable or we can apply Lemma 7. Then we know that $f_k(n) = m - 1$.

Table 1 shows our results for $k = 4$ and $n \in \{15, \ldots, 28\}$. We use the basic method to encode the girth requirement and choose a frequency of 1/5. For all tables in the current section, the runtimes are given in seconds; for SMS we provide in parenthesis the fraction of the total time spent for the minimality check. The columns labeled sat give the minimal time over all intervals in $I_{n,m}$; the columns labeled unsat give the maximum. An entry n/a indicates that the unsatisfiability check is covered by Lemma 7; t.o. indicates that the timeout of 4 hours has been reached without producing a result.

Table 1 Results for $f_4(n)$.

		sat		unsat	
n	$f_4(n)$	SMS	Static	SMS	Static
15	26	0.11(67%)	1.26(0.30)	n/a	n/a
16	28	0.07(59%)	0.56(0.61)	0.91(71%)	529.42(32.20)
17	31	0.13(66%)	0.58(0.48)	n/a	n/a
18	34	0.08(53%)	2.80(0.60)	n/a	n/a
19	38	0.11(53%)	1.06(0.57)	n/a	n/a
20	41	2.41(73%)	2457.85(161.41)	n/a	n/a
21	44	0.20(61%)	1.90(151.84)	0.98(72%)	7319.96(1019.41)
22	47	1.28(72%)	3.44(16.69)	11.74(74%)	t.o.(t.o.)
23	50	2.95(79%)	1.58(367.83)	177.11(75%)	t.o.(t.o.)
24	54	30.91(74%)	638.37(80.74)	n/a	n/a
25	57	193.68(72%)	204.00(655.66)	t.o.	t.o.(t.o.)
26	61	74.63(74%)	t.o.(168.90)	n/a	n/a
27	65	1270.38(68%)	t.o.(193.44)	n/a	n/a
28	68	37.84(75%)	t.o.(t.o.)	t.o.	t.o.(t.o.)

We would like to emphasize that the purpose of the experiments is not to identify which algorithm is the fastest but rather to gain insights into the potential of a dynamic symmetry breaking for graph generation. We provide for reference the running times of our encoding of the static symmetry breaking (columns labeled Static) and the times reported by Codish et al. [10] with their "improved lexicographic break" (given in parentheses) for the same problems. This is not meant as a direct comparison, as the results by Codish et al. [10] have been run on different hardware, but just to give a rough idea on the order of magnitude the two approaches take. It is not completely clear how Codish et al. combined runtimes over all intervals $I_{n,m}$ into one single result. This has no impact on the unsat-times, because for those there is only a single interval in $I_{n,m}$.

We can see that SMS is significantly faster for the unsatisfiable instances. For example, SMS determines the unsatisfiable case for $n = 22$ quickly, although the static approach reached the timeout. SMS could also establish the unsatisfiability case for $n = 23$. We see that SMS uses a large fraction of the time for the minimality check. Therefore, a speedup for the check would have a significant impact on the runtime.

Table 2 Results for $f_5(n)$ and $f_6(n)$.

n	$f_5(n)$	sat	unsat	$f_6(n)$	sat	unsat
15	22	0.25(15%)	5.40(14%)	18	0.24(6%)	3.28(9%)
16	24	0.25(9%)	0.66(15%)	20	0.60(8%)	n/a
17	26	0.59(15%)	1.56(13%)	22	1.14(9%)	n/a
18	29	0.54(15%)	n/a	23	1.09(8%)	61.35(5%)
19	31	9.30(11%)	8.12(9%)	25	2.34(6%)	n/a
20	34	13.08(9%)	n/a	27	2.36(7%)	n/a
21	36	3.87(9%)	97.16(6%)	29	11.33(5%)	n/a
22	39	22.49(7%)	n/a	31	16.37(4%)	n/a
23	42	9.49(7%)	n/a	33	10.46(5%)	n/a
24	45	56.01(6%)	n/a	36	3.59(5%)	n/a
25	48	50.55(6%)	n/a	37	17.84(4%)	t.o.
26	52	21.08(6%)	n/a	39	12.05(5%)	174.83(3%)
27	53	40.13(5%)	t.o.	41	217.73(3%)	449.78(3%)
28	56	25.35(5%)	376.81(5%)	43	4961.74(2%)	1245.43(3%)

Table 3 Results for different frequencies.

frequency	$n = 27$ (sat)	$n = 23$ (unsat)
1/1	t.o.	240.00(94.25%)
1/2	2514.35(85.55%)	140.06(88.67%)
1/5	1293.92(67.36%)	103.72(72.46%)
1/10	208.26(51.44%)	96.99(50.79%)
1/20	782.71(30.03%)	125.25(29.45%)
1/50	636.99(12.72%)	243.30(13.63%)
1/100	4454.68(5.23%)	214.43(6.51%)
1/200	3860.19(2.80%)	608.77(3.06%)
1/500	t.o.	774.36(0.97%)
1/1000	t.o.	1103.35(0.46%)

Codish et al. [10] used some further methods to improve their results, i.e., they included embedded stars in their graphs, leading to a significant speedup. We do not use these improvements in our experiments.

Next, we report on results for computing $f_k(n)$ for $k \in \{5, 6\}$. For these cases, we used the girth-constraints based on edge-removal from Section 6.1. In these experiments, we see that far less time is spent on the minimality check than in the previous experiments, although there we had a frequency of 1/5. Most likely, the reason is the additionally created variables for the girth-constraints, because the minimality check is only called when a variable $e_{i,j}$ is assigned. The results are shown in Table 2.

Table 3 shows the influence of the parameter frequency on SMS's performance. For this analysis, we took the unsatisfiable case for $f_4(23)$ with the degree interval $[4, 5]$ and the satisfiable case for $f_4(27)$ with the degree interval $[4, 6]$.

Interestingly, the frequency shows a very clear pattern. Up to a frequency of 1/10, the runtime decreases and then increases again. The reason seems to be the high fraction of the time spent in MinCheck for a high frequency and possibly the increased number of added clauses.

7 Application: Diameter-2-Critical Graphs

The *diameter* of a graph G is the largest distance among all pairs of vertices in G, where the *distance* of two vertices is the length of a shortest path between them. A disconnected graph has diameter ∞. A graph is *diameter-d-critical* if its diameter is d, but the deletion of any edge decreases the diameter. The study of extremal properties of graphs with prescribed diameter has been initiated by Erdős and Rényi in the early 1960s [14] and has been the subject of intensive research. An important topic in the field is the characterization of diameter-d-critical graphs [7, 8, 23, 26]. An intriguing open problem is whether the Simon-Murty Conjecture [7] holds, which states that if G is a diameter-2-critical graph with n vertices and m edges, then $m \leq \lfloor n^2/4 \rfloor$, with equality precisely for the complete bipartite graph $K_{\lceil n/2 \rceil, \lfloor n/2 \rfloor}$ (i.e., similar to Mantel's Theorem mentioned above).

Using the list of non-isomorphic graphs generated with Nauty [29], Radosavljević and Živković [33] computed all diameter-2-critical graphs with up to 10 vertices. Also, Dailly et al. [11] report on a "computer search" for graphs with up to 11 vertices, focusing on graphs with a certain number of edges. With SMS we were able to extend these results to graphs with 12 vertices. The basis for this computation is a SAT encoding that produces for given integers n and m a propositional CNF formula $D_2(n, m)$ which is satisfiable if and only if there is a diameter-2-critical graph G with n vertices and m edges. As above, one can construct G from the satisfying assignment.

7.1 Encoding

Equivalently to the above definition, a graph is diameter-2-critical if and only if (i) its diameter is at most 2 and (ii) when any edge is deleted, the diameter is larger than 2. We observe that property (i) allows graphs with diameter one, i.e., complete graphs. However, after deleting any edge, the diameter would still be at most 2 for $n > 3$, which violates property (ii).

Our encoding of $D_2(n, m)$ handles both properties separately. To encode property (i), we use

$$\bigwedge_{1 \leq i < j \leq n} \big(e_{i,j} \vee \bigvee_{1 \leq k \leq n} (e_{i,k} \wedge e_{j,k})\big).$$

For property (ii), we first define a subformula $N(i, j, c)$ which encodes that vertices i and j have exactly c common neighbors, so $N(i, j, c) \leftrightarrow |\{\, k \in V : e_{i,k} \wedge e_{j,k} \,\}| = c$. We can use cardinality constraints to encode this (see, e.g., [4]) or directly use features of the ASP solver Clingo to express the cardinality constraints. With the help of this subformula, we can encode property (ii) as follows:

$$\bigwedge_{1 \leq i < j \leq n} e_{i,j} \rightarrow \Big(N(i, j, 0) \vee \bigvee_{1 \leq k \leq n} \big(e_{i,k} \wedge N(j, k, 1)\big) \vee \bigvee_{1 \leq k \leq n} \big(e_{j,k} \wedge N(i, k, 1)\big)\Big).$$

If $N(i, j, 0)$ is satisfiable, then $dist_{G-ij}(i, j) > 2$; in the other cases, either $dist_{G-ij}(j, k) > 2$ or $dist_{G-ij}(i, k) > 2$.

7.2 Results

We use the encoding $D_2(n, m)$ to enumerate all $\preceq$-minimal, diameter-2-critical graphs in $\mathcal{G}_n$. Therefore, we run SMS repeatedly; each time we find a new graph, we explicitly exclude it from the search space until no further graph is found. Additionally, we abort the minimality check after a certain number of steps, because there are some rare cases, where the check takes far too long. We us a frequency of 1/5.

Table 4 shows the results of this computation. Column #-sol gives the number of solutions found; column time gives the runtime in seconds; as above, the percentage of the runtime that has been spent for the minimality check is given in parenthesis. Column Static gives the number of solutions found with the static symmetry breaking method, without filtering isomorphic solutions. The static version could not find all solutions for $n = 12$ within 4 hours.

Table 4 Results for generating all diameter-2-critical graphs with $n \leq 10$.

	SMS		Static	
n	#-sol	time	#-sol	time
3	1	0.11(0%)	1	0.01
4	2	0.11(1%)	2	0.01
5	3	0.12(4%)	4	0.02
6	5	0.15(12%)	11	0.05
7	10	0.26(40%)	32	0.14
8	30	0.83(67%)	163	0.82
9	103	2.53(73%)	1018	6.62
10	519	10.00(63%)	9727	149.20
11	**3746**	80.47(47%)	133316	9214.83
12	**40866**	1338.09(33%)	t.o.	t.o.

When using a cutoff within SMS, we cannot guarantee that all the resulting graphs are unique up to isomorphism. Nevertheless, a check with Nauty showed that indeed all the computed graphs are unique. The number of solutions for $n \in \{11, 12\}$, stated in boldface, were unknown, as this goes beyond a generate-and-test approach.

Checking the computed graphs, we could confirm the Simon-Murty Conjecture for graphs with up to 12 vertices. By a minor adaption of the encoding, i.e., enforcing that the number of edges is $\geq \lfloor n^2/4 \rfloor$, we could extend this to $n = 13$. If we know the degree of the vertices in advance, we can create an initial GOP for the minimality check, such that only vertices with the same degree can be permuted. So, we can use SMS for every possible combination of vertex degrees. Trivially, all cases where a vertex has degree 1 can be excluded. Additionally, we can use the following theorem by Fan [16] to discard further combinations in advance ($d_G(v)$ denotes the degree of vertex v in G):

▶ **Theorem 9** ([16]). *If G is a diameter-2-critical with n vertices and m edges, then $\sum_{v \in V(G)} d_G(v)^2 \leq \frac{4}{15} n^3$. If $n \leq 24$ or $n = 26$, then $m \leq \lfloor n^2/4 \rfloor$.*

Consequently, since $\sum_{v \in V(G)} d_G(v) = 2m$, Theorem 9 limits the combinations of vertex degrees. Adding these degree constraints, we could confirm the conjecture for $n \in \{14, \ldots, 17\}$. For the case $n = 18$, we used an additional theorem to further restrict the combinations:

▶ **Theorem 10** ([22]). *If G is a diameter-2-critical with n vertices and maximum degree $\geq 0.7 \cdot n$, then G has fewer than $\lfloor n^2/4 \rfloor$ edges.*

We give some details on the computation in Table 5.

▶ **Corollary 11.** *The Simon-Murty Conjecture holds for graphs with up to 18 vertices.*

Table 5 Confirming the Simon-Murty Conjecture for $n \in \{14, \ldots, 18\}$. Column n denotes the number of vertices, column #-comb the number of degree combinations, max-time the maximal runtime of a single combination, and total-time the accumulated runtime over all combinations. All times are given in seconds.

n	total time	max-time	#-comb
14	14512	131	1021
15	156116	604	4319
16	923660	2847	6494
17	11700237	19582	24067
18	46612962	216384	12974*

8 Conclusion

We presented SMS, a novel approach for SAT-based graph generation that utilizes dynamic symmetry breaking. A key ingredient of SMS is the concept of partially defined graphs and an algorithm that checks the lexicographical minimality of such graphs. We evaluated a prototype implementation of SMS on two showcase problems from Extremal Graph Theory, related to the graph invariants girth and diameter, respectively. We compared SMS with static symmetry breaking. We used the same encoding for the graph property and the same underlying SAT solver for both approaches. We think that this double strategy might be of independent interest, as it supports comparing the very same SAT-encoding on both methods. The experiments show encouraging results for SMS, in particular on unsatisfiable instances. As a side effect of our experiments on diameter-2-critical graphs, we could compute some values that haven't been known before and confirm the Simon-Murty Conjecture for graphs with up to 18 vertices.

We see several avenues for improving SMS in the future. An obvious area for improvement is the minimality check, where we currently use a relatively simple algorithm written from scratch. This leaves much room for improvement for algorithm design and engineering. The parameter frequency which controls the calls to the minimality check is currently a static parameter that stays constant for an entire SMS run. Here, dynamic changes of this parameter that depend on the current state of the solving progress could significantly increase the efficiency of SMS.

References

1 E. Abajo and A. Diánez. Exact value of $ex(n; \{C_3, \ldots, C_s\})$ for $n \leq \lfloor \frac{25(s-1)}{8} \rfloor$. *Discrete Math.*, 185:1–7, 2015. doi:10.1016/j.dam.2014.11.021.

2 Vikraman Arvind, Bireswar Das, and Johannes Köbler. A logspace algorithm for partial 2-tree canonization. In Edward A. Hirsch, Alexander A. Razborov, Alexei L. Semenov, and Anatol Slissenko, editors, *Computer Science - Theory and Applications, Third International Computer Science Symposium in Russia, CSR 2008, Moscow, Russia, June 7-12, 2008, Proceedings*, volume 5010 of *Lecture Notes in Computer Science*, pages 40–51. Springer, 2008. doi:10.1007/978-3-540-79709-8_8.

3 Rolf Backofen and Sebastian Will. Excluding symmetries in constraint-based search. *Constraints*, 7(3-4):333–349, 2002. doi:10.1023/A:1020533821509.

4 Olivier Bailleux and Yacine Boufkhad. Efficient CNF encoding of boolean cardinality constraints. In Francesca Rossi, editor, *Principles and Practice of Constraint Programming - CP 2003, 9th International Conference, CP 2003, Kinsale, Ireland, September 29 - October 3, 2003, Proceedings*, volume 2833 of *Lecture Notes in Computer Science*, pages 108–122. Springer, 2003. doi:10.1007/978-3-540-45193-8_8.

5 Clark W. Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. Satisfiability modulo theories. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 1267–1330. IOS Press, second edition, 2021.

6 Béla Bollobás. *Extremal graph theory*. Academic Press, 1978.

7 Louis Caccetta and Roland Häggkvist. On diameter critical graphs. *Discrete Math.*, 28(3):223–229, 1979. doi:10.1016/0012-365X(79)90129-8.

8 Ya-Chen Chen and Zoltán Füredi. Minimum vertex-diameter-2-critical graphs. *J. Graph Theory*, 50(4):293–315, 2005. doi:10.1002/jgt.20111.

9 Geoffrey Chu, Maria Garcia de la Banda, Christopher Mears, and Peter J. Stuckey. Symmetries, almost symmetries, and lazy clause generation. *Constraints*, 19(4):434–462, 2014. doi:10.1007/s10601-014-9163-9.

10 Michael Codish, Alice Miller, Patrick Prosser, and Peter J. Stuckey. Constraints for symmetry breaking in graph representation. *Constraints*, 24(1):1–24, 2019. doi:10.1007/s10601-018-9294-5.

11 Antoine Dailly, Florent Foucaud, and Adriana Hansberg. Strengthening the Murty-Simon conjecture on diameter 2 critical graphs. *Discrete Math.*, 342(11):3142–3159, 2019. doi:10.1016/j.disc.2019.06.023.

12 Jo Devriendt, Bart Bogaerts, and Maurice Bruynooghe. Symmetric explanation learning: Effective dynamic symmetry handling for SAT. In Serge Gaspers and Toby Walsh, editors, *Theory and Applications of Satisfiability Testing - SAT 2017 - 20th International Conference, Melbourne, VIC, Australia, August 28 - September 1, 2017, Proceedings*, volume 10491 of *Lecture Notes in Computer Science*, pages 83–100. Springer Verlag, 2017. doi:10.1007/978-3-319-66263-3_6.

13 Jo Devriendt, Bart Bogaerts, Broes De Cat, Marc Denecker, and Christopher Mears. Symmetry propagation: Improved dynamic symmetry breaking in SAT. In *IEEE 24th International Conference on Tools with Artificial Intelligence, ICTAI 2012, Athens, Greece, November 7-9, 2012*, pages 49–56. IEEE Computer Society, 2012. doi:10.1109/ICTAI.2012.16.

14 P. Erdős and A. Rényi. On a problem in the theory of graphs. *Magyar Tud. Akad. Mat. Kutató Int. Közl.*, 7:623–641 (1963), 1962.

15 Torsten Fahle, Stefan Schamberger, and Meinolf Sellmann. Symmetry breaking. In Toby Walsh, editor, *Principles and Practice of Constraint Programming - CP 2001, 7th International Conference, CP 2001, Paphos, Cyprus, November 26 - December 1, 2001, Proceedings*, volume 2239 of *Lecture Notes in Computer Science*, pages 93–107. Springer Verlag, 2001. doi:10.1007/3-540-45578-7_7.

16 Genghua Fan. On diameter 2-critical graphs. *Discret. Math.*, 67(3):235–240, 1987. doi:10.1016/0012-365X(87)90174-9.

17 David K. Garnick, Y. H. Harris Kwong, and Felix Lazebnik. Extremal graphs without three-cycles or four-cycles. *J. Graph Theory*, 17(5):633–645, 1993. doi:10.1002/jgt.3190170511.

18 Martin Gebser, Tomi Janhunen, and Jussi Rintanen. SAT modulo graphs: Acyclicity. In Eduardo Fermé and João Leite, editors, *Logics in Artificial Intelligence - 14th European Conference, JELIA 2014, Funchal, Madeira, Portugal, September 24-26, 2014. Proceedings*, volume 8761 of *Lecture Notes in Computer Science*, pages 137–151. Springer Verlag, 2014. doi:10.1007/978-3-319-11558-0_10.

19 Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Max Ostrowski, Torsten Schaub, and Philipp Wanko. Theory solving made easy with Clingo 5. In Manuel Carro, Andy King, Neda Saeedloei, and Marina De Vos, editors, *Technical Communications of the 32nd International Conference on Logic Programming, ICLP 2016 TCs, October 16-21, 2016, New York City, USA*, volume 52 of *OASICS*, pages 2:1–2:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/OASIcs.ICLP.2016.2.

20 Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Clingo = ASP + control: Preliminary report. *CoRR*, abs/1405.3694, 2014. arXiv:1405.3694.

21 Ian P. Gent, Karen E. Petrie, and Jean-François Puget. Symmetry in constraint programming. In Francesca Rossi, Peter van Beek, and Toby Walsh, editors, *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*, pages 329–376. Elsevier, 2006.

22 Teresa Haynes, Michael Henning, Lucas Merwe, and Anders Yeo. A maximum degree theorem for diameter-2-critical graphs. *Open Mathematics*, 12(12):1882–1889, 2014. doi:10.2478/s11533-014-0449-3.

23 Teresa W. Haynes, Michael A. Henning, Lucas C. van der Merwe, and Anders Yeo. Progress on the Murty-Simon Conjecture on diameter-2 critical graphs: a survey. *J. Comb. Optim.*, 30(3):579–595, 2015. doi:10.1007/s10878-013-9651-7.

24 Markus Kirchweger and Stefan Szeider. SAT modulo symmetries for graph generation, 2021. doi:10.5281/zenodo.5170575.

25 Mark H. Liffiton and Jordyn C. Maglalang. A cardinality solver: More expressive constraints for free - (poster presentation). In Alessandro Cimatti and Roberto Sebastiani, editors, *Theory and Applications of Satisfiability Testing - SAT 2012 - 15th International Conference, Trento, Italy, June 17-20, 2012. Proceedings*, volume 7317 of *Lecture Notes in Computer Science*, pages 485–486. Springer Verlag, 2012. doi:10.1007/978-3-642-31612-8_47.

26 Po-Shen Loh and Jie Ma. Diameter critical graphs. *J. Combin. Theory Ser. B*, 117:34–58, 2016. doi:10.1016/j.jctb.2015.11.004.

27 W. Mantel. Problem 28. *Wiskundige Opgaven*, 10:60–61, 1907.

28 João Marques-Silva and Sharad Malik. Propositional SAT solving. In Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors, *Handbook of Model Checking*, pages 247–275. Springer, 2018. doi:10.1007/978-3-319-10575-8_9.

29 Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, II. *J. Symbolic Comput.*, 60:94–112, 2014. doi:10.1016/j.jsc.2013.09.003.

30 Hakan Metin, Souheib Baarir, Maximilien Colange, and Fabrice Kordon. CDCLSym: introducing effective symmetry breaking in SAT solving. In Dirk Beyer and Marieke Huisman, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 24th International Conference, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings, Part I*, volume 10805 of *Lecture Notes in Computer Science*, pages 99–114. Springer, 2018. doi:10.1007/978-3-319-89960-2_6.

31 Amit Metodi and Michael Codish. Compiling finite domain constraints to SAT with BEE. *Theory Pract. Log. Program.*, 12(4-5):465–483, 2012. doi:10.1017/S1471068412000130.

32 Jean-Francois Puget. Symmetry breaking using stabilizers. In Francesca Rossi, editor, *Principles and Practice of Constraint Programming - CP 2003, 9th International Conference, CP 2003, Kinsale, Ireland, September 29 - October 3, 2003, Proceedings*, volume 2833 of *Lecture Notes in Computer Science*, pages 585–599. Springer Verlag, 2003. doi:10.1007/978-3-540-45193-8_40.

33 Jovan Radosavljević and Miodrag Živković. The list of diameter-2-critical graphs with at most 10 nodes. *IPSI Trans. Adv. Res.*, 16(1):1–5, 2020. URL: http://ipsitransactions.org/journals/papers/tar/2020jan/p9.pdf.

34 Bas Schaafsma, Marijn Heule, and Hans van Maaren. Dynamic symmetry breaking by simulating Zykov contraction. In Oliver Kullmann, editor, *Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings*, volume 5584 of *Lecture Notes in Computer Science*, pages 223–236. Springer Verlag, 2009. doi:10.1007/978-3-642-02777-2_22.

35 Jianmin Tang, Yuqing Lin, Camino Balbuena, and Mirka Miller. Calculating the extremal number $\mathrm{ex}(v;\{C_3,C_4,\ldots,C_n\})$. *Discr. Appl. Math.*, 157(9):2198–2206, 2009. doi:10.1016/j.dam.2007.10.029.

36 P. Wang, G. W. Dueck, and S. MacMillan. Using simulated annealing to construct extremal graphs. *Discrete Math.*, 235(1-3):125–135, 2001. Combinatorics (Prague, 1998). doi:10.1016/S0012-365X(00)00265-X.