# The Quest for Perfect and Compact Symmetry Breaking for Graph Problems

Marijn J.H. Heule
The University of Texas at Austin, United States

***Abstract*—Symmetry breaking is a crucial technique to solve many graph problems. However, current state-of-the-art techniques break graph symmetries only partially, causing search algorithms to unnecessarily explore many isomorphic parts of the search space. We study properties of perfect symmetry breaking for graph problems. One promising and surprising result on small-sized graphs —up to order five— is that perfect symmetry breaking can be achieved using a compact propositional formula in which each literal occurs at most twice. At least for small graphs, perfect symmetry breaking can be expressed more compactly than the existing (partial) symmetry-breaking methods. We present several techniques to compute perfect symmetry-breaking formulas and analyze them.**

## I. Introduction

Over the last two decades, the speed and capacity of satisfiability (SAT) solvers has improved by several orders of magnitude, enabling solutions to some long-standing open problems such as Erdős' discrepancy problem [14] and the Boolean Pythagorean triples problem [10]. However, a main weakness of SAT solvers in some applications is their inability to capitalize on symmetries, that is, avoiding needless exploration of isomorphic sub-problems. Several methods have been proposed to counter this weakness, in particular by adding symmetry-breaking predicates [7]. Existing methods are not strong enough to make the SAT approach successful for long-standing open problems in graph theory, such as computing Ramsey numbers [9]. We present a novel approach to address symmetries in graph problems in order to make advances towards solving some of these open problems.

For hard combinatorial problems with few symmetries, such as Van der Waerden numbers [8], [15] and Erdős' discrepancy problem, general purpose methods, in particular SAT solvers, are the current state-of-the art. However, hard combinatorial problems with lots of symmetries, such as Ramsey numbers, are still best solved using dedicated approaches. Although SAT has been applied to Ramsey numbers [16], [5], the most impressive result, computing $R(4,5)$ [18], is two decades old and has not been reproduced with general purpose methods.

This contrast can be explained by a gap in ability to fully break all symmetries. When there are just a few symmetries, it is relatively easy to break them using a small predicate, so solvers can avoid isomorphic parts of the search space. However, when there are many symmetries, such as when permuting all the vertices of a graph, then there is no sub-exponential method that can fully break them yet.

The current state-of-the-art symmetry-breaking methods for SAT [1] or specifically for graphs [6] are unable to break all symmetries for graph problems of order five and larger. We will show that the average number of active graphs per isomorphism class —after symmetry breaking, with both methods— is quadratic in the size of the graph. Any *perfect* symmetry-breaking technique would ensure that only one graph is active per isomorphism class. Slightly reducing the average number of active graphs per isomorphism class clearly improves performance [6]. Any method that would perfectly break graph symmetries is therefore expected to boost the capabilities of general purpose solvers significantly.

The question arises: how expensive is it to perfectly break all graph symmetries? We decided to use the number of clauses required to achieve perfect symmetry breaking as the measurement. The main motivation for this focus is that more high-level measurements could be expressed using clauses, while this does not hold for the other way around. Consequently, there may exist polynomial-sized perfect symmetry breaking for graph problems using clauses, while high-level representations might be exponential in size.

In this paper we present several approaches to answering that question. One surprising result is that breaking all graph symmetries may be possible with compact predicates. For example, up to order five, the largest size for which we could compute optimal results, literals occur at most twice in the smallest predicates. Moreover, our compact and perfect predicates are smaller than the most compact representation of existing (partially) symmetry-breaking methods, at least for small graphs.

We present our study of perfect symmetry breaking for graph problems using the concept of *isolators*: predicates, over Boolean variables representing potential edges of graphs of a given order, which rule out only redundant graphs. We developed algorithms to compute isolators that are perfect (break all symmetries) or optimal (perfect and minimal in size). We show that interesting patterns can be observed in the graphs that are admitted by optimal isolators.

## II. Background and Related Work

We denote by $\mathcal{G}_k$ the set of all labeled, undirected graphs of order $k$. Graphs $G, H \in \mathcal{G}_k$ are in the same *isomorphism class* if $G$ can be obtained by relabeling the vertices of $H$.

**Example 1.** Consider the set of all labeled, undirected graphs of order three using the vertex labels $a$, $b$, and $c$. We will represent graphs as a set of edges where each edge is written as the two vertices it connects. $\mathcal{G}_3$ is:

$$\{\{\},\{ab\},\{ac\},\{bc\},\{ab,ac\},\{ab,bc\},\{ac,bc\},\{ab,ac,bc\}\}$$

Graphs $\{ab, ac\}$ and $\{ac, bc\}$ are in the same isomorphism class, because $\{ab, ac\}$ can be obtained from $\{ac, bc\}$ by swapping the vertex labels $a$ and $c$.

A *graph existence problem* of order $k$ asks whether there exists an unlabeled, undirected graph of order $k$ with a given property. Since the graphs are unlabeled, only one graph from each isomorphism class needs to be considered. The Ramsey numbers are famous graph existence problems. Graph existence problems have been thoroughly studied, as can be observed in a survey pointing to over 600 papers on the subject [19].

The state-of-the-art symmetry-breaking tool for SAT problems (not restricted to graph problems) is `shatter` [1]. For graph existence problems, the symmetries —detected on the clausal level— correspond to permutations of the vertices. Given a graph existence problem of order $k$, `shatter` adds symmetry-breaking predicates that sort the vertices. The addition of the predicates can reduce the SAT solving time by orders of magnitude.

More specifically, let the vertices be named $v_1, \dots, v_k$. Given a graph $G$, $A_{i,j}$ denotes the $i^{\text{th}}$ row of the adjacency matrix of $G$ without columns $i$ and $j$. Symmetry-breaking predicate $p_{\preceq}(v_i, v_j)$ enforces a lexicographic order between $A_{i,j}$ and $A_{j,i}$, denoted by $A_{i,j} \preceq A_{j,i}$. Predicate $p_{\preceq}(v_i, v_j)$ can be encoded with about $6k$ clauses using auxiliary (non-edge) variables. Using only the edge variables, it costs about $2^k$ clauses to express this constraint. Hence auxiliary variables can reduce the encoding from exponential to linear in the number of vertices.

The symmetry-breaking clauses added by `shatter` to graph existence problems of order $k$ correspond to the constraint $p_{\preceq}(v_1, v_2) \wedge p_{\preceq}(v_2, v_3) \wedge \dots \wedge p_{\preceq}(v_{k-1}, v_k)$. We will call this symmetry-breaking technique the quad method as it adds $\mathcal{O}(k^2)$ clauses. Codish *et al.* [6] made two observations regarding the predicates $p_{\preceq}(v_i, v_j)$ for graph existence problems: i) $p_{\preceq}(v_i, v_j)$ is not transitive; and ii) it is valid to add all predicates $p_{\preceq}(v_i, v_j)$ with $1 \leq i < j \leq k$ to graph existence problems. We will refer to this latter method as the cubic method as it adds $\mathcal{O}(k^3)$ clauses.

We define the *redundancy ratio* of a graph symmetry-breaking method as the ratio between the number of assignments that satisfy the predicates and the number of isomorphism classes. One can view the redundancy ratio as the average number of graphs per isomorphism class that are not eliminated by a graph symmetry-breaking method. We call a graph symmetry breaking *perfect* for order $k$ if the redundancy ratio is one for order $k$.

Figure 1 shows the redundancy ratios of the quad and cubic methods, which are only perfect up to order four. The cubic method outperforms the quad method, but for both methods, the redundancy ratio increases almost quadratically for higher orders within the experimental range: approximately $(k-5)^2$ for quad and $(k-6)^2$ for cubic. Although their difference in redundancy ratio is modest, the cubic method is able to solve some graph existence problems that are too hard for the quad method [6] to solve. Therefore, it is expected that a perfect graph symmetry-breaking technique would boost performance on graph existence problems significantly.

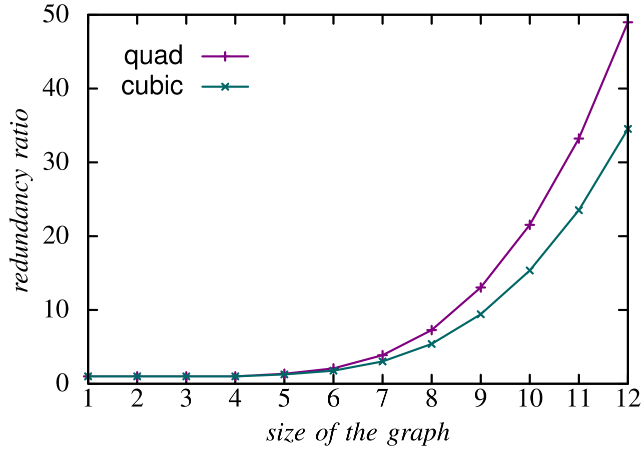


Fig. 1. The redundancy ratios of the quad and cubic symmetry-breaking methods.

A recent paper [12] presents a perfect symmetry-breaking approach based on so-called canonical sets. This approach realizes a redundancy ratio of one, but has several disadvantages. Most importantly, the number of clauses and variables required to express these symmetry-breaking predicates grows exponentially in the size of the graph. For example, perfect symmetry-breaking for graphs of order five via canonical sets uses 225 clauses and 55 variables. In contrast, the method we propose in this paper produces perfect symmetry breaking for graphs of order five using only 12 clauses and 10 variables. Due to the exponential growth, it is impossible to use this method for any graph existence problems of order 11 and higher. Additionally, the canonical sets method does not allow us to answer the main question of this paper: how expensive is it to perfectly break all graph symmetries? Canonical sets are only able to express a subset of the possible symmetry-breaking options. In particular, our compact perfect symmetry-breaking predicates cannot be expressed using canonical sets.

## III. Perfect Isolators and Canonical Forms

Consider propositional formulas over variables representing all possible edges between $k$ vertices. We say that a graph $G \in \mathcal{G}_k$ is *admitted* by such a formula $F$ if there exists a satisfying assignment of $F$ in which each edge variable is assigned to true if and only if the edge occurs in $G$. An *isolator* of $\mathcal{G}_k$, written $I$, is such a formula that admits at least one graph in each isomorphism class of $\mathcal{G}_k$. We write each edge's variable and its positive literal in the same way as the edge itself. Negation of literals is notated with an overline.

**Example 2.** Consider the isolator $I_{\text{ex}} := (ab \vee \overline{ac}) \wedge (ac \vee \overline{bc})$ of $\mathcal{G}_3$ using the vertex labels $a$, $b$, and $c$. Four full assignments satisfy $I_{\text{ex}}$ (using $\mathbf{t}$ for true and $\mathbf{f}$ for false):

$$ab = \mathbf{f}, ac = \mathbf{f}, bc = \mathbf{f}; \qquad ab = \mathbf{t}, ac = \mathbf{f}, bc = \mathbf{f};$$
$$ab = \mathbf{t}, ac = \mathbf{t}, bc = \mathbf{f}; \qquad ab = \mathbf{t}, ac = \mathbf{t}, bc = \mathbf{t}.$$

These assignments correspond to the following four graphs:

$$\{\};\quad \{ab\};\quad \{ab, ac\};\quad \{ab, ac, bc\}.$$

Observe that each graph occurs in a different isomorphism class as each graph has a different number of edges.

Throughout this paper, we distinguish three special types of isolators. The *trivial* isolator equals the empty formula and thus admits all graphs $G \in \mathcal{G}_k$. A *perfect* isolator admits exactly one graph from each isomorphism class. An *optimal* isolator is a perfect isolator with a minimal number of clauses. $I_{\text{ex}}$ in Example 2, which is equivalent to $P'_3$ in Example 3, is an optimal isolator for $\mathcal{G}_3$. Notice that a perfect isolator breaks *all* graph symmetries in graph existence problems, i.e. the reduction ratio is one.

A *canonical labeling* $\mathcal{C}$ of $\mathcal{G}_k$ is a subset of $\mathcal{G}_k$ containing exactly one graph from each isomorphism class. Given a canonical labeling $\mathcal{C}$, a graph $G \in \mathcal{C}$ is the *canonical form* of all graphs occurring in the isomorphism class of $G$. Several canonical labeling algorithms have been implemented, such as `nauty` [17] and `bliss` [13]. For each perfect isolator $I$ of $\mathcal{G}_k$, there is an induced canonical labeling $\mathcal{C}$, containing the graphs that are admitted by $I$. As we will show below, it is also possible to convert a canonical labeling into a perfect isolator.

**Example 3.** Consider the graphs of order three with vertex labels $a$, $b$, and $c$. There are four isomorphism classes of $\mathcal{G}_3$: graphs with zero edges, one edge, two edges, and three edges. There are two different canonical labelings of $\mathcal{G}_3$ (modulo vertex renaming) which are shown below as $\mathcal{C}_3$ and $\mathcal{C}'_3$.

$$\mathcal{C}_3 := \{\{\}, \{ab\}, \{ac, bc\}, \{ab, ac, bc\}\}$$
$$\mathcal{C}'_3 := \{\{\}, \{ab\}, \{ab, ac\}, \{ab, ac, bc\}\}$$

For both canonical labelings there exists a perfect isolator consisting of two binary clauses with Boolean variables $ab$, $ac$, and $bc$ expressing that edges $ab$, $ac$, and $bc$ are present.

$$P_3 := (ac \vee \overline{bc}) \wedge (\overline{ac} \vee bc) \quad // \text{ equals}: \ ac \leftrightarrow bc$$
$$P'_3 := (ab \vee \overline{ac}) \wedge (ac \vee \overline{bc}) \quad // \text{ equals}: \ bc \rightarrow ac \rightarrow ab$$

A canonical labeling can easily be converted into a perfect isolator, albeit one of exponential size. Let $L(G)$ denote the representation of a graph $G$ as a set of literals: $L(G)$ contains for each present edge in $G$ the corresponding positive literal, and for each absent edge the corresponding negative literal. For example, take a graph $G \in \mathcal{G}_4$: if $G = \{ab, ad, bc, cd\}$, then $L(G) = \{ab, \overline{ac}, ad, bc, \overline{bd}, cd\}$. Let $\mathcal{C}$ be a canonical labeling of $\mathcal{G}_k$. A perfect isolator in disjunctive normal form (DNF) based on $\mathcal{C}$ can be constructed as follows:

$$P_{\text{DNF}} := \bigvee_{G \in \mathcal{C}} \left(\bigwedge l \in L(G)\right)$$

The size of any $P_{\text{DNF}}$ of $\mathcal{G}_k$ is exponential in $k$, because the number of isomorphism classes is exponential in $k$. In order to use such isolators for SAT solving, a transformation into CNF is required. We write $P_{\text{CNF}}$ for the Tseitin transformation [22] of $P_{\text{DNF}}$. $P_{\text{CNF}}$ is larger than $P_{\text{DNF}}$ by a factor of about $k^2$.

The size of the isolator $P_{\text{CNF}}$ can be reduced significantly. There exist two tools that can simplify propositional formulas: `espresso` [4] and `bica` [11]. Both tools can simplify a formula to its smallest CNF representation. We denote by $P_{\text{simp}}$ the smallest formula in CNF that is logically equivalent to $P_{\text{DNF}}$. The sizes of different representations of perfect isolators based on `nauty`'s canonical labelings are shown in Table I. Computing the $P_{\text{DNF}}$ and $P_{\text{CNF}}$ is cheap, but computing $P_{\text{simp}}$ with `bica` is costly for larger graphs (seconds for $k = 6$, minutes for $k = 7$, and hours for $k = 8$).

TABLE I
THE SIZE OF PERFECT ISOLATORS IN CUBES ($P_{\text{DNF}}$) OR IN CLAUSES ($P_{\text{CNF}}$ AND $P_{\text{simp}}$) BASED ON THE `nauty`'S CANONICAL LABELINGS AND FORMULA SIMPLIFICATIONS BY `bica`.

| $k$ | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| $\lvert P_{\text{DNF}}\rvert$ | 2 | 4 | 11 | 34 | 156 | 1,044 | 12,346 |
| $\lvert P_{\text{CNF}}\rvert$ | 3 | 13 | 67 | 341 | 2,341 | 21,925 | 345,689 |
| $\lvert P_{\text{simp}}\rvert$ | 0 | 2 | 9 | 24 | 77 | 311 | > 1,839 |

We also simplified canonical labelings produced by `bliss`. The sizes of the resulting simplified formulas were similar to those produced via `nauty`. However, `bica` is significantly slower in reducing the `bliss`-based formulas. We tried to use `espresso`, but it is not powerful enough to minimize perfect isolators of order six and larger.

Although the sizes of $P_{\text{simp}}$ are minimal for a given canonical labeling, much smaller perfect isolators may exist for other canonical labelings. An optimal isolator of $\mathcal{G}_k$ is the smallest $P_{\text{simp}}$ among *all* canonical labelings of $\mathcal{G}_k$.

## IV. Optimal Isolators via Satisfiability Solving

Perfect isolators of order four and up are hard to compute. As a potential solution, we propose to translate the optimal isolator problem into Boolean satisfiability (SAT). Let $F_{k,m}$ be the SAT problem encoding that there exists a perfect isolator of order $k$ consisting of $m$ clauses. We will refer to such clauses as *isolator clauses*. To find an optimal isolator for a given $k$, we need to find an $m$ such that $F_{k,m}$ is satisfiable, while $F_{k,m-1}$ is unsatisfiable. We first describe some details about the encoding of $F_{k,m}$ followed by some results.

### A. Encoding

Let $E_k$ be the set of edges that occur in graphs in $\mathcal{G}_k$. Set $L_k$ contains a positive and negative literal for each element in $E_k$. The main variables used in the encoding of $F_{k,m}$, namely $x_{l,i}$ with $l \in L_k$ and $i \in \{1, \ldots, m\}$, describe the isolator clauses $C_i$ and are defined as follows:

$$x_{l,i} := \begin{cases} \mathbf{t} & \text{if } l \in C_i \\ \mathbf{f} & \text{otherwise} \end{cases}$$

Additionally, we have variables $y_{G,i}$ denoting that isolator clause $C_i$ satisfies graph $G \in \mathcal{G}_k$. An isolator clause $C_i$ satisfies a graph $G$ if and only if there exists a literal $l \in C_i$ such that $l \in L(G)$. This can be encoded with $m \cdot |E_k| \cdot |\mathcal{G}_k|$

binary clauses and $m \cdot |\mathcal{G}_k|$ clauses of length $|E_k|$ which together represent the following definition using the logical OR constraint:

$$y_{G,i} := \text{OR}(\{x_{l,i} \mid l \in L(G)\})$$

Notice that the above encoding quickly becomes very large. For example, using $k = 6$, the number of clauses is close to $m \cdot 10^6$. Using auxiliary variables, the above constraint can be encoded with $2m \cdot |\mathcal{G}_k|$ binary clauses and $m \cdot |\mathcal{G}_k|$ clauses of length $|E_k|$.

Finally, variables $z_G$ denote whether graph $G$ is satisfied by all $m$ isolator clauses, or, equivalently, whether graph $G$ is admitted by the isolator. This can be realized by the straightforward encoding of the following logical AND constraint, requiring $\mathcal{O}(m \cdot |\mathcal{G}_k|)$ clauses.

$$z_G := \text{AND}(y_{G,1}, \ldots, y_{G,m})$$

The only constraints in $F_{k,m}$ that are not definitions, express that exactly one graph from each isomorphism class is satisfied by all $m$ isolator clauses. This graph can be seen as the canonical form of that isomorphism class. Let $\mathcal{I}_k$ denote the partitioning of $\mathcal{G}_k$ into isomorphism classes. For each isomorphism class $I \in \mathcal{I}_k$, we add the following ExactlyOne constraint, for which compact encodings exist [20]:

$$\text{ExactlyOne}(\{z_G \mid G \in I\})$$

To guide the solver, some redundant clauses can be added, for example ensuring that $C_i$ cannot be a tautology.

### B. Results

Using the encoding described above and solving the formulas with `glucose 3.0` [2] and `treengeling` [3], we computed optimal isolators for graphs up to order five[1]. For graphs of order six or larger, we were not able to compute an upper bound, i.e., find a satisfying assignment for any $F_{k,m}$ using parallel SAT solvers running on 24 cores with a 24 hour time limit. Crucial for the lower bound (UNSAT) results is breaking the symmetry of the isolator clauses, which is realized by adding constraints that enforce a lexicographic order between the isolator clauses. Table II shows the results of the experiments.

TABLE II
Statistics of SAT solving optimal isolator problems. `glucose` (G) was the fastest solver on the easy formulas, while `treengeling` (T) was stronger on hard formulas. Runtimes are in wall clock seconds running on a quad core Intel Xeon E31280 CPU.

| *formula* | *result* | *variables* | *clauses* | *runtime* |
|---|---|---|---|---|
| $F_{4,6}$ | UNSAT | 756 | 2,458 | 0.18 (G) |
| $F_{4,7}$ | SAT | 861 | 2,827 | 0.01 (G) |
| $F_{5,11}$ | UNSAT | 14,480 | 54,756 | 3,510.36 (T) |
| $F_{5,12}$ | SAT | 15,609 | 59,281 | 102.69 (G) |

[1]The isolators and CNF formulas mentioned in this paper are available at http://www.cs.utexas.edu/~marijn/isolator/

An optimal isolator of order four, $P_4$, shown below, consists of seven clauses: five binary and two ternary. Notice that $P_4$ is a renamable Horn formula, as are the optimal isolators of order three (recall $P_3$ and $P'_3$).

$$\begin{aligned} P_4 \;:=\; & (ad \vee \overline{bd}) \wedge (bd \vee \overline{ac}) \wedge (cd \vee \overline{bc}) \wedge (ab \vee \overline{bc}) \wedge \\ & (bc \vee \overline{ac}) \wedge (ab \vee bd \vee \overline{cd}) \wedge (bc \vee bd \vee \overline{ad}) \end{aligned}$$

Optimal isolators of order five, such as $P_5$ below, consist of only twelve clauses. It is surprising to see that such a small formula —just slightly larger than the number of edges, similar to order four— admits exactly one graph from each of the 34 isomorphism classes. Moreover, all clauses in $P_5$, apart from the last one, have length three or less.

$$\begin{aligned} P_5 \;:=\; & (ad \vee \overline{bd}) \wedge (bd \vee \overline{ac}) \wedge (cd \vee \overline{bc}) \wedge (bc \vee \overline{ad}) \wedge \\ & (ae \vee \overline{ce}) \wedge (be \vee \overline{ae}) \wedge (ab \vee bd \vee \overline{cd}) \wedge \\ & (ae \vee de \vee \overline{be}) \wedge (ad \vee ce \vee \overline{de}) \wedge (ab \vee \overline{cd} \vee \overline{de}) \wedge \\ & (ac \vee \overline{ad} \vee \overline{ce}) \wedge (ce \vee \overline{ab} \vee \overline{ae} \vee \overline{bc}) \end{aligned}$$

Optimal isolators $P_4$ and $P_5$ have four clauses in common: the first three binary clauses and the first ternary clause. Another property they share is that each literal occurs at most twice. If the latter holds for optimal isolators of larger orders —though unlikely— then their size would be linear in the number of edge variables.

### C. Analysis of Optimal Isolators

We studied the canonical labelings induced by optimal isolators. Figure 2 visualizes the canonical labelings induced by the optimal isolators $P_3$, $P_4$, and $P_5$. We call two canonical forms *connected* if they differ by exactly one edge. In Figure 2 connections are shown with an arrow from the graph without the edge to the graph with the edge.

Notice that there are several similarities in these visualizations. For example, in all three cases, there are two root canonical forms (i.e., graphs without incoming arcs): the edge-less graph and a path of two edges. Furthermore, the canonical forms of the single edge graph and the two-edge path together form a triangle. We also looked at visualizations of the canonical labelings produced by `nauty`, `bliss`, and `shatter`. The latter pattern (the triangle) is *not* present in those canonical labelings.

The order in which edges are added starting from the empty graph are similar. Comparing the visualizations of $P_4$ and $P_5$ reveals that edges are added in the following order: $ab$, $cd$, $bc$, $ad$, $bd$, and $ac$. Also the canonical form of order $k$ of the star with $k-1$ edges has the vertex with the highest label as center of the star. Finally, notice that the canonical forms admitted by $P_5$ are either part of a chain or a big cluster.

These and other patterns may provide some insight in how to construct compact isolators for orders larger than five.

## V. Perfect Isolators via Random Probing

Above, we discussed two methods for computing perfect isolators: i) simplifying a formula representing a canonical

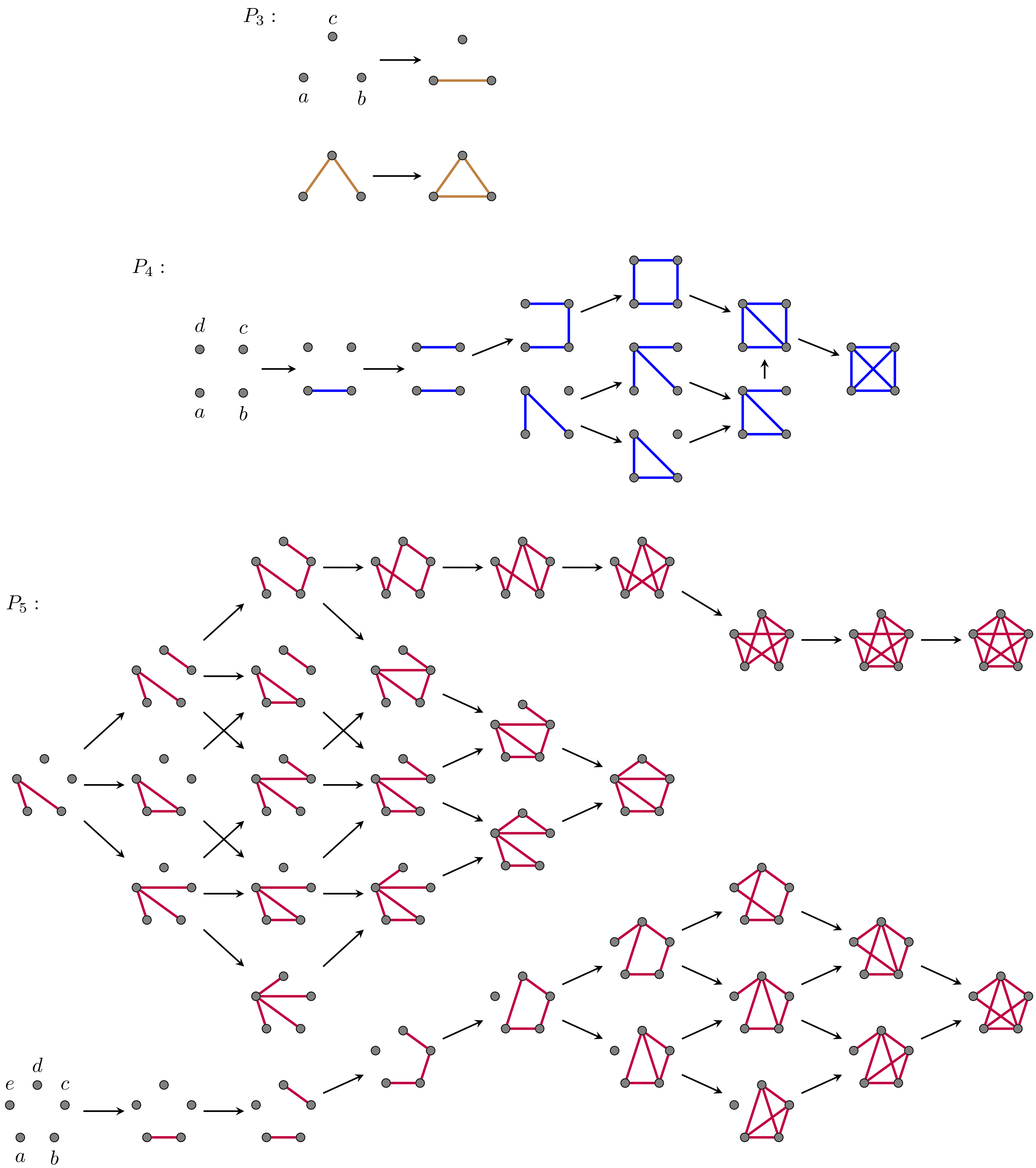


Fig. 2. The canonical forms of graphs based on the smallest perfect isolators $P_3$ (top), $P_4$ (middle), and $P_5$ (bottom). When two graphs differ by exactly one edge, there is an arrow from the graph without the edge to the graph with the edge.

labeling; and ii) encoding the problem into SAT. The first method works for graphs up to order eight, but the resulting isolators are relatively large. The second method can compute optimal isolators up to order five, but cannot deal with larger graphs. In this section, we present a third method which scales reasonably well, while producing more compact perfect isolators than the first method.

### A. Random Probing Algorithm

The last method we present to compute perfect isolators is based on *random probing*. The algorithm starts with the trivial isolator. In each step, a clause is added to the isolator using some randomized heuristics. The algorithm terminates when the isolator becomes perfect.

The trivial isolator admits all graphs, while a perfect isolator admits only one graph per isomorphism class. In order to compute a compact perfect isolator, one wants to pick a clause to extend the current isolator that reduces the number of graphs that are admitted by the isolator as much as possible — bringing it closer to a perfect isolator. Yet not all clauses can be picked as it is required that at least one graph is admitted from each isomorphism class.

The greedy version of the randomized probing algorithm picks a clause that reduces the number of graphs admitted by the isolator the most, breaking ties randomly. More specifically, the *reduction measurement* of a clause with respect to an isolator is the number of graphs that are admitted by the isolator, but no longer admitted once the clause is added to the isolator. The algorithm that always picks a clause with the highest reduction measurement is not able to compute an optimal isolator for graphs of order five, regardless of how ties are broken — because there is no optimal isolator that contains clauses with only the highest reduction measurement.

The algorithm needs two improvements to find optimal isolators of graphs of order five. The first improvement ranks all the clauses based on the reduction measurement, again breaking ties randomly. But instead of picking the top ranked clause, the new algorithm picks the $n^{\text{th}}$ element in the ranking with probability $0.5^n$. So with $50\%$ chance the top element is picked, with $25\%$ chance the second element is picked, etc.

After this modification, the algorithm could in theory compute any perfect isolator, although the probability for most of them is extremely small. In practice, the algorithm does not find an optimal isolator of graphs of size five after millions of random probes with a very high probability. The main reason is that most top ranked clauses perform exactly the same reduction, i.e., the set of graphs that are ruled out by those clauses is exactly the same. Consequently, it does not matter whether you pick the first, second, or third ranked clause, because in most cases they are equivalent as a candidate for extending the isolator.

The second modification was developed to counter this effect. Apart from a ranking, each clause gets a hash value based on the set of graphs that are ruled out by that clause. In case multiple clauses have the same reduction and hash value, only one of them appears in the ranking and the other ones are ignored.

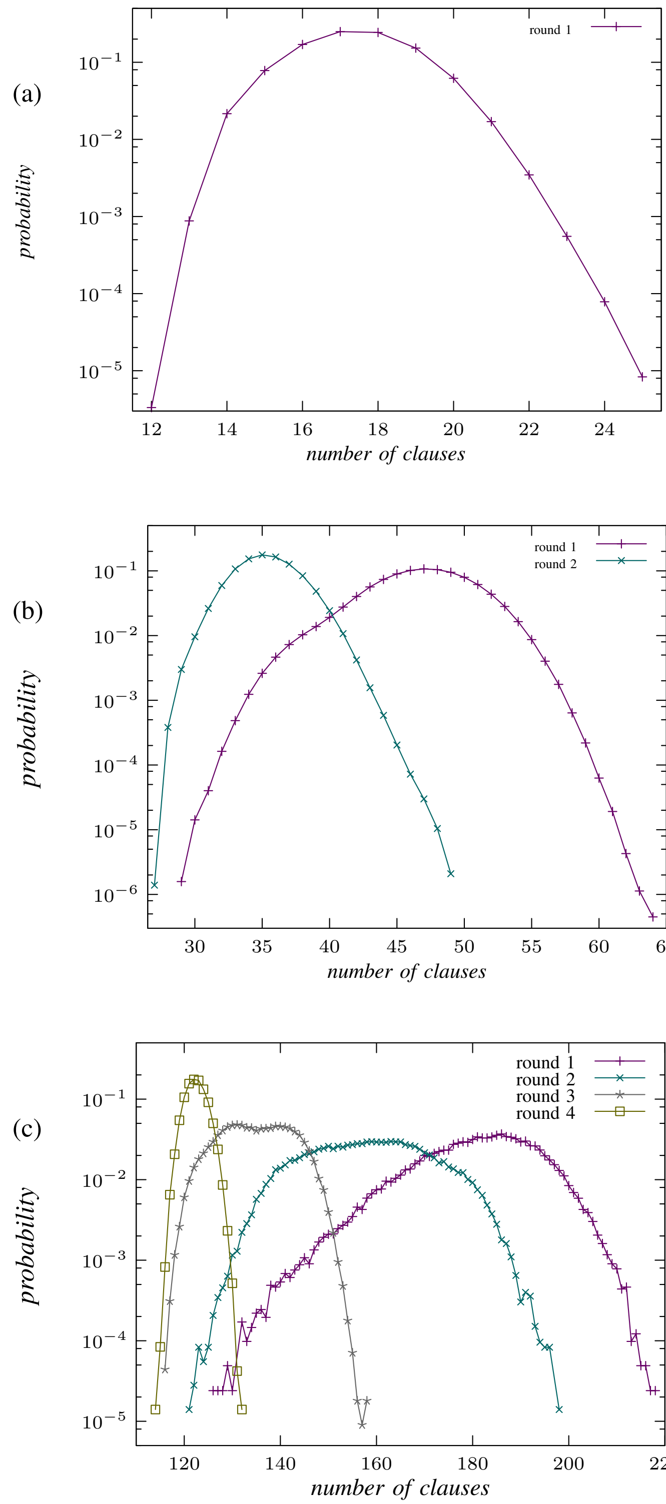


Fig. 3. Distribution of the size of perfect isolators using the random probing algorithm on graphs of order five (a), six (b), and seven (c). Round 1 experiments used the trivial isolator as starting points. Round $r > 1$ experiments initialized isolators using the first $10(r-1)$ clauses of one of the 50 best probes of Round $r-1$.

### B. Implementation Optimizations

Several optimizations were implemented to perform random probing reasonably efficiently. The initial stable version was too slow to perform large-scale experiments. The optimizations described below improved the performance by more than two orders of magnitude when computing isolators of order six and larger.

First, the results of one step can be partially reused for the next step. Clauses can be partitioned into three sets: conflicting, redundant, and useful clauses. Conflicting clauses rule out all remaining graphs in some isomorphism class. Redundant clauses admit all remaining graphs in all isomorphism classes. Useful clauses rule out some remaining graphs, but still admit at least one graph in each isomorphism class. Once a clause is known to be conflicting or redundant, it can be ignored from that point onwards, as it will stay conflicting or redundant in future steps.

Second, further implementation optimizations can be derived from the subsumption relation between clauses: if a clause $C$ is subsumed by a clause $D$, then the reduction of $C$ is less or equal to the reduction of $D$. Since we are interested in useful clauses with a high reduction, a clause is ignored if there exists at least one useful clause that subsumes it. Moreover, if a clause $D$ is redundant, then all clauses $C \supset D$ are redundant as well. Hence, all clauses that are subsumed by redundant clauses can be marked redundant without computing their reduction. The subsumption relation is checked efficiently using a hash table.

### C. Results

We ran the random probing algorithm starting with the trivial isolators of orders five to seven. The results of two million random probes on order five are shown in Figure 3 (a). With a high probability, the random probing algorithm computes a perfect isolator around seventeen clauses long. With a very small probability, slightly more than one in a million, the algorithm computes an optimal isolator of order five, consisting of only twelve clauses. The improvements discussed in Section V-B were crucial to finding optimal isolators. The average runtime of a single probe is approximately 0.02 seconds. Although a single probe is cheap, computing an optimal isolator using randomized probing is relatively expensive as it may require hundreds of thousands of probes. The SAT solving approach is much more efficient since it can compute an optimal isolator of order five in a few minutes.

Random probing for isolators of order six are shown in Figure 3 (b). Using the same setup as with order five, the smallest perfect isolator after 400,000 probes consisted of 29 clauses, with each probe running for about 0.5 seconds. In order to improve these results, the smallest 50 isolators discovered were used as starting points for a second round of 400,000 probes. For this second round, the first step consists of choosing the first ten clauses of one of the 50 best isolators. After this initialization, the probing algorithm continued as usual. During this second round, perfect isolators were discovered consisting of only 27 clauses.

The random probing algorithm was somewhat changed for perfect isolators of order seven: we turned the first modification off, i.e., always picked the highest ranked clause, because it resulted in smaller perfect isolators. This is probably caused by the smaller sample size (80,000 probes per round), which was necessary because for order seven, the runtime of a single probe was on average 7 minutes. The smallest isolator we found consisted of 114 clauses after four rounds. Details are shown in Figure 3 (c).

Computing a perfect isolator of order eight required starting with a non-trivial isolator, because the number of initial graphs, $|\mathcal{G}_8| = 2^{28}$, was too large for our implementation to handle. We used the symmetry-breaking predicate of the quad method of order eight (see Section II) as the initial isolator, which consists of 170 clauses and adds 28 auxiliary variables. A single probe with that starting point resulted in a perfect isolator of 956 total clauses in two days.

The focus of this paper is on computing small perfect isolators and not yet on exploiting them. However, we believe that small perfect isolators are not only interesting from a theoretical point of view, but also from a practical one. For example, Itzhakov and Codish [12] determine the number of graphs that have no clique and no co-clique of size four (also known as Ramsey $R(4,4,k)$ graphs) and claw-free graphs after perfect symmetry breaking. Table III shows that breaking symmetries using perfect isolators produced by the random probing results in much smaller formulas for which all solutions can be computed much faster.

TABLE III
COMPARISON OF THE CANONICAL SETS METHOD AND PERFECT ISOLATORS BY RANDOM PROBING ON THE SIZE OF THE SYMMETRY-BREAKING PREDICATES ($n$ DENOTES NUMBER OF VARIABLES, AND $m$ DENOTES NUMBER OF CLAUSES) AND THE COSTS TO COMPUTE ALL SOLUTIONS ON RAMSEY $R(4,4,k)$ GRAPHS AND CLAW-FREE $CF(k)$ GRAPHS. COSTS FOR CANONICAL SETS ARE TAKEN FROM [12], WHILE WE COMPUTED ALL SOLUTIONS USING sharpSAT [21].

| | $F$ | | $F$ + canonical sets | | | $F$ + probe isolator | | |
|---|---|---|---|---|---|---|---|---|
| *problem* | $n$ | $m$ | $n$ | $m$ | time | $n$ | $m$ | time |
| $R(4,4,6)$ | 15 | 30 | 72 | 315 | 0.01 | **15** | **57** | **0.00** |
| $R(4,4,7)$ | 21 | 70 | 286 | 1395 | 0.05 | **21** | **184** | **0.01** |
| $R(4,4,8)$ | 28 | 140 | 2177 | 10885 | 1.69 | **56** | **1096** | **0.04** |
| $CF(6)$ | 15 | 60 | 72 | 345 | 0.01 | **15** | **87** | **0.00** |
| $CF(7)$ | 21 | 140 | 286 | 1465 | 0.03 | **21** | **254** | **0.01** |
| $CF(8)$ | 28 | 280 | 2177 | 11025 | 1.08 | **56** | **1236** | **0.03** |

## VI. CONCLUSIONS

We studied the concept of perfect isolators for small graphs. One surprising and encouraging result is that there exist very small perfect isolators for graphs up to order five — the largest order for which we could compute optimal (smallest perfect) isolators. For graphs up to order eight, perfect isolators were obtained via a random probing algorithm. These isolators are likely not optimal.

The main question that remains unanswered is the growth rate of optimal isolators. Focussing only at the known optimal isolators, the growth rate appears to be quadratic in the size of the graph: All optimal isolators of order $k$ have approximately (but fewer than) $|E_k| + k$ clauses. However, when the best (non-optimal) results of larger graphs are taken into account, the growth rate appears much steeper. This discrepancy might be explained by the lack of using auxiliary variables when constructing perfect isolators. Auxiliary variables are crucial to realize compact (partial) symmetry-breaking predicates via existing methods.

In future research we want to compute optimal and perfect isolators for graphs of larger orders. We expect that such isolators will be helpful in tackling hard graph existence problems, such as Ramsey numbers.

**Acknowledgements** The author is supported by the National Science Foundation under grant number CCF-1526760 and acknowledges the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing grid resources that have contributed to the research results reported within this paper.

## REFERENCES

[1] Fadi A. Aloul, Karem A. Sakallah, and Igor L. Markov. Efficient symmetry breaking for boolean satisfiability. In Georg Gottlob and Toby Walsh, editors, *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 271–276. Morgan Kaufmann, 2003.

[2] Gilles Audemard and Laurent Simon. Predicting learnt clauses quality in modern SAT solvers. In Craig Boutilier, editor, *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 399–404, 2009.

[3] Armin Biere. Lingeling, Plingeling and Treengeling entering the SAT competition 2013. *Proceedings of SAT Competition 2013*, page 51, 2013.

[4] Robert King Brayton, Alberto L. Sangiovanni-Vincentelli, Curtis T. McMullen, and Gary D. Hachtel. *Logic Minimization Algorithms for VLSI Synthesis*. Kluwer Academic Publishers, Norwell, MA, USA, 1984.

[5] Michael Codish, Michael Frank, Avraham Itzhakov, and Alice Miller. Computing the ramsey number R(4, 3, 3) using abstraction and symmetry breaking. *CoRR*, abs/1510.08266, 2015.

[6] Michael Codish, Alice Miller, Patrick Prosser, and Peter J. Stuckey. Breaking symmetries in graph representation. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 510–516. AAAI Press, 2013.

[7] James Crawford, Matthew Ginsberg, Eugene Luks, and Amitabha Roy. Symmetry-breaking predicates for search problems. In *Proc. KR96, 5th Int. Conf. on Knowledge Representation and Reasoning*, pages 148–159. Morgan Kaufmann, 1996.

[8] Michael R. Dransfield, Lengning Liu, Victor W. Marek, and Miroslaw Truszczynski. Satisfiability and computing van der waerden numbers. *Electr. J. Comb.*, 11(1), 2004.

[9] Ronald L. Graham, Bruce L. Rothschild, and Joel H. Spencer. *Ramsey Theory*. A Wiley-Interscience publication. Wiley, 1990.

[10] Marijn J. H. Heule, Oliver Kullmann, and Victor W. Marek. Solving and verifying the boolean pythagorean triples problem via cube-and-conquer, 2016. Accepted for SAT 2016.

[11] Alexey Ignatiev, Alessandro Previti, and Joao Marques-Silva. Sat-based formula simplification. In Marijn Heule and Sean Weaver, editors, *Theory and Applications of Satisfiability Testing – SAT 2015*, volume 9340 of *Lecture Notes in Computer Science*, pages 287–298. Springer International Publishing, 2015.

[12] Avraham Itzhakov and Michael Codish. Breaking symmetries in graph search with canonizing sets, 2015. http://arxiv.org/abs/1511.08205.

[13] Tommi Junttila and Petteri Kaski. Engineering an efficient canonical labeling tool for large and sparse graphs. In *Proceedings of the Meeting on Algorithm Engineering & Expermiments*, pages 135–149, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[14] Boris Konev and Alexei Lisitsa. A sat attack on the erdos discrepancy conjecture. In Carsten Sinz and Uwe Egly, editors, *Theory and Applications of Satisfiability Testing SAT 2014*, volume 8561 of *Lecture Notes in Computer Science*, pages 219–226. Springer International Publishing, 2014.

[15] Michal Kouril and Jerome L. Paul. The van der waerden number $W(2, 6)$ is 1132. *Experimental Mathematics*, 17(1):53–61, 2008.

[16] Oliver Kullmann. Green-tao numbers and sat. In Ofer Strichman and Stefan Szeider, editors, *Theory and Applications of Satisfiability Testing SAT 2010*, volume 6175 of *Lecture Notes in Computer Science*, pages 352–362. Springer Berlin Heidelberg, 2010.

[17] Brendan D. McKay. *Practical Graph Isomorphism*. Technical report (Vanderbilt University. Department of Computer Science). Department of Computer Science, Vanderbilt University, 1981.

[18] Brendan D. McKay and Stanislaw P. Radziszowski. R(4, 5) = 25. *Journal of Graph Theory*, 19(3):309–322, 1995.

[19] Stanisław P. Radziszowski. Small Ramsey numbers. *The Electronic Journal of Combinatorics*, #DS1, 2014.

[20] Carsten Sinz. Towards an optimal CNF encoding of boolean cardinality constraints. In *Proc. of the 11th Intl. Conf. on Principles and Practice of Constraint Programming (CP 2005)*, pages 827–831, Sitges, Spain, October 2005.

[21] Marc Thurley. sharpsat: Counting models with advanced component caching and implicit bcp. In *Proceedings of the 9th International Conference on Theory and Applications of Satisfiability Testing*, SAT'06, pages 424–429, Berlin, Heidelberg, 2006. Springer-Verlag.

[22] Grigori S. Tseitin. On the complexity of derivation in propositional calculus. In J. Siekmann and G. Wrightson, editors, *Automation of Reasoning 2*, pages 466–483. Springer-Verlag, 1983.