Trimming Graphs Using Clausal Proof Optimization

Marijn J.H. Heule

Department of Computer Science, The University of Texas at Austin

Abstract. We present a method to gradually compute a smaller and smaller unsatisfiable core of a propositional formula by minimizing proofs of unsatisfiability. The goal is to compute a minimal unsatisfiable core that is relatively small compared to other minimal unsatisfiable cores of the same formula. We try to achieve this goal by postponing deletion of arbitrary clauses from the formula as long as possible—in contrast to existing minimal unsatisfiable core algorithms. We applied this method to reduce the smallest known unit-distance graph with chromatic number 5 from 553 vertices and 2 720 edges to 529 vertices and 2 670 edges.

1 Introduction

Today's satisfiability (SAT) solvers can not only determine whether a propositional formula can be satisfied, but they can also produce a certificate in case no satisfying assignments exists. These certificates, known as proofs of unsatisfiability, can be used for multiple purposes ranging from checking the correctness of the unsatisfiability claim [16,9,26,8,14] to computing interpolants [11]. In this paper, we focus on another application of proofs of unsatisfiability: computing an unsatisfiable core of the formula [15,27,1,3]. We observed that the size of proofs tends to correlate to the size the corresponding unsatisfiable cores: the smaller the proof, the smaller the unsatisfiable core. We present a method to exploit this relation by computing a smaller and smaller proof of unsatisfiability to compute a small unsatisfiable core. This method was developed to improve the upper bound of the smallest unit-distance graph with chromatic number 5, which is currently a Polymath project. Details about the problem and this project are described below. The presented method was developed as existing techniques performed poorly on this application. Yet it could help with other applications that use unsatisfiable cores too—which we plan to study in the near future.

The chromatic number of the plane, a problem first proposed by Nelson in 1950 [25], asks how many colors are needed to color all points of the plane such that no two points at distance 1 from each other have the same color. Early results showed that at least four and at most seven colors are required. By the de Bruijn–Erdős theorem, the chromatic number of the plane is the largest possible chromatic number of a finite unit-distance graph [4]. The Moser Spindle, a unit-distance graph with 7 vertices and 11 edges, shows the lower bound [24], while the upper bound is shown by a 7-coloring of the entire plane by Isbell [25].

In a breakthrough for this problem in April 2018, Aubrey de Grey improved the lower bound by providing a unit-distance graph with 1 581 vertices with chromatic number 5 [10]. This discovery by de Grey started a Polymath project to find smaller graphs. The current record is a graph with 553 vertices and 2 720 edges [13]. We present a new technique to construct a large unit-distance graph with chromatic number 5, which we reduce with the proposed method to a graph with "only" 529 vertices and 2 670 edges. This graph is much more symmetric compared to earlier small unit-distance graphs with chromatic number 5. The total costs to compute this graph were roughly 100 000 CPU hours.

2 Preliminaries

Propositional Formulas. We will minimize graphs on the propositional level. We consider formulas in *conjunctive normal form* (CNF), which are defined as follows. A *literal* is either a variable x (a *positive literal*) or the negation $\overline{x}$ of a variable x (a *negative literal*). The *complement* $\bar{l}$ of a literal l is defined as $\bar{l} = \overline{x}$ if $l = x$ and $\bar{l} = x$ if $l = \overline{x}$. For a literal l, $var(l)$ denotes the variable of l. A *clause* is a disjunction of literals and a *formula* is a conjunction of clauses.

An *assignment* is a function from a set of variables to the truth values 1 (*true*) and 0 (*false*). A literal l is *satisfied* by an assignment α if l is positive and $\alpha(var(l)) = 1$ or if it is negative and $\alpha(var(l)) = 0$. A literal is *falsified* by an assignment if its complement is satisfied by the assignment. A clause is satisfied by an assignment α if it contains a literal that is satisfied by α. A formula is satisfied by an assignment α if all its clauses are satisfied by α. A formula is *satisfiable* if there exists an assignment that satisfies it and *unsatisfiable* otherwise.

For a formula F and assignment α, we denote by $F|_\alpha$ a reduced copy of F without clauses satisfied by α and literals falsified by α. A *unit clause* is a clause with only one literal. The result of applying the *unit clause rule* to a formula F is the formula $F|l$ where (l) is a unit clause in F. The iterated application of this rule to a formula, until no unit clauses are left, is called *unit propagation*. If unit propagation yields the empty clause $\bot$, we say that it derived a *conflict*.

Clausal Proofs. A clause C is *redundant* with respect to a formula F if F and $F \wedge C$ are satisfiability equivalent. For instance, the clause $C = (x \vee y)$ is redundant with respect to the formula $F = (\overline{x} \vee \overline{y})$ since F and $F \wedge C$ are satisfiability equivalent (although they are not logically equivalent). This redundancy notion allows us to add redundant clauses to a formula while preserving satisfiability.

Given a formula $F = \{C_1, \ldots, C_m\}$, a *clausal derivation* of a clause C_n from F is a sequence $C_{m+1}, \ldots, C_n$ of clauses. Such a sequence gives rise to formulas $F_m, F_{m+1}, \ldots, F_n$, where $F_i = \{C_1, \ldots, C_i\}$. We call F_i the *accumulated formula* corresponding to the i-th proof step. A clausal derivation is *correct* if every clause C_i $(i > m)$ is redundant with respect to the formula F_{i-1} and if this redundancy can be checked in polynomial time with respect to the size of the proof. A clausal derivation is a *proof* of a formula F if it derives the unsatisfiable empty clause. Clearly, since every clause-addition step preserves satisfiability, and since the empty clause is always false, a proof of F certifies the unsatisfiability of F.

Checking the correctness of a clause C_i in a derivation consists of computing a justification why C_i is redundant with respect by formula F_{i-1}. The most commonly used method for this purpose is *reverse unit propagation* (RUP): Let α be the assignment that falsifies all literals in C_i. Clause C_i has the RUP property if and only if unit propagation on $F_{i-1}|\alpha$ results in a conflict. In this case the justification of C_i consists of all clauses that were required to derive the conflict. Clausal proofs that can be validated using this method are called RUP proofs. Most SAT-solving techniques can be compactly expressed as RUP.

Example 1. Consider the formula below consisting of 3 variables and 7 clauses:

$$F := (\overline{y} \vee z) \wedge (x \vee z) \wedge (\overline{x} \vee y) \wedge (\overline{x} \vee \overline{y}) \wedge (x \vee \overline{y}) \wedge (y \vee \overline{z}) \wedge (x \vee \overline{z})$$

A clausal proof of F is $\overline{y}, z, \bot$. A justification of this proof is shown in Fig. 1. This justification shows that $\overline{y}$ and z do not depend on each other. As a consequence, swapping them results in another correct proof. Notice that clause $(x \vee \overline{z})$ is not used in this justification and it is thus not part of the core of F.

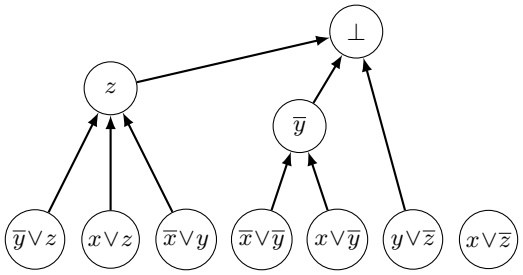

Fig. 1. A justification of the proof of the example formula. Each clause in the proof depends on its incoming arcs. The clauses without incoming arcs represent the formula.

In practice, clausal proofs also contain deletion information. The presence of deletion information significantly reduces the cost to compute a justification. Clausal proofs, which can be validated using the RUP method and include deletion information, are known as DRUP proofs. We mostly ignore the deletion information aspect of clausal proofs to simplify the presentation. All techniques discussed in this paper work with deletion information as well.

Chromatic Number of the Plane. The Chromatic Number of the Plane (CNP) [25] asks how many colors are required in a coloring of the plane to ensure that there exists no monochromatic pair of points with distance 1. A *unit-distance graph* is a graph formed from a set of points in the plane by connecting two points by an edge whenever the distance between the two points is exactly one. A lower bound for CNP of k colors can be obtained by showing that a unit-distance graph has chromatic number k.

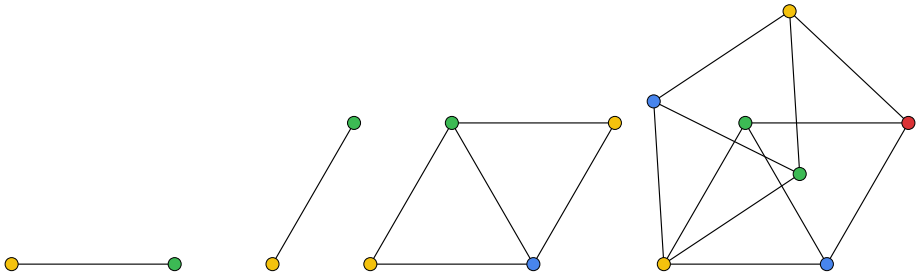

Fig. 2. From left to right: illustrations of unit-distance graphs A, B, $A \oplus B$, and the Moser Spindle. The graphs shown have chromatic number 2, 2, 3, and 4, respectively. The illustrations show valid colorings with the fewest number of colors.

We will use three operations to construct larger and larger graphs: the Minkowski sum [12], rotation, and merging. Given two sets of points A and B, the Minkowski sum of A and B, denoted by $A \oplus B$, equals $\{a + b \mid a \in A, b \in B\}$. Consider the sets of points $A = \{(0,0),(1,0)\}$ and $B = \{(0,0),(1/2,\sqrt{3}/2)\}$, then $A \oplus B = \{(0,0),(1,0),(1/2,\sqrt{3}/2),(3/2,\sqrt{3}/2)\}$.

Given a positive integer i, we denote by θ_i the rotation around point $(0,0)$ with angle $\arccos(\frac{2i-1}{2i})$ and by θ_i^k the application of θ_i k times. Let p be a point with distance $\sqrt{i}$ from $(0,0)$, then the points p and $\theta_i(p)$ are exactly distance 1 apart and thus would be connected with an edge in a unit-distance graph. Consider again the set of points $A \oplus B$ above. The points $A \oplus B \cup \theta_3(A \oplus B)$ form the Moser Spindle [24] with chromatic number 4. Figure 2 shows visualizations of these sets with connected vertices colored differently.

3 Overview of the Approach

The smallest known unit-distance graph with chromatic number 5 has 553 vertices and 2 720 edges [13]. This graph was found using the following method. Start with a large unit-distance graph G with chromatic number 5. Now reduce the size of that graph by solving the formula that encodes whether the graph can be colored with 4 colors. That formula is unsatisfiable. From the proof of unsatisfiability, an unsatisfiable core can be extracted that represents a subgraph with chromatic number 5. This step is repeated again and again as long as the graph is reduced. In the last step, vertices are randomly eliminated to make the graph vertex-critical: removing any additional vertex introduces a 4-coloring.

In this paper we present two improvements. The most important one is a new method, presented in Section 4, to produce short proofs of unsatisfiability. We observed that for the formulas studied in this paper that the shorter the proof, the smaller the unsatisfiable core and thus the smaller the subgraph. The second improvement is the construction of a new large unit-distance graph G that we use as a starting point to find a smaller unit-distance graph with chromatic number 5. The construction of this graph is explained in Section 5. These two methods allowed us to find a unit-distance graph with 529 vertices and 2 670 edges. This graph is much more symmetric compared to the graph with 553 vertices.

4 Clausal Proof Optimization

Most SAT solvers can emit a clausal proof of unsatisfiability. There exist several checkers for such proofs, including formally-verified ones [7,19]. We extended the checker DRAT-trim [15] that allows optimizing the clausal proof as well as extracting an unsatisfiable core. One can obtain multiple unsatisfiable cores from a single clausal proof—in contrast to a resolution proof [28]. The existing method works via backward checking [9]: Given a proof of unsatisfiability, the last clause (the empty clause) of the proof is marked. Now the proof is validated in reverse order. For each marked clause it is determined which clauses (occurring earlier in the proof or in the formula) are required for the validation. Those clauses will be marked (if they were not marked already). The order in which unit propagation is applied influences which clauses become marked. Unmarked clauses are not validated. After the proof is verified, the marked clauses in the formula form an unsatisfiable core and the marked clauses in the proof form an optimized proof. We present two new extensions that further reduce the size of the formula.

4.1 Justification Order Shuffling

A clausal proof typically has many different justifications and a justification can typically be converted into many different clausal proofs, i.e., clauses appear in a different order in the sequence. Here we exploit this property by 1) computing a justification for a given clausal proof, 2) removing the clauses that are redundant based on that justification, and 3) shuffle the remaining clauses in the proof based on that justification. These steps are repeated multiple times.

Figure 3 shows the pseudo code of that algorithm. The procedure RemoveRedundancy removes from a given clausal proof P and justification J all the clauses in P that do not occur in any of the justifications of J. Given a justification J, the procedure ShuffleProof produces a random permutation of the clauses in J such that each clause C appears 1) later in the proof than all the clauses in the justification of C and 2) before all clauses that list C in their justification. Additionally, ShuffleProof randomly shuffles the literals of each clause in J.

Clause deletion is not mentioned in the algorithm, but can also be helpful to optimize proofs. A clause C can be deleted in a proof as soon as none of the clauses occurring later in the proof uses C in their justification. On the other hand, one could delete C at a later point in the proof (or not at all) to

```
   OptimizeProof (clausal proof P, formula F)
1    do
2      J := ComputeJustification (P, F)
3      J := RemoveRedundancy (J)
4      P := ShuffleProof (J)
5    while (progress)
6    return P
```

Fig. 3. Optimizing a proof by iterative computing a new justification.

allow clauses later in the proof to incorporate C in their justification in the next iteration. We randomly postpone deleting clauses in proofs in a certain window. The window is slightly increased in each next iteration.

4.2 Iterative Trimming the Formula

Given a unsatisfiable formula that encodes the existence of a k-coloring of a graph, an unsatisfiable core of that formula represents a subgraph that cannot be colored with k colors. To find a small subgraph we would like a minimal unsatisfiable core and ideally the smallest minimal unsatisfiable core. Although there has been some research in to the latter [17,20,23], it is already hard to compute a minimal unsatisfiable core. Existing algorithms for computing a minimal unsatisfiable core [21,22] focus more on easy problems. For harder problems it is required to trim the formulas using a preprocessing step [3].

In preliminary experiments we observed that existing algorithms got stuck. It turned out that if a "wrong" vertex is removed from the graph, then proving that the remaining graph still has chromatic number 5 is very expensive. A proof that the initial graph has chromatic number 5 consists of roughly 10,000 clauses. After removing a clause that represents a "wrong" vertex, the proof consists of millions of clauses. We concluded that existing tools are not effective for this application, because they remove clauses arbitrary. This will eventually result in removing a clause representing a "wrong" vertex. Although the checking costs are a serious problem, there is a more problematic issue: as soon as it requires millions of clauses to prove that the graph has chromatic number five, then many vertices are involved in the proof and the minimal unsatisfiable core will be relatively large. As a consequence, this also holds for the graph represented by this core. We address this issue by taking away the elimination of arbitrary clauses. Instead, we only remove clauses via trimming and proof optimization.

Figure 4 shows the pseudo codes of two algorithms to trim a formula: one algorithm, called TrimFormulaPlain, that simply adds proof optimization to the trimming loop and another one, called TrimFormulaInteract, that additionally interacts with the original formula to further optimize the proof. We focus on the latter algorithm, which is one of the main contributions of this paper.

```
   TrimFormulaPlain (formula F)
1    F_core := F
2    do
3      P := ComputeProof (F_core)
4      P := OptimizeProof (P, F_core)
5      F_core := ComputeCore (P, F_core)
6    while (progress)
7    return F_core
```

```
   TrimFormulaInteract (formula F)
1    F_core := F
2    do
3      P := ComputeProof (F_core)
4      P := OptimizeProof (P, F_core)
5      P := OptimizeProof (P, F)
6      F_core := ComputeCore (P, F)
7    while (progress)
8    return F_core
```

Fig. 4. Pseudo code of two algorithms to trim the size of a formula using proof optimization: TrimFormulaPlain and TrimFormulaInteract. The latter algorithm interacts with the original formula to further optimize the proof.

Algorithm TrimFormulaInteract takes advantage of the following property of (D)RUP proofs: If (D)RUP proof P is a correct proof of formula F, then P is a correct proof of any formula F' such that $F' \supseteq F$. Observe that additional clauses cannot break the RUP check: if unit propagation on F results in a conflict, then unit propagation on F' results in a conflict.

In each step of the main loop of TrimFormulaInteract, we first compute a proof of unsatisfiability of the trimmed formula F_{core}. The size of this proof is crucial for the quality of the trimming. One could therefore solve F_{core} multiple times by shuffling the clauses and select the smallest proof of these runs. Afterwards, this proof is optimized using F_{core} via the algorithm shown in Fig. 3. Next, we use the property discussed above and further optimize the proof using F and the same optimization algorithm. The algorithm has now more options to minimize the proof as $F \supseteq F_{\text{core}}$. Moreover, the algorithm allows for a novel way to compute a smaller core: In an earlier step a clause may have been removed that allows for a small proof of unsatisfiability and/or small unsatisfiable core. Since each step considers again all clauses of F, that clause may be pulled back into F_{core}.

The size of F_{core} does not necessarily decrease with each iteration and may actually increase if a low quality proof is computed in line 3. We repeat the algorithm as long as there is progress. In this case, we measured progress by the reduction of the size of F_{core}.

The result of these trimming algorithms is rarely a minimal unsatisfiable core of the formula. We applied the classical destructive method [5] to reduce F_{core} to a minimal unsatisfiable core. We observed (some details are presented in Section 6.2) that the size of the minimal unsatisfiable core can vary significantly based on the selection of the clauses to remove. As a consequence we ran this method multiple (thousands of) times on the cluster to obtain a relatively small minimal unsatisfiable core of F_{core}.

5 Observed patterns of points in $\mathbb{Q}[\sqrt{3}, \sqrt{11}] \times \mathbb{Q}[\sqrt{3}, \sqrt{11}]$

The smallest known unit-distance graph with chromatic number 5, called G_{553}, has 553 vertices [13]. Its key component is a set of 420 points embedded in $\mathbb{Q}[\sqrt{3}, \sqrt{11}] \times \mathbb{Q}[\sqrt{3}, \sqrt{11}]$ that have a limited number (19) of the colorings of the points at distance 2 from the origin (central vertex) when coloring the set with 4 colors. Our strategy to compute a small unit-distance graph with chromatic number 5 is finding a small set of vertices with the same property. We explored many large graphs with points in $\mathbb{Q}[\sqrt{3}, \sqrt{11}] \times \mathbb{Q}[\sqrt{3}, \sqrt{11}]$ and computed the size of proofs of unsatisfiability of the formula that determines the existence of a 4-coloring while blocking the limited number of the colorings of the points at distance 2. This section describes how we obtained the large graph with the smallest proof of unsatisfiability that we encountered.

We denote by H_R the graph consisting of i) a regular hexagon with maximal radius R and ii) its center. The points of H_R in the plane are $(0, 0)$, $(R, 0)$, $(R/2, R\sqrt{3}/2)$, $(-R/2, R\sqrt{3}/2)$, $(-R, 0)$, $(-R/2, -R\sqrt{3}/2)$ and $(R/2, -R\sqrt{3}/2)$. Furthermore, we denote by H'_R a copy of H_R rotated by 90 degrees.

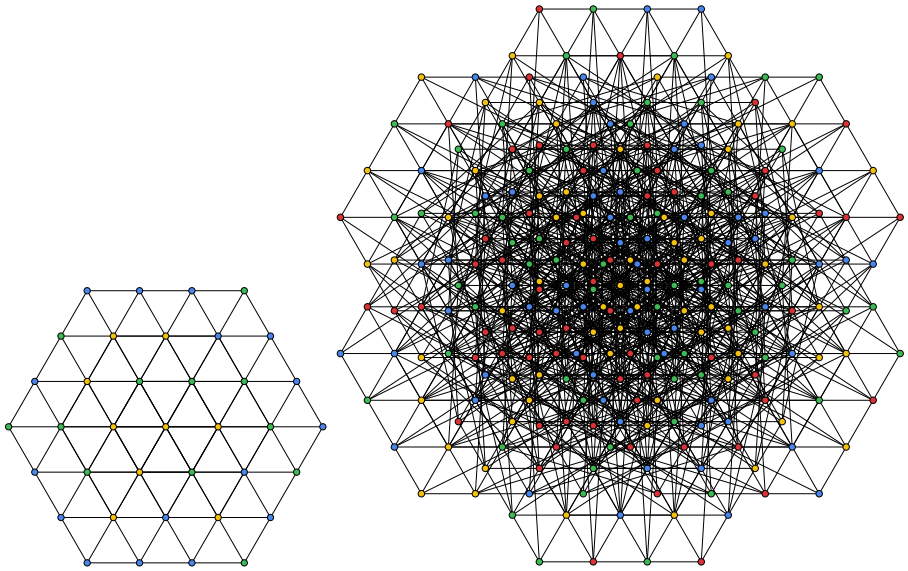

Fig. 5. A 3-coloring of the graph $H_{\frac{1}{3}} \oplus H_{\frac{1}{3}} \oplus H_{\frac{1}{3}}$ (left) and a 4-coloring of the graph $H_{\frac{1}{3}} \oplus H_{\frac{1}{3}} \oplus H_{\frac{1}{3}} \oplus H'_{\frac{\sqrt{3}+\sqrt{11}}{6}}$ (right).

We observed some interesting patterns when combining the graphs $H_{\frac{1}{3}}$ and $H'_{\frac{\sqrt{3}+\sqrt{11}}{6}}$. Figure 5 (left) shows the graph $H_{\frac{1}{3}} \oplus H_{\frac{1}{3}} \oplus H_{\frac{1}{3}}$, which is a triangular grid with diameter 1. This graph has 37 vertices and 48 edges and can be colored with 3 colors. However, the Minkowski sum of this triangular grid and $H'_{\frac{\sqrt{3}+\sqrt{11}}{6}}$, shown in Fig. 5 (right), is not 3-colorable. Notice that there are many edges between the seven triangular grids. Actually, the graph has 259 vertices and 1 056 edges and most of these edges (720) are between triangular grids. There exist many 4-colorings of this graph and most of them have no observable pattern.

Patterns start to emerge when applying the Minkowski sum again. Figure 6 shows a 4-coloring of the resulting graph $H_{\frac{1}{3}} \oplus H_{\frac{1}{3}} \oplus H_{\frac{1}{3}} \oplus H'_{\frac{\sqrt{3}+\sqrt{11}}{6}} \oplus H'_{\frac{\sqrt{3}+\sqrt{11}}{6}}$. Observe the clustering of vertices with the same color in circles of roughly a diameter of 1 in size. This pattern can be observed in many of the found 4-colorings of this graph, although there also exist some 4-colorings without this pattern. It appears that assigning the same color to nearby vertices is the easiest way to color this graph (using a SAT solver).

Applying the Minkowski sum another time breaks the prior pattern completely, as no more 4-colorings exist with clusters of vertices having the same color. However, new patterns emerge, as can be seen in Fig 7. For example, notice the reflection in the central vertical axis of the blue and green vertices.

Based on these observations, we experimented with ways to combine $H_{\frac{1}{3}}$ and $H'_{\frac{\sqrt{3}+\sqrt{11}}{6}}$. An effective combination turned out to be unit-distance graph G_{2167}. This graph is constructed as follows. Let C_{13} denote the union of $H_{\frac{1}{3}}$ and

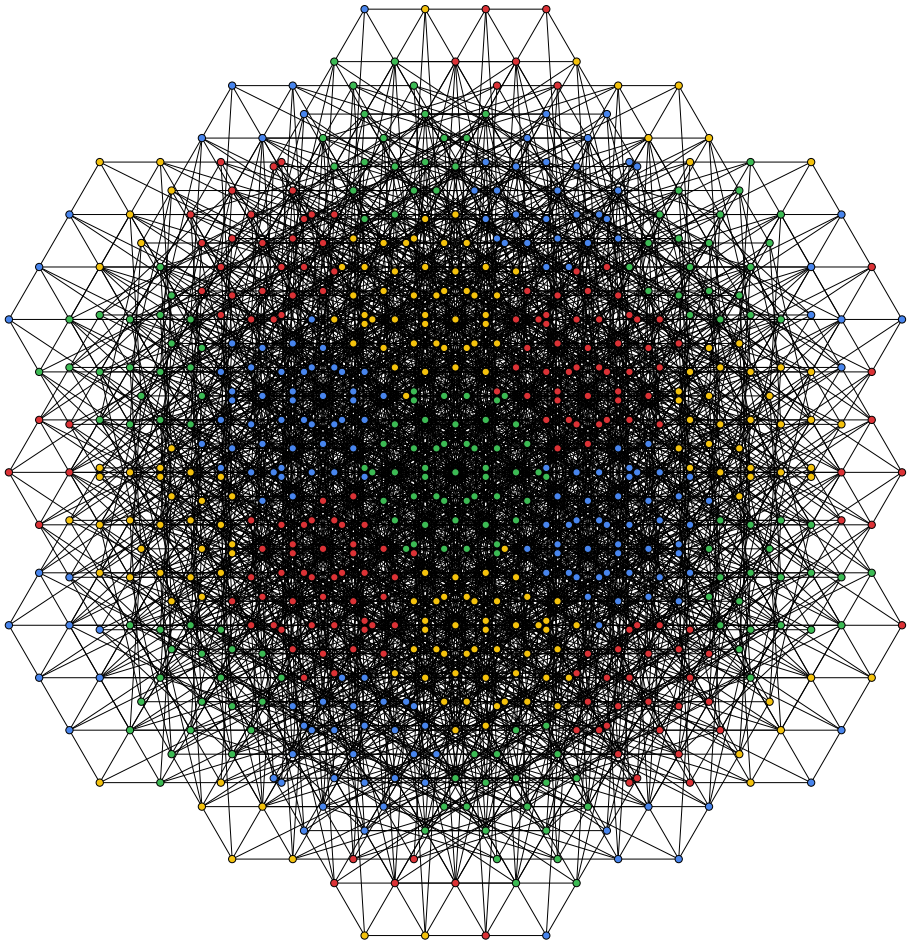

Fig. 6. A 4-coloring of the graph $H_{\frac{1}{3}} \oplus H_{\frac{1}{3}} \oplus H_{\frac{1}{3}} \oplus H'_{\frac{\sqrt{3}+\sqrt{11}}{6}} \oplus H'_{\frac{\sqrt{3}+\sqrt{11}}{6}}$.

$H'_{\frac{\sqrt{3}+\sqrt{11}}{6}}$. Now G_{2167} equals $C_{13} \oplus C_{13} \oplus C_{13} \oplus C_{13} \oplus C_{13} \oplus C_{13} \oplus C_{13}$ without the points that have a distance larger than 2 from the central vertex. This graph has 2 167 vertices and 16 512 edges and is shown in Fig. 8. Notice that the average vertex degree is larger than 15. This is quite high for a graph with chromatic number 4.

Observe the vertical monochromatic lines in Fig. 8: Points with the same horizontal coordinate have the same color. This pattern appears in many 4-colorings (modulo a rotation of 60 degrees). There are solutions with vertical lines with two colors, but none of the 4-colorings have more colors on a single vertical line (again, modulo a rotation of 60 degrees). The only reason why such solutions can exist is that the construction of G_{2167} does not generate points with distance 1 that have the same horizontal coordinate. There appears no

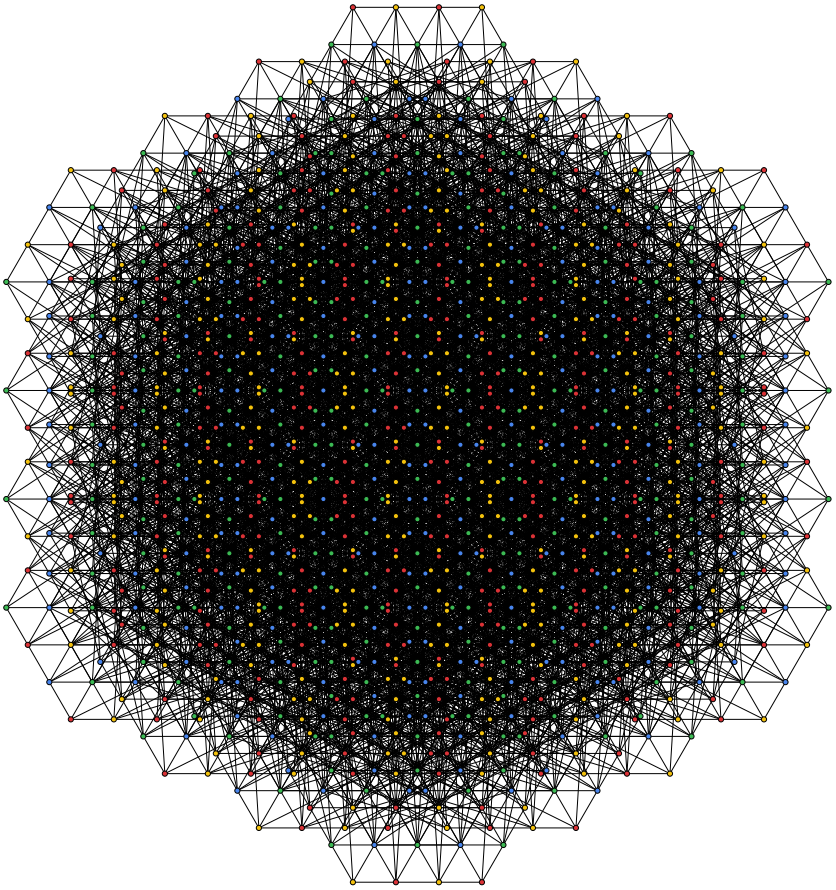

Fig. 7. A 4-coloring of the graph $H_{\frac{1}{3}} \oplus H_{\frac{1}{3}} \oplus H_{\frac{1}{3}} \oplus H'_{\frac{\sqrt{3}+\sqrt{11}}{6}} \oplus H'_{\frac{\sqrt{3}+\sqrt{11}}{6}} \oplus H'_{\frac{\sqrt{3}+\sqrt{11}}{6}}$.

obvious way to add such points in a way that the resulting graph has chromatic number 5. Another pattern that can be observed in Fig. 8 is that points with the same vertical coordinate that are 2/3 apart from each other also have the same color. Also any two points that are 1/3 apart have a different color. For example, forcing that any vertex at distance 1/3 from the origin has the same color as the central vertex eliminates all 4-colorings. Hence 1/3 is a so-called virtual-edge in 4-colorings of unit-distance graphs.

6 Small Unit-Distance Graph with Chromatic Number 5

In this section we present our SAT-based approach to improve the smallest known unit-distance graph with chromatic number 5. We first explain how we encode the problem and afterwards apply the new trimming algorithm presented in Section 4.2.

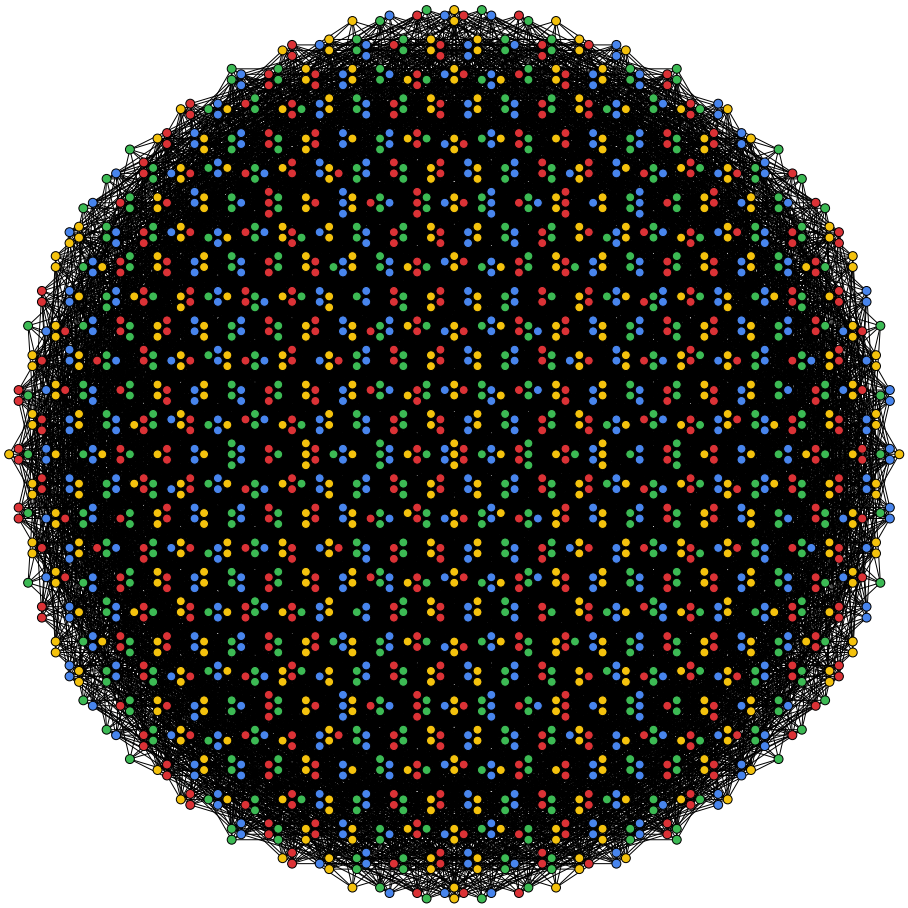

Fig. 8. A 4-coloring of graph G_{2167}.

6.1 Encoding

We can compute the chromatic number of a graph G as follows. Construct two formulas, one asking whether G can be colored with $k-1$ colors, and one whether G can be colored with k colors. Now, G has chromatic number k if and only if the former is unsatisfiable while the latter is satisfiable.

The construction of these two formulas can be achieved using the following encoding [13]: Given a graph $G = (V, E)$ and a parameter k, the encoding uses $k|V|$ boolean variables $x_{v,c}$ with $v \in V$ and $c \in \{1, \ldots, k\}$. These variables have the following meaning: $x_{v,c}$ is true if and only if vertex v has color c. Now we can construct a propositional formula F_k that is satisfiable if and only if G can be colored with k colors:

$$F_k := \bigwedge_{v \in V} (x_{v,1} \vee \cdots \vee x_{v,k}) \wedge \bigwedge_{\{v,w\} \in E} \bigwedge_{c \in \{1,\ldots,k\}} (\overline{x}_{v,c} \vee \overline{x}_{w,c})$$

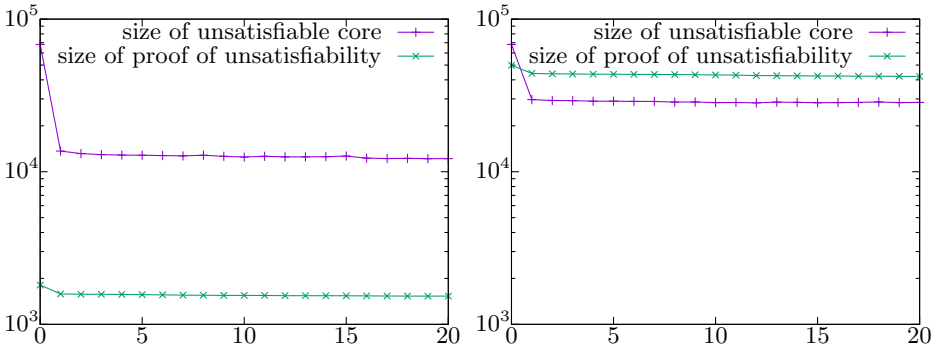

Fig. 9. The size (number of clauses) of the unsatisfiable core and the optimized proof of unsatisfiability (y-axis) of the first twenty steps (x-axis) of the OptimizeProof algorithm, when starting with F_4^+ and the smallest proof (left) or the largest proof (right).

The first type of clauses, called vertex clauses, ensures that each vertex has at least one color, while the second type of clauses, called edge clauses, forces that two connected vertices are colored differently. Additionally, we could include clauses to require that each vertex has at most one color. However, these clauses are redundant and would be eliminated by blocked clause elimination [18], a SAT preprocessing technique. We experimented using formulas with and without blocked clauses. Although the results were quite similar, we had the impression that without blocked clauses is slightly better.

We added symmetry-breaking predicates [6] during all experiments to speed up solving and proof minimization. The color symmetries were broken by fixing the vertex at $(0, 0)$ to the first color, the vertex at $(1, 0)$ to the second color, and the vertex at $(1/2, \sqrt{3}/2)$ to the third color. These three points are at distance 1 from each other and occurred in all our graphs. The speedup is roughly a factor of 24 $(= 4 \cdot 3 \cdot 2)$, when proving the absence of a 4-coloring.

6.2 Reducing the Large Part

The smallest known unit-distance graph with chromatic number 5 has 553 vertices and consists of two parts: a large part with 420 vertices and a small part with 134 vertices. The large part and small part have one vertex in common: the origin. Analysis of these parts [13] showed that they have different purposes: the large part limits the number of valid 4-coloring of 12 vertices at distance 2 from the origin to 19. The small part prevents these 12 vertices to having any of these 19 4-colorings. Some important details are missing from this analysis and they will be discussed later. We focused our effort to search for a small unit-distance graph with chromatic number 5 by looking for a more compact large part.

In the first step, we constructed the formula whether graph G_{2167} has a 4-coloring. Apart from the symmetry-breaking predicates, we added 19 clauses that block the above mentioned 4-colorings that remain in the large part. This

formula, called F_4^+, is unsatisfiable and has 8 668 variables and 68 237 clauses. In the next step we produce a proof of unsatisfiability of this formula. We used the SAT solver glucose 3.0 [2] (without preprocessing techniques) for this purpose. This solver allows to randomly initialize the decision heuristics (VSIDS), which is a feature that can easily be added to most SAT solvers. This initialization can have a significant impact on the size of the proof *and* on the size of the core. For example, we solved the formula with 100 different seeds for the initialization. The smallest proof had 1 809 clause addition steps, while the largest proof had 49 838 clause addition steps. The default glucose 3.0, i.e., without decision heuristics initialization, produced a proof with 2 475 clause addition steps.

Figure 9 shows the effect of using the smallest and largest proof as input for the OptimizeProof algorithm, which has been implemented in the DRAT-trim proof checker (available at `https://github.com/marijnheule/drat-trim`) [15]. In both cases the size of the proof reduction is modest. However, a much smaller unsatisfiable core can be extracted from the optimized smallest proof compared to the optimized largest proof. The smaller core also corresponds to a smaller subgraph (963 versus 1 609 vertices).

We also experimented with the two algorithms presented in Section 4.2. Figure 10 shows the size of the subgraph (extracted from the core) for the first 20 iterations with formula F_4^+ as input using TrimFormulaPlain (left) or TrimFormulaInteract (right). Each experiment was run five times. The figure shows that TrimFormulaInteract produces significantly smaller subgraphs. The TrimFormulaPlain algorithm, as shown in Fig. 4, actually performs significantly worse than the performance presented in Fig. 10. This poor performance is caused by the removal of edge clauses and symmetry-breaking predicates from the core. We improved the TrimFormulaPlain algorithm by adding back the removed edge clauses and symmetry-breaking predicates in each iteration.

We studied the resulting graphs and observed that they were close to symmetric: Taking the union of the graph with rotated copies (120 degrees rotation in the origin) added only a few dozen vertices. We decided to check whether

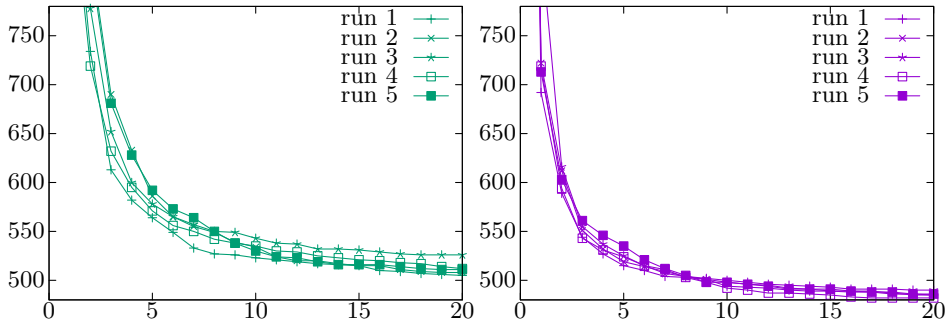

Fig. 10. The size of subgraphs corresponding to the unsatisfiable cores when using the algorithms TrimFormulaPlain (left) and TrimFormulaInteract (right).

this observation could be used to further shrink the large part by taking this union as initial graph (instead of G_{2167}) and rerun the procedure. This turned out to be effective and allowed removing some additional vertices. We ran the entire experiment many times on the Lonestar 5 cluster of the Texas Advanced Computing Center. Several runs resulted in a graphs with "only" 393 vertices. These graphs turned out to be the same (modulo rotation and reflection). We call this graph L_{393}. One can make L_{393} symmetric, i.e., it maps onto itself when rotating it by 120 degrees along the central vertex, by adding a single vertex.

6.3 Finalizing the Graph

The graph L_{393}, produced in the previous subsection, needs to be extended with a "small part" to establish a unit-distance graph with chromatic number 5. Initially we tried to use the small part of G_{553}. However, the resulting graph is 4-colorable, because L_{393} has fewer connections with that small part compared to the large part of G_{553}. We fixed this as follows: The small part of G_{553} got expanded by merging it with copies that are 60 degrees rotated in the origin. This resulted in a graph with 181 vertices (while the small part of G_{553} has 134 vertices), which we call S_{181}. The union of L_{393} and S_{181} has chromatic number 5. We applied the same techniques as described in the previous subsection to further reduce the size of this graph. This resulted in a graph being the union of L_{393} and a new small part with 137 vertices.

Figure 11 shows the final graph G_{529} consisting of 529 vertices and 2 670 edges. This graph almost maps onto itself when rotating it with 120 degrees in the origin. The figure shows a coloring in which only the origin has the fifth color (white). Such a coloring exists for each vertex as the graph is vertex critical. The shown coloring is a randomly selected one. Observe the clustering of vertices with the same color. This pattern looks similar to the one shown in Fig. 6.

Graph G_{529} is available at `https://github.com/marijnheule/CNP-SAT` as a list of points in the plane and a list of unit-distance edges. The repository also contains a CNF formula encoding whether G_{529} is 4-colorable and a proof of unsatisfiability that can be validated in a few seconds.

7 Conclusions

We presented a new algorithm to trim a formula by first optimizing a proof of unsatisfiability. The algorithm optimizes the proof using both the shrinking formula and the original formula. This allows reintroducing clauses in the shrinking formula, which could further improve the trimming.

We constructed a unit-distance graph with points in $\mathbb{Q}[\sqrt{3}, \sqrt{11}]\times\mathbb{Q}[\sqrt{3}, \sqrt{11}]$. The 4-colorings of this graph, G_{2167}, have some interesting properties such as 1) many (and in some 4-colorings all) vertices with the same horizontal coordinate have the same color; 2) vertices that are 1/3 apart having a different color; and 3) vertices with the same vertical coordinate that are 2/3 apart have the same color. All these properties are for a rotation of G_{2167} by 0, 120, or 240 degrees.

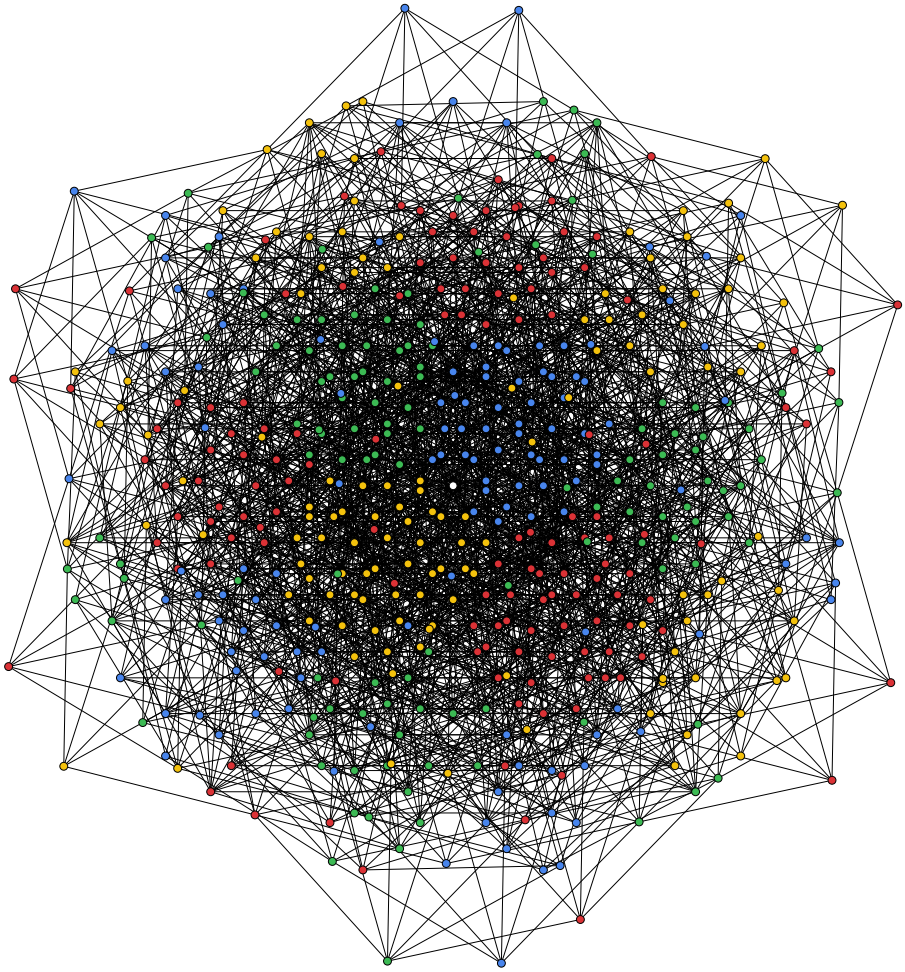

Fig. 11. A 529-vertex unit-distance graph with chromatic number 5. In the shown coloring, only the origin has the fifth color (white).

By combining the new algorithm and the new graph, we were able to reduce the smallest known unit-distance graph with chromatic number 5 to a graph with 529 vertices and 2670 edges (down from 553 vertices and 2720 edges). This graph is also much more symmetric. It is generally easier to understand why a symmetric object has a certain property compared to an asymmetric object. It may thus provide some insight how to obtain a unit-distance graph with chromatic number 6 (if they exist). Using the techniques in the paper we constructed several graphs with up to 100 000 vertices, but all were 5-colorable.

As future work, we plan to study the effectiveness of the new algorithm on other applications that require minimal unsatisfiable cores.

Acknowledgements

The author is supported by the National Science Foundation (NSF) under grant CCF-1813993. The author acknowledges the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC resources that have contributed to the research results reported within this paper.

References

1. Asín, R., Nieuwenhuis, R., Oliveras, A., Rodríguez-Carbonell, E.: Efficient generation of unsatisfiability proofs and cores in sat. In: Cervesato, I., Veith, H., Voronkov, A. (eds.) LPAR. LNCS, vol. 5330, pp. 16–30. Springer (2008)
2. Audemard, G., Simon, L.: Predicting learnt clauses quality in modern SAT solvers. In: 21st International Joint Conference on Artificial Intelligence. pp. 399–404 (2009)
3. Belov, A., Heule, M., Marques-Silva, J.: MUS extraction using clausal proofs. In: Sinz, C., Egly, U. (eds.) Theory and Applications of Satisfiability Testing - SAT 2014. Lecture Notes in Computer Science, vol. 8561, pp. 48–57. Springer (2014)
4. de Bruijn, N.G., Erdős, P.: A colour problem for infinite graphs and a problem in the theory of relations. Nederl. Akad. Wetensch. Proc. Ser. A 54, 371–373 (1951)
5. Chinneck, J.W., Dravnieks, E.W.: Locating minimal infeasible constraint sets in linear programs. INFORMS Journal on Computing 3, 157–168 (1991)
6. Crawford, J.M., L., G.M., Luks, E.M., Roy, A.: Symmetry-breaking predicates for search problems. In: Knowledge Representation and Reasoning – KR 1996. pp. 148–159. Morgan Kaufmann (1996)
7. Cruz-Filipe, L., Heule, M.J.H., Hunt Jr., W.A., Kaufmann, M., Schneider-Kamp, P.: Efficient certified RAT verification. In: Automated Deduction – CADE 26. pp. 220–236. Springer (2017)
8. Cruz-Filipe, L., Marques-Silva, J.P., Schneider-Kamp, P.: Efficient certified resolution proof checking. In: TACAS 2017. Lecture Notes in Computer Science, vol. 10205, pp. 118–135 (2017)
9. Goldberg, E.I., Novikov, Y.: Verification of proofs of unsatisfiability for CNF formulas. In: DATE. pp. 10886–10891 (2003)
10. de Grey, A.D.N.J.: The chromatic number of the plane is at least 5. Geombinatorics XXVIII, 18–31 (2018)
11. Gurfinkel, A., Vizel, Y.: DRUPing for interpolants. In: Proceedings of the 14th Conference on Formal Methods in Computer-Aided Design. pp. 19:99–19:106. FMCAD '14, FMCAD Inc, Austin, TX (2014)
12. Hadwiger, H.: Minkowskische addition und subtraktion beliebiger punktmengen und die theoreme von erhard schmidt. Mathematische Zeitschrift 53(3), 210–218 (Jun 1950)
13. Heule, M.J.H.: Computing small unit-distance graphs with chromatic number 5. Geombinatorics XXVIII, 32–50 (2018)
14. Heule, M.J.H., Hunt Jr., W.A., Kaufmann, M., Wetzler, N.D.: Efficient, verified checking of propositional proofs. In: Ayala-Rincón, M., Muñoz, C.A. (eds.) Interactive Theorem Proving: 8th International Conference, ITP 2017, Brasília, Brazil, September 26–29, 2017, Proceedings. pp. 269–284. Springer (2017)
15. Heule, M.J.H., Hunt, Jr., W.A., Wetzler, N.D.: Trimming while checking clausal proofs. In: Formal Methods in Computer-Aided Design. pp. 181–188. IEEE (2013)

16. Heule, M.J.H., Hunt, Jr., W.A., Wetzler, N.D.: Bridging the gap between easy generation and efficient verification of unsatisfiability proofs. Software Testing, Verification, and Reliability (STVR) 24(8), 593–607 (2014)
17. Ignatiev, A., Previti, A., Liffiton, M., Marques-Silva, J.: Smallest mus extraction with minimal hitting set dualization. In: Pesant, G. (ed.) Principles and Practice of Constraint Programming. pp. 173–182. Springer International Publishing, Cham (2015)
18. Järvisalo, M., Biere, A., Heule, M.J.H.: Simulating circuit-level simplifications on cnf. Journal of Automated Reasoning 49(4), 583–619 (2012)
19. Lammich, P.: Efficient verified (UN)SAT certificate checking. In: Automated Deduction – CADE 26. pp. 237–254. Springer (2017)
20. Liffiton, M., Mneimneh, M., Lynce, I., Andraus, Z., Marques-Silva, J., Sakallah, K.: A branch and bound algorithm for extracting smallest minimal unsatisfiable subformulas. Constraints 14(4), 415 (Aug 2008)
21. Lynce, I., Marques Silva, J.P.: On computing minimum unsatisfiable cores. In: SAT 2004 - The Seventh International Conference on Theory and Applications of Satisfiability Testing, 10-13 May 2004, Vancouver, BC, Canada, Online Proceedings (2004)
22. Marques-Silva, J., Lynce, I.: On improving mus extraction algorithms. In: Proceedings of the 14th International Conference on Theory and Application of Satisfiability Testing. pp. 159–173. SAT'11, Springer-Verlag, Berlin, Heidelberg (2011)
23. Mneimneh, M., Lynce, I., Andraus, Z., Marques-Silva, J., Sakallah, K.: A branch-and-bound algorithm for extracting smallest minimal unsatisfiable formulas. In: Bacchus, F., Walsh, T. (eds.) Theory and Applications of Satisfiability Testing. pp. 467–474. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
24. Moser, L., Moser, W.: Solution to problem 10. Can. Math. Bull. 4, 187–189 (1961)
25. Soifer, A.: The Mathematical Coloring Book (2008), ISBN-13: 978-0387746401
26. Van Gelder, A.: Verifying RUP proofs of propositional unsatisfiability. In: ISAIM (2008)
27. Wetzler, N.D., Heule, M.J.H., Hunt Jr., W.A.: DRAT-trim: Efficient checking and trimming using expressive clausal proofs. In: Theory and Applications of Satisfiability Testing – SAT 2014. pp. 422–429. Springer International Publishing, Cham (2014)
28. Zhang, L.: Searching for Truth: Techniques for Satisfiability of Boolean Formulas. Ph.D. thesis, Princeton University, Princeton, NJ, USA (2003)