Co-Certificate Learning with SAT Modulo Symmetries

Markus Kirchweger , **Tomáš Peitl** and **Stefan Szeider**
Algorithms and Complexity Group, TU Wien, Austria
{mk, peitl, sz}@ac.tuwien.ac.at

Abstract

We present a new SAT-based method for generating all graphs up to isomorphism that satisfy a given co-NP property. Our method extends the SAT Modulo Symmetry (SMS) framework with a technique that we call co-certificate learning. If SMS generates a candidate graph that violates the given co-NP property, we obtain a certificate for this violation, i.e., 'co-certificate' for the co-NP property. The co-certificate gives rise to a clause that the SAT solver, serving as SMS's backend, learns as part of its CDCL procedure. We demonstrate that SMS plus co-certificate learning is a powerful method that allows us to improve the best-known lower bound on the size of Kochen-Specker vector systems, a problem that is central to the foundations of quantum mechanics and has been studied for over half a century. Our approach is orders of magnitude faster and scales significantly better than a recently proposed SAT-based method.

1 Introduction

SAT modulo symmetries (SMS) is a recently proposed framework that brings efficient symmetry breaking to conflict-driven (CDCL) SAT solvers and has achieved state-of-the-art results on several symmetry-rich combinatorial search problems, enumerating or proving the non-existence of graphs, planar graphs, directed graphs, and matroids with particular properties [Kirchweger and Szeider, 2021; Kirchweger *et al.*, 2022; Kirchweger *et al.*, 2023b]. In this paper, we propose to extend SMS to a class of problems that do not admit a succinct SAT encoding because they involve quantifier alternation: where we are asked to find a combinatorial object (the existential part) that has some co-NP-complete property (the universal part, stated as 'all candidate polynomial-size witnesses fail'). We call such problems *alternating search* problems; a simple concrete example of an alternating search problem is the well-studied question posed by Erdős [1967], of finding a smallest triangle-free graph that is not properly k-colorable, for a fixed $k \geq 3$. Encoding the non-k-colorability property for $k \geq 3$ into a family of polynomially sized propositional formulas is impossible unless NP $=$ co-NP, since checking k-colorability is NP-complete [Karp, 1972].

SMS has some advantages over alternative methods such as isomorphism-free exhaustive enumeration by canonical construction path [McKay, 1998] as implemented in tools like Nauty [McKay and Piperno, 2014], or different symmetry-breaking methods for SAT. The former can very efficiently generate all objects of a given order, but is very difficult to integrate with complex constraints and learning, while the latter is either intractable (full 'static' symmetry breaking [Codish *et al.*, 2016; Itzhakov and Codish, 2015], which requires constraints of exponential size), or ineffective (partial static symmetry breaking [Codish *et al.*, 2019]). Because SMS strikes a better balance than these other methods, with both native constraint reasoning and learning as well as effective and efficient symmetry breaking, we chose it as our basis for alternating search.

We call our new method SMS plus *co-certificate learning (CCL)*, and it works as follows. We run an SMS solver on an encoding of the existential part of our alternating search problem, giving us a model that corresponds to a *solution candidate*. In the context of our running example, this candidate would be a triangle-free graph. Next, we test the co-NP property (non-k-colorability). If the graph is not colorable, we have found a solution (and we may or may not proceed to enumerate all solutions as with any other SAT encoding). If it is colorable, we find a coloring, which is a *co-certificate* of the graph *not* having our desired non-colorability property. The co-certificate gives rise to a clause that is learned by the SMS solver and prevents any solutions with the same co-certificate (colorable by the same coloring), and search resumes. We describe the method in full detail in Section 3—for now suffice it to say that we essentially apply SMS incrementally with respect to the learned co-certificates.

Co-certificate learning is a general method that applies to any alternating search problem, but it is much easier to explain on a concrete example like graph coloring. In Sections 4 and 5, we present a more involved application of CCL, to the existence of *Kochen-Specker (KS) systems*, a combinatorial object that features in the proof of the Bell-Kochen-Specker theorem in quantum mechanics. With SMS+CCL we achieve an orders-of-magnitude speed-up over a different recently proposed SAT-based approach to the KS problem, and we prove that any KS system must have at least 24 vectors.

2 Preliminaries

For a positive integer n, we write $[n] = \{1, 2, \ldots, n\}$. We assume familiarity with fundamental notions of propositional logic [Prestwich, 2009]. In this paper we are presenting a general method for alternating combinatorial search problems on particular examples with graphs; below we review basic notions from graph theory relevant to our discussion.

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. The *order* of a graph G is the number of its vertices, $|V(G)|$. We write $\mathcal{G}_n$ to denote the class of all graphs with $V(G) = [n]$. The *adjacency matrix* of a graph $G \in \mathcal{G}_n$, denoted by A_G, is the $n \times n$ matrix where the element at row v and column u, denoted by $A_G[v][u]$, is 1 if $vu \in E$ and 0 otherwise.

Isomorphisms. For a permutation $\pi : [n] \to [n]$, $\pi(G)$ denotes the graph obtained from $G \in \mathcal{G}_n$ by the permutation π, where $V(\pi(G)) = V(G) = [n]$ and $E(\pi(G)) = \{\, \pi(u)\pi(v) : uv \in E(G) \,\}$. Two graphs $G_1, G_2 \in \mathcal{G}_n$ are *isomorphic* if there is a permutation $\pi : [n] \to [n]$ such that $\pi(G_1) = G_2$; in this case G_2 is an *isomorphic copy* of G_1.

Coloring. A *(proper) k-coloring* of a graph G is a map $c : V(G) \to [k]$ with the property that if $uv \in E(G)$, then $c(u) \neq c(v)$ (adjacent vertices have different colors).

Partial graphs. 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 $U(G) = \emptyset$. A partially defined graph G can be *extended* to a graph H if $D(G) \subseteq E(H) \subseteq D(G) \cup U(G)$.

SAT Modulo Symmetries (SMS). SMS is a framework that augments a CDCL (conflict-driven clause learning) SAT solver [Fichte *et al.*, 2023; Marques-Silva *et al.*, 2009] with a custom propagator that can reason about symmetries, allowing to search modulo isomorphisms for graphs in $\mathcal{G}_n$ which satisfy constraints described by a propositional formula. During search the SMS propagator can trigger additional conflicts on top of ordinary CDCL and consequently learn *symmetry-breaking clauses*, which exclude isomorphic copies of graphs. More precisely, only those copies which are lexicographically minimal (*canonical*) by concatenating the rows of the adjacency matrix are kept. A key component is a minimality check, which decides whether a partially defined graph can be extended to a minimal graph; if it cannot, a corresponding clause is learned. For a full description of SMS, we refer to the original work where the framework was introduced [Kirchweger and Szeider, 2021].

3 Co-Certificate Learning (CCL)

In this paper we propose to extend SMS to alternating search problems. The way we do this is by applying SMS incrementally. In this section we shall explain the procedure by showing how it would play out in our running example of finding a triangle-free graph with n vertices and without a k-coloring.

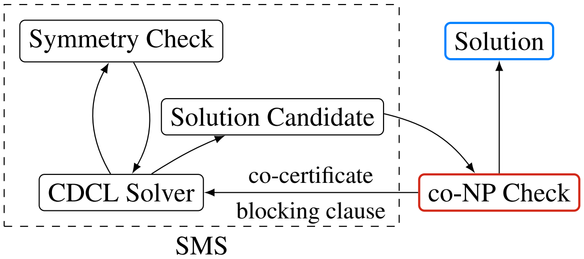

Figure 1: Co-certificate learning = SMS + co-NP property check.

Throughout this section we will talk about propositional formulas that encode graphs with the vertices $[n]$ using the variables $e_{u,v}$, where $u, v \in [n]$. Any truth assignment α defines the (possibly partially defined) graph G_α with $V(G_\alpha) = [n]$ and $E(G_\alpha) = \{uv \; : \; \alpha(e_{u,v}) = 1\})$.

First, we generate a triangle-free graph, which is obtained from a SAT solver as a model of a (polynomial-size) propositional formula encoding triangle-freeness—the clauses

$$\neg e_{u,v} \vee \neg e_{v,w} \vee \neg e_{u,w} \text{ for } u, v, w \in V, u < v < w$$

forbid all triangles. We feed the candidate graph into a custom coloring algorithm and test whether it is k-colorable. We will only deal with small enough graphs that the time cost of coloring, though exponential in the worst case, is going to be negligible. On the other hand, it could be the case that candidate graphs have many colorings, and the *choice* of a particular coloring might matter much more. We will elaborate on this in Section 5.1.

Returning to our candidate graph, if it is not colorable, we obtain a solution. Depending on whether we want just one or all solutions we can stop or add a clause blocking this particular solution and resume. If, on the other hand, the graph is k-colorable, we obtain a certificate for this in the form of a coloring $c : [n] \to [k]$. We learn and add a *coloring clause*

$$C_c = \bigvee_{u<v\in[n], c(u)=c(v)} e_{u,v}.$$

This clause says that future generated graphs must not be colorable by the coloring c.

Finally, we go back to step one and generate another graph, this time respecting the newly learned coloring clause. We repeat this until either a non-k-colorable graph is found or the formula becomes unsatisfiable.

The full process is illustrated in Figure 1. The crucial innovation is the learning of coloring clauses. Thanks to the fact that a single coloring is valid for multiple graphs, we do not have to visit every candidate graph in the search as we would with a naive method that does not interleave graph generation with coloring. Instead, it is sufficient to ‘visit enough colorings,’ in the sense that once a set of colorings (and their corresponding clauses) that jointly color all colorable graphs of order n is generated (and any non-colorable graphs are found), the formula immediately becomes unsatisfiable.

We call this process *co-certificate learning (CCL)*, after the fact that each learned coloring is a ‘co-certificate’ that the graph does *not* have the desired non-colorability property. It follows immediately that such a co-certificate learning can be formulated for any combinatorial search problem where the sought-for object has a co-NP property. Here we chose to present CCL on the conceptually simpler triangle-free non-colorable problem, but in Section 5 we will present a more involved showcase for SMS+CCL, one where it goes significantly beyond the state of the art.

We have glossed over this until now, but CCL is not running in an ordinary SAT solver but on top of SMS. This means every now and then we hook into the SAT solver and check whether the currently considered graph, which may not be fully defined, is canonical. If it is not canonical, we learn a clause that causes the solver to immediately backtrack. This is the same as ordinary SMS, and likewise, all candidate graphs produced are canonical and thus unique w.r.t. isomorphism. As a consequence, the set of co-certificates (colorings) that we need to learn is in fact only required to work for canonical graphs, and not all graphs of order n.

3.1 Related Methods

In the last 20 years many different methods have been developed to extend SAT solvers with external algorithms in order either to improve performance or to grant SAT solvers the ability to solve non-Boolean problems. While these methods are superficially similar, the specifics of how the collaboration is carried out and the type of external algorithm differ significantly. Among the best known are *SAT modulo Theories (SMT)* [Barrett *et al.*, 2021] and *lazy clause generation (LCG)* [Ohrimenko *et al.*, 2009]; other methods that fall into this category are the *SAT+CAS* system [Zulkoski *et al.*, 2017], where the SAT solver communicates with a computer algebra system (CAS), and *counterexample-guided abstraction-refinement (CEGAR)* solvers for quantified Boolean formulas (QBF) [Janota *et al.*, 2012], which typically use two or more communicating SAT solvers.

Our approach, SMS+CCL, also belongs to this family. SMS delegates the symmetry breaking to a specialized algorithm; CCL delegates the co-certificate search to another specialized algorithm. The most similar approach to CCL is an abstraction-based algorithm for 2QBF due to Janota and Marques-Silva [2011], which solves formulas of the form $\exists X \forall Y(\neg\phi)$ where ϕ is a CNF formula. The part in common with our method is that solution candidates are computed and excluded by adding additional clauses if necessary. What makes SMS+CCL stand out is the high degree of integration of its components and the fact that the existential and universal parts are handled separately. Both symmetry breaking and co-certificate learning are carried out by our code, tightly integrated into a single binary, having access to the entire state of the solver. Another hallmark of our approach is that we focus on graphs (and, more generally, combinatorial objects), so our symmetry-breaking and co-certificate learning do not operate on a propositional encoding (as one is forced to when solving pure QBFs, for example). However, they understand the high-level structure and can thus be more specialized and effective. At the same time, our method remains general enough to apply to new alternating search problems with only a little implementation effort.

n	Nauty	PSS+CCL	SMS+CCL
10	3.06	0.02	0.08
11	29.23	0.10	0.15
12	375.38	1.67	0.42
13	6507.02	774.39	2.95
14	$\sim$ 2 days	>4 days	147.57

Table 1: Running times in seconds of different approaches to the Δ-free non-colorable problem. The time for Nauty includes post-processing to filter out 3-colorable graphs, using the same code as for the other methods.

3.2 Comparison with Alternatives

For the remainder of this section, we will present an experimental comparison of SMS+CCL to various alternative methods on the task of finding non-3-colorable triangle-free graphs with 10–14 vertices, showing how we can visit fewer candidate graphs and learn small sets of colorings thanks to SMS and CCL. The methods we compare are:

- Nauty [McKay and Piperno, 2014]: enumeration modulo isomorphism of triangle-free graphs via Nauty's built-in `geng -t`, followed by filtering out non-3-colorable graphs by our custom code;
- PSS+CCL: enumeration of triangle-free graphs with a SAT solver and with *partial static symmetry breaking* [Codish *et al.*, 2019], interleaved with CCL for 3-colorability (same custom code as above, custom implementation of the symmetry breaking constraints);
- SMS+CCL: as described above (using the same SAT solver as PSS).

Another alternative that we did not consider for this comparison is to use full static symmetry breaking, which produces a constraint of exponential size and is known to scale to no more than 12 vertices [Codish *et al.*, 2016].

From Table 1 we can see that CCL provides a speed-up of many orders of magnitude over isomorphism-free exhaustive generation with Nauty followed by 3-colorability filtering. Table 2 explains this speed-up clearly in terms of the number of graphs that need to be enumerated. Nauty always needs to enumerate all (non-isomorphic) graphs, the number in the first column. This is many orders of magnitude larger than the number of (fully defined) graphs visited by SMS + CCL: those in the third column. Even more striking is the tiny number of colorings that are sufficient to color all canonical graphs, shown in the parentheses in the last column of Table 2. In the case of $n = 14$, a single 3-coloring covers almost 50000 canonical graphs on average! Incomplete static symmetry breaking on the other hand, while allowing the use of CCL, is not as good at reducing the search space, as can be witnessed by the running times and the number of graphs enumerated, as well as the total number of graphs that pass the symmetry breaking constraint, shown in Table 2.

Note that in this comparison we did not tweak the details of the methods so as to extract every last bit of performance (see, e.g., the work by Goedgebeur [2020] for a high-tech Nauty-based solution to the triangle-free non-colorable problem). The point of this comparison is to demonstrate the inherent

n	Nauty	PSS	PSS+CCL	SMS+CCL
10	12172	309987	171	54(54)
11	105071	9969561	618	147(146)
12	1262180		3668	505(481)
13	20797002		171780	3124(2014)
14	467871369		$> 2 \times 10^7$	85668(9407)

Table 2: 1st column: the number of non-isomorphic Δ-free graphs of order n; Nauty has to process all of them. 2nd column: the number of Δ-free graphs with n vertices that pass the PSS constraint of Codish *et al.* [2019] (we did not compute the missing entries, because $n = 11$ already took more than 8 hours). This gives an idea of the general effectiveness of PSS vis-à-vis the true number of non-isomorphic graphs. 3rd and 4th column: the number of graphs visited by PSS+CCL and SMS+CCL respectively, in parentheses of the 4th column the number of colorings (co-certificates) learned by SMS+CCL; contrast this number with the vastly larger total number of graphs in the first column.

limits of each of these approaches, as well as to compare the 'off-the-shelf' versions of every method.

We thus conclude that, at least in the context of alternating problems like graph coloring, co-certificate learning is absolutely crucial to prevent exhaustive enumeration, and SMS is the symmetry-breaking method that scales best.

Having introduced our general framework of co-certificate learning, we shall now turn our attention to the interesting multi-faceted application that is the search for a smallest Kochen-Specker vector system.

4 Kochen-Specker Systems

Kochen-Specker (KS) vector systems are specific sets of vectors in at least 3-dimensional space that form the basis of the Bell-Kochen-Specker Theorem, a central result in the foundations of quantum mechanics. The existence of a KS vector system shows that quantum mechanics is in conflict with classical models in which the result of a measurement does not depend on which other compatible measurements are jointly performed, a phenomenon known as *contextuality* [Budroni *et al.*, 2022]. In their original 1967 paper, Kochen and Specker [1967] proposed a 3-dimensional KS vector system of size 117. Since then, researchers have been striving to find smaller KS vector systems and establish lower bounds on their size. KS vector systems of a higher dimension $n \geq 3$ are also considered in the literature, with the additional property that any pair of vectors belongs to a set of n mutually orthogonal vectors [Pavičić *et al.*, 2005]. Higher dimensions allow for smaller KS systems, whereas the additional property increases the size of smallest KS systems. In the following we focus on KS vector systems of dimension 3.

The smallest KS vector system (of dimension 3) known to date is due to Conway and Kochen and has 31 vectors [Peres, 1991]. The first lower bound, of 18, was given by Arends *et al.* [2011]; later it was improved to 22 by Uijlen and Westerbaan [2016], and recently to 23 by Li *et al.* [2022]. All these lower bounds were obtained by computer search methods for undirected graphs associated with KS vector systems. Since the search space is enormous and further increases by several orders of magnitude with each increment, the methods must become more and more sophisticated. Arends *et al.* obtain their lower bound with backtracking search that rules out all KS vector systems with 17 or fewer vectors; Uijlen and Westerbaan improved this method by deriving additional necessary restrictions on the graphs, ruling out KS vector systems with up to 21 vectors. Li *et al.* propose a SAT-based method where the SAT solver interacts with a computer algebra system (CAS) and prove non-existence of KS systems with 22 vectors. We will discuss the difference between their and our approach below.

We give a lower bound of 24 using SMS+CCL to exclude KS systems with up to 23 vectors.

Next, we define the properties of graphs associated with KS vector systems, closely following Arends *et al.*'s approach, which has also been adopted by subsequent authors [Uijlen and Westerbaan, 2016; Li *et al.*, 2022].

A *KS graph* is a simple undirected graph which is not 010-colorable but embeddable, where these two properties are defined as follows. A graph G is *010-colorable* if we can assign 0 or 1 to its vertices in such a way that (i) no two adjacent vertices are both assigned 0, and (ii) vertices forming a triangle are not all assigned 1. G is *embeddable* if its vertices can be mapped to three-dimensional real vectors such that adjacent vertices are orthogonal and there is no collinear pair. Figure 2 shows Conway and Kochen's 31-vertex KS graph and an embedding.

As shown by Arends *et al.* [2011], there exists a KS vector system with n vectors if an only if there exists a KS graph with n vertices. Moreover, Arends *et al.* give the following necessary properties of any smallest KS graph G:

1. G is square-free (i.e., has no cycle of length 4);
2. G is 4-colorable;
3. G has minimum degree at least 3;
4. each vertex of G belongs to a triangle.

Non-010-colorable graphs satisfying the previous four necessary properties are called *KS candidates*. They are only 'candidates', because they are not guaranteed to be embeddable. All known lower bounds on the size of a KS system were obtained by enumerating all KS candidates (modulo isomorphisms), and checking that none are embeddable. We now turn to the description of our own process for proving KS lower bounds, which also follows this two-phase pattern, but differs in our use of the new SMS+CCL method for the enumeration phase, and in some aspects of the embeddability check. We note that checking non-010-colorability is co-NP-complete [Arends *et al.*, 2011], hence this property is well-suited for our method.

5 New Lower Bound for KS Systems

This section is devoted to a description of our SAT encoding for the search of KS graphs, followed by a discussion of experimental results. We first produce a formula F_n, whose non-010-colorable models correspond to KS candidates; we then enumerate the non-010-colorable models of F_n with SMS+CCL; and finally we check the obtained graphs for embeddability.

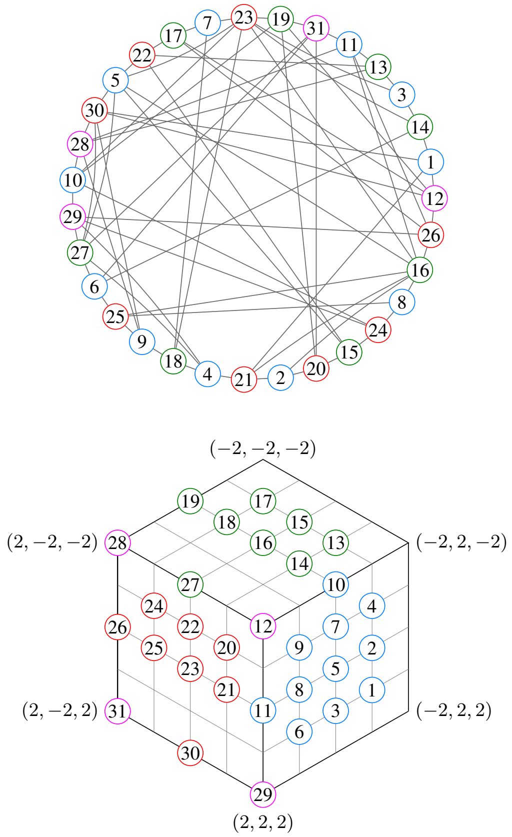

Figure 2: Top: smallest known KS graph with 31 vertices and 71 edges, drawn properly 4-colored. Bottom: an embedding of this graph, where each vertex corresponds to a vector starting at the center of the cube. For instance, the vertices 1 and 30 are adjacent, and hence their corresponding vectors $(-1, 2, 1)$ and $(2, 0, 2)$, respectively, are indeed orthogonal (i.e., perpendicular) since $-1 \cdot 2 + 2 \cdot 0 + 2 \cdot 1 = 0$.

5.1 Encoding and CCL

We use the variables $e_{v,u}$ to indicate whether a certain edge is present (*edge variables*) and the auxiliary variables t_{v_1,v_2,v_3} indicating whether v_1, v_2, v_3 forms a triangle. A graph can be extracted from a model of F_n by looking at the assignment of the edge variables. Our encoding is almost identical to previous work [Li *et al.*, 2022] with the exception that coloring clauses are learned lazily. The necessary conditions mentioned at the end of Section 4 are encoded as follows:

1. $\neg e_{v_1,v_2} \vee \neg e_{v_2,v_3} \vee \neg e_{v_3,v_4} \vee \neg e_{v_1,v_4}$ for $v_1, v_2, v_3, v_4 \in [n]$ where $v_i \neq v_j$ if $i \neq j$ and $v_1 < v_2 < v_4$, $v_1 < v_3$, to forbid only one orientation of each 4-cycle.
2. To express that G must be 4-colorable, we need additional variables $c_{v,i}$, $v \in [n], i \in [4]$, for the color of each vertex. To ensure each vertex is colored we add the clauses $\bigvee_{i \in [4]} c_{v,i}$ for $v \in [n]$, and to avoid monochromatic edges we add $\neg e_{v,u} \vee \neg c_{v,i} \vee \neg c_{u,i}$ for $i \in [4]$.
3. To enforce minimum degree at least 3, we use sequential counters [Sinz, 2005].
4. The clauses $\bigvee_{v_2,v_3 \in [n] \setminus \{v_1\}, v_2 < v_3} t_{v_1,v_2,v_3}$ for $v_1 \in [n]$ ensure that each vertex lies on a triangle.

Additionally, we have definitional clauses to ensure that $t_{v_1,v_2,v_3} \leftrightarrow (e_{v_1,v_2} \wedge e_{v_2,v_3} \wedge e_{v_1,v_3})$ holds.

Whenever SMS returns a fully defined graph G respecting the above constraints, we check if it is 010-colorable. We use a simple backtracking algorithm for constructing 010-colorings. Given a 010-coloring c of a graph G, the clause

$$\bigvee_{c(v)=c(u)=0} e_{v,u} \vee \bigvee_{c(v_1)=c(v_2)=c(v_3)=1} t_{v_1,v_2,v_3}$$

blocks all graphs colored by c.

Our 010-coloring algorithm always attempts to color the vertices in a fixed order, but in case both colors are available for the next vertex, it uses a heuristic to determine the order in which the two colors should be tried. This heuristics has access to edge frequencies among previously seen graphs, and prefers the color whose blocking clause will be less likely satisfied based on the frequency analysis. The intuition behind this style of heuristic is this. Edge distribution in canonical graphs is highly unbalanced; therefore different possible learnable clauses have a different likelihood of being falsified and leading to further conflicts, depending on which edge variables they contain. We attempt to find colorings that will give rise to clauses where many literals will be falsified because their edges hardly ever occur in canonical graphs; we use the edge frequency statistics to guide the heuristic.

Taken to the extreme, we could, instead of computing a single heuristic 010-coloring, explicitly optimize in the space of all 010-colorings. Our heuristic results are already good enough, and this is a bit out of the scope of this paper, but we believe interesting research questions may lie down this path.

5.2 Parallelization

With our approach we are in fact able to verify KS nonexistence for all $n \leq 22$ in time that is still practical for single-core sequential execution (see Table 4). In order to solve $n = 23$, however, we will need to parallelize.

The most successful approach to parallelization in the context of combinatorial search problems is called *cube-and-conquer* [Heule *et al.*, 2018], where the original problem is split by applying mutually exclusive assumptions (*cubes*) whose disjunction is implied by the original formula. The main challenge with cube-and-conquer is finding cubes so that the subproblems are evenly hard. In the original *cube-and-conquer* paper, Heule *et al.* use a look-ahead solver for generating the cubes. The problem with this approach in our case is that we would prefer cubes that take into account graph canonicity, which significantly skews the hardness and distribution of solutions in the search space, and which is not directly represented in the encoding itself. In order to construct representative cubes, we would effectively have to reimplement SMS in a look-ahead solver; we instead opted for the simpler option of using the same SMS solver for the cubing process as for the subsequent solving.

We proceed as follows for generating the cubes. We start the solver and whenever the number of assigned edge variables exceeds a pre-defined threshold, we add the partial assignment on the edge variables to the list of cubes, and exclude the current partial assignment on the edge variables from the solver. We continue until the solver concludes unsatisfiability. To further improve this approach, we first let the solver run for a specific amount of time (*prerun*) before generating the cubes. The idea behind this is to let the solver learn some symmetry and coloring clauses first in order to get a more representative picture of the search space.

Note that with this technique the generated cubes are not necessarily mutually exclusive, i.e., one model can agree with several different cubes on the assigned variables. As we will see in our experiments, we can easily filter them.

Another important point is that the original encoding F_n alone does *not* imply the disjunction of our cubes, as would be the case in pure cube-and-conquer. The reason is that we learn symmetry-breaking and coloring clauses along the way, and these further restrict the search space so that when the formula becomes unsatisfiable it is not just due to the cubes but also due to these additional clauses. Instead, F_n together with the learned symmetry-breaking and coloring clauses implies the disjunction of the cubes; we shall say F_n entails the cubes *with advice*.

5.3 Embeddability Check

The next step in the process of finding KS graphs is to check the candidate, non-010-colorable graphs for embeddability. This check amounts to solving a first-order formula over multivariate polynomial equations over the real numbers, a problem that can be solved for example by the REDUCE algebra system [Dolzmann and Sturm, 1997] or the SMT solver Z3 [de Moura and Bjørner, 2008]. Because these equations can be hard, following previous work [Li *et al.*, 2022], we first check whether the graph contains some known unembeddable subgraph[1], and only if it does not, we resort to a full-blown Z3 embeddability check. We use the Glasgow Subgraph Solver [McCreesh *et al.*, 2020] to test subgraph containment, being able to test all our candidates in less than an hour of CPU time in total. After this process, we are left with only two candidates which do not contain any known unembeddable subgraph of order ≤ 14, and for those we have to fire up Z3. We next describe the encoding of embeddability, which also requires some non-trivial tricks.

Given a graph G we want to decide whether we can find for each vertex three-dimensional real vectors so that adjacent vertices map to orthogonal vectors, and no two vectors are collinear. We could simply write down these constraints in the language of Z3, but that would require 3 real-valued variables for the coordinates of each vertex, and seemed too difficult to solve when we tried it. It turns out we can save a good chunk of the variables by substituting cross products for some of the vertices, as follows.

Since we are only interested in angles (orthogonality) and not magnitudes, we are effectively trying to map the vertices (injectively) to 'directions' (1-dimensional subspaces) in $\mathbb{R}^3$. Imagine we guess the directions for individual vertices one by one. Once two vertices have their directions determined, any shared neighbor has only a single possible direction left, the direction given by the cross product. It so happens that guessing the coordinates of only very few vertices often uniquely determines all other coordinates via cross products. We formalize this phenomenon in the notion of a *cross-product cover*.

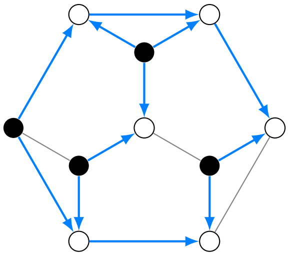

Figure 3: An illustration of a cross-product cover of a graph with ten vertices, one proved unembeddable by Uijlen and Westerbaan [2016]. Free vertices are in black, arrows indicate the cross-product relation. The remaining edges are drawn as simple lines.

We say (S, w) is a cross-product cover of a graph G if $S \subseteq V(G)$, w maps each vertex $v \in V(G) \setminus S$ to a pair of its neighbors $w_v = \{w_v^1, w_v^2\}$, $vw_v^1, vw_v^2 \in E(G)$, so that the transitive closure of $\{ (u, v) : v \in V(G) \setminus S, u \in w_v \}$ is anti-symmetric. In other words, given the coordinates of vectors corresponding to the vertices in S, the vectors corresponding to the remaining vertices are uniquely determined by taking successive cross products, and there are no circular references. We call the vertices in S and their vectors *free*, and the others *bound*. Figure 3 gives an example of a cross-product cover.

We write p_v for the vector assigned to a vertex v. Without loss of generality, we can assume that for one edge $v_1 v_2 \in E(G)$ the corresponding coordinates are $p_{v_1} = (1, 0, 0)$, and $p_{v_2} = (0, 1, 0)$, provided that v_1 and v_2 are free vertices.

Observation 1. *Let G be a graph, (S, w) a cross-product cover, $v_1 v_2 \in E(G)$ an edge between free vertices. Then G is embeddable iff there is a function $p : V(G) \to \mathbb{R}^3$ satisfying:*

1. $p_{v_1} = (1, 0, 0)$, $p_{v_2} = (0, 1, 0)$*;*
2. $p_v \times p_u \neq 0$ *for all $u, v \in V(G), u \neq v$: vectors are not collinear;*
3. $p_v = p_{w_v^1} \times p_{w_v^2}$ *for all bound v, where $w_v = \{w_v^1, w_v^2\}$;*
4. $p_v \cdot p_u = 0$ *for all $vu \in E(G)$: the inner product must be zero if p_v and p_u are orthogonal.*

Given a graph G, for any cross-product cover (S, w) and an edge $v_1 v_2 \in E(G)$ between free vertices, Observation 1 yields a set of constraints in polynomial real arithmetic that can be solved using Z3. Note that with the cross-product formulation we need variables only for free vectors (two of which are fixed), a total of only $3(|S| - 2)$ real-valued variables. For most graphs we were able to find a cross-product cover with $|S| \leq 4$. When we were unable to solve one set of constraints in 10 seconds we tried another cross-product cover. We also tried normalizing the free vectors, but surprisingly this only made the problem harder.

5.4 Comparison to SAT+CAS

We have now laid out our process of search for KS systems. Before we discuss experimental results and our new lower

[1] https://kochen-specker.info/smallGraphs/ provides a list of all unembeddable subgraphs with minimum degree 3 up to order 14.

bound on the size of a KS system, let us compare SMS+CCL to the SAT+CAS approach recently used for the same problem [Li *et al.*, 2022].[2]

Both approaches have in common that they are based on SAT and are able to enumerate all graphs up to isomorphism described by an input encoding. The first difference is in the canonical form: both produce lexicographically minimal graphs, but SMS produces lexicographically minimal graphs defined by concatenating the rows of the adjacency matrix whilst the SAT+CAS approach concatenates the above-diagonal entries of the columns. An important property of the latter is that the subgraph given by the first k vertices of a canonical graph is also canonical.

SMS uses a partition-based minimality check based on the partially defined graph which allows to construct symmetry breaking clauses even if only a few edges are assigned. In the SAT+CAS approach, one checks if the induced subgraph given by the first $k \leq n$ vertices is canonical, whenever all edge-variables between the first k vertices are assigned. A comparison between the amount of time spent in the solver and for the minimality check would be interesting.

The two approaches also differ in the way of ensuring that the resulting graphs are non-010-colorable. We add coloring clauses dynamically with CCL, while Li *et al.* added all coloring clauses for 010-colorings where at most $\lceil \frac{n}{2} \rceil$ vertices have color 1 upfront, leading to an exponential number of clauses.

The third difference is that Li *et al.* did not check for canonical form whilst cubing; they only used incomplete static symmetry-breaking constraints. This has the advantage that a different solver can be used, and they used the march-cu solver [Heule *et al.*, 2018], but it foregoes the extra power of entailment with advice (see Subsection 5.2) and so might produce more cubes. As we will see in our experiments, we enforce larger cubes in comparison to SAT+CAS; but since Li *et al.* do not report the number of cubes, we cannot tell whether at the same time we produce more cubes. We suspect that SMS should be able to assign more variables thanks to the minimality check based on partially defined graphs.

5.5 Computations

In this section, we will describe our experimental setup. In the original SMS paper [Kirchweger and Szeider, 2021], Clingo was used as solver. In our implementation, the user can choose between Cadical with the IPASIR-UP interface [Fazekas *et al.*, 2023] (a state of the art incremental CDCL SAT-solver with inprocessing) and Clingo (an ASP solver containing a CDCL SAT solver). For our experiments we decided to use Cadical. The SAT encodings are created by a Python script and the embeddability check is also implemented in Python using Z3's Python interface. All our code is available on GitHub[3].

We ran our experiments on a cluster with different processors[4] under Ubuntu 18.04. We use version 4.11.2 of Z3 and all tests are executed with a single thread.

n	#existential	#colorings
13	34	3
14	216	10
15	2352	32
16	27394	91
17	373646	267
18	6050114	832

Table 3: Comparison between the number of graphs satisfying the existential part F_n of the encoding of KS candidates and the number of colorings computed to color all 010-colorable graphs. For $n = 17$ there is one non-010-colorable graph among the 373646, all others counted in this table are 010-colorable.

n	#KS candidates	times
17	1	12.93 sec (1.2 min)
18	0	71.67 sec (7.8 min)
19	8	8.43 min (2.46 hrs)
20	147	1.68 hrs (39.71 hrs)
21	2497	26.24 hrs (42.56 days)
22	88282	26.17 days (5.26 years)
23	**3747950**	**1.36 years (not computed)**

Table 4: Number of KS candidates. Time gives the total CPU time for solving this instance (including cubing for $n \in \{22, 23\}$) and in brackets the times from Li *et al.* [2022] ran on Intel E5-2683 v4@2.1GHz CPUs are given. The times for Clingo+SMS for $17 \leq n \leq 20$ are 8.73, 70.47, 684, and 3904 seconds respectively.

We start with an experiment (Table 3) showing the impact of CCL for computing KS candidates by computing the number of graphs satisfying the existential part F_n and the number of 010-colorings needed to filter out all 010-colorable graphs. Similarly to Table 2, we can see that the number of learned colorings is significantly lower than the total number of graphs satisfying the existential part, illustrating that CCL is a powerful technique for computing KS candidates.

Table 4 summarizes the computation times for finding all KS candidates for a given order n and gives the computation times provided from Li *et al.* [2022] for comparison. The number of KS candidates up to 22 vertices coincides with our computations. For $n \in \{22, 23\}$ we applied cube-and-conquer for parallelization, some details including prerun time (see Section 5.2), the number of assigned edge variables, number of cubes, average solving time and longest time for a cube are given in Table 5. Note that there is a large difference between the average time and the longest time for a cube. In the future, we want to investigate methods for generating more balanced cubes.

As already mentioned in Section 5.2 the KS candidates can have duplicates. This is indeed the case, for example for $n = 23$ we got a total of 3752684 graphs, of which 4734 were duplicates, resulting in 3747950 unique graphs. Only two of them do not contain an unembeddable subgraph of size ≤ 14 (we call them *odd*). We use Z3 to verify that these two are also unembeddable. We thus prove Theorem 1.

Theorem 1. *Every KS-graph has at least* 24 *vertices.*

[2] The implementation of the latter is not available online.

[3] https://github.com/markirch/sat-modulo-symmetries

[4] Intel Xeon {E5540, E5649, E5-2630 v2, E5-2640 v4}@ at most 2.60 GHz, AMD EPYC 7402@2.80GHz

n	#cubes	prerun	#var.	$\bar{t}$	$\max(t)$
22	18659	2 days	120	112 sec	8946 sec
23	313665	2 days	140	137 sec	63812 sec

Table 5: Some details of constructing and solving the cubes for $n \in \{22, 23\}$. Column "#cubes" gives the number of cubes, "prerun" the prerun time before starting cubing, "#var." the lower-bound on literals for each cube, "$\bar{t}$" the average time for solving a cube, and "$\max(t)$" the time of the hardest cube to solve.

5.6 Proofs for Computations

Proving the absence of KS graphs of order ≤ 23 involves several computations. We discuss next how one can produce machine-verifiable proofs for most of these steps. For the results generated by SMS, we can produce DRAT proofs [Wetzler *et al.*, 2014] in a similar way as described by Kirchweger *et al.* [2022], assuming that the symmetry-breaking and coloring clauses are part of the original encoding. We can check soundness of symmetry-breaking and coloring clauses separately. To simplify this validation, we can store permutations and colorings associated with these clauses.

As discussed in Subsection 5.2, the disjunction of the cubes must be entailed with advice by the formula F_n to ensure all models are accounted for. While constructing the cubes, we exclude each partial assignment corresponding to a constructed cube from the search space, and thus the cubing process ends with the formula

$$\Omega = F_n \wedge (\neg c_1 \wedge \ldots \wedge \neg c_k) \wedge \Sigma \wedge \Gamma,$$

where Σ and Γ are the learned symmetry-breaking and coloring clauses. Hence, a DRAT proof of unsatisfiability of Ω shows the completeness of the cubing process. Note that we may find some KS candidates already during the prerun before cubing; this is not a problem, a fully defined graph is just another cube.

Solving the formula under the assumption of a cube c_i is equivalent to solving $F_n \wedge c_i$. During search we might find some KS candidates represented by the cubes $g_1, \ldots, g_t$ and exclude them from the search space, as well as learn symmetry clauses Σ_i and coloring clauses Γ_i. Consequently, a DRAT proof of

$$F_n \wedge c_i \wedge \Sigma_i \wedge \Gamma_i \wedge \neg g_1 \wedge \ldots \wedge \neg g_t$$

shows that there are no more KS candidates for the assumption c_i.

We calculated an estimate of the proof size for $n = 23$ by producing proofs for 5% of the cubes. Assuming them to be representative, the proof would have around 5 terabytes.

When we prove a graph unembeddable because it contains a small known unembeddable subgraph, we can simply take the embedding given by the subgraph solver as a certificate.

The final case is the full embeddability check when no known unembeddable subgraph is found. Verifying that a cross-product cover (S, w) of G is correct is straightforward, but unfortunately Z3 does not produce checkable proofs for nonlinear real arithmetic at the moment. We tested our implementation by deciding the embeddability of all square-free graphs with degree at least 2 up to order 12. The results coincide with previous work. Interestingly, we found two graphs claimed minimally unembeddable by previous works, from which an edge can be removed while preserving unembeddability. The graphs are still minimally unembeddable with respect to vertex removal, and also among graphs with minimum degree 3. Further, we extracted a common unembeddable subgraph of order 15 from the two odd graphs by greedily removing as many vertices as possible while preserving unembeddability. This graph is shown in Figure 4.

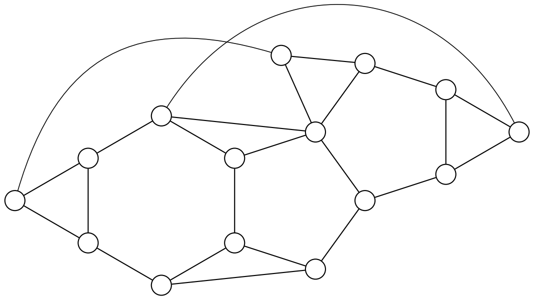

Figure 4: The shared 15-vertex subgraph of the two *odd* graphs from Subsection 5.5. Its graph6 string is `NGw@?i??GHgA@aCtQC?`.

6 Conclusion

This work extended the SAT modulo Symmetries (SMS) framework with the co-certificate learning (CCL) technique for alternating search problems. We showed that SMS+CCL can drastically reduce the search space compared to alternative methods, and demonstrated its potential by applying it to the prominent problem arising from quantum mechanics of determining the smallest Kochen-Specker (KS) graphs. With our method we improved the best known lower bound and proved that KS graphs have at least 24 vertices.

A natural question is whether we can settle the case for $n = 24$ with our current methods. We generated cubes for $n = 24$ and tested 10% of them; we expect a solving time of roughly 125 CPU years assuming that the solving time of the cubes is representative. We plan to improve our current approach by excluding unembeddable partially defined graphs during the search.

Another point for improvement is finding techniques for producing more balanced cubes, minimizing the solving time for the hardest cubes.

SMS+CLL is a general technique applicable to many problems. In recent work, we applied SMS+CLL to make progress on the Erdős-Faber-Lovász Conjecture [Kirchweger *et al.*, 2023a] and there are many more problems which we plan to attack, for example, computing small non-2-colorable n-uniform hypergraphs [Östergård, 2014], finding color-critical graphs with few edges [Jensen and Toft, 1995], computing width-critical graphs for minor-closed width measures [Chlebíková, 2002].

Acknowledgments

We acknowledge support from the Austrian Science Fund (FWF), projects P32441 and J4361, and from the Vienna Science and Technology Fund (WWTF), project ICT19-065.

References

[Arends *et al.*, 2011] Felix Arends, Joël Ouaknine, and Charles W. Wampler. On searching for small Kochen-Specker vector systems. In Petr Kolman and Jan Kratochvíl, editors, *Graph-Theoretic Concepts in Computer Science - 37th International Workshop, WG 2011, Teplá Monastery, Czech Republic, June 21-24, 2011. Revised Papers*, volume 6986 of *Lecture Notes in Computer Science*, pages 23–34. Springer Verlag, 2011. Based on Felix Arends' MSc thesis "A Lower Bound on the Size of the Smallest Kochen-Specker Vector System in Three Dimensions," University of Oxford, 2011.

[Barrett *et al.*, 2021] Clark Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. Satisfiability modulo theories. In A. Biere, M. J. H. Heule, H. van Maaren, and T. Walsh, editors, *Handbook of Satisfiability, 2nd Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, chapter 33, pages 1267–1329. IOS Press, 2021.

[Budroni *et al.*, 2022] Costantino Budroni, Adán Cabello, Otfried Gühne, Matthias Kleinmann, and Jan øAke Larsson. Kochen-Specker contextuality. *Rev. Mod. Phys.*, 94:045007, 2022.

[Chlebíková, 2002] Janka Chlebíková. The structure of obstructions to treewidth and pathwidth. *Discr. Appl. Math.*, 120(1-3):61–71, 2002.

[Codish *et al.*, 2016] Michael Codish, Graeme Gange, Avraham Itzhakov, and Peter J. Stuckey. Breaking symmetries in graphs: The nauty way. In *Principles and Practice of Constraint Programming - 22nd International Conference, CP 2016, Toulouse, France, September 5-9, 2016, Proceedings*, volume 9892 of *Lecture Notes in Computer Science*, pages 157–172. Springer Verlag, 2016.

[Codish *et al.*, 2019] Michael Codish, Alice Miller, Patrick Prosser, and Peter J. Stuckey. Constraints for symmetry breaking in graph representation. *Constraints*, 24(1):1–24, 2019.

[de Moura and Bjørner, 2008] Leonardo Mendonça de Moura and Nikolaj S. Bjørner. Z3: an efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer Verlag, 2008.

[Dolzmann and Sturm, 1997] Andreas Dolzmann and Thomas Sturm. REDLOG: computer algebra meets computer logic. *SIGSAM Bull.*, 31(2):2–9, 1997.

[Erdős, 1967] P. Erdős. Some remarks on chromatic graphs. *Colloq. Math.*, 16:253–256, 1967.

[Fazekas *et al.*, 2023] Katalin Fazekas, Aina Niemetz, Mathias Preiner, Markus Kirchweger, Stefan Szeider, and Armin Biere. IPASIR-UP: User propagators for CDCL. In Meena Mahajan and Friedrich Slivovsky, editors, *The 26th International Conference on Theory and Applications of Satisfiability Testing (SAT 2023), July 04-08, 2023, Alghero, Italy*, LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. To appear.

[Fichte *et al.*, 2023] Johannes K. Fichte, Markus Hecher, Daniel Le Berre, and Stefan Szeider. The silent (r)evolution of SAT. *Communications of the ACM*, 66(6):64–72, June 2023.

[Goedgebeur, 2020] Jan Goedgebeur. On minimal triangle-free 6-chromatic graphs. *J. Graph Theory*, 93(1):34–48, 2020.

[Heule *et al.*, 2018] Marijn J. H. Heule, Oliver Kullmann, and Armin Biere. Cube-and-conquer for satisfiability. In Youssef Hamadi and Lakhdar Sais, editors, *Handbook of Parallel Constraint Reasoning*, pages 31–59. Springer, 2018.

[Itzhakov and Codish, 2015] Avraham Itzhakov and Michael Codish. Breaking symmetries in graph search with canonizing sets. *CoRR*, abs/1511.08205, 2015.

[Janota and Marques-Silva, 2011] Mikolás Janota and João P. Marques-Silva. Abstraction-based algorithm for 2QBF. In Karem A. Sakallah and Laurent Simon, editors, *Theory and Applications of Satisfiability Testing - SAT 2011*, volume 6695 of *Lecture Notes in Computer Science*, pages 230–244. Springer Verlag, 2011.

[Janota *et al.*, 2012] Mikolás Janota, William Klieber, João Marques-Silva, and Edmund M. Clarke. Solving QBF with counterexample guided refinement. In *Theory and Applications of Satisfiability Testing - SAT 2012*, volume 7317 of *Lecture Notes in Computer Science*, pages 114–128. Springer Verlag, 2012.

[Jensen and Toft, 1995] Tommy R. Jensen and Bjarne Toft. *Graph coloring problems*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, Ltd, 1995.

[Karp, 1972] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972)*, pages 85–103. Plenum, New York, 1972.

[Kirchweger and Szeider, 2021] Markus Kirchweger and Stefan Szeider. SAT modulo symmetries for graph generation. In *27th International Conference on Principles and Practice of Constraint Programming (CP 2021)*, LIPIcs, page 39:1–39:17. Dagstuhl, 2021.

[Kirchweger *et al.*, 2022] Markus Kirchweger, Manfred Scheucher, and Stefan Szeider. A SAT attack on Rota's Basis Conjecture. In *Theory and Applications of Satisfiability Testing - SAT 2022 - 25th International Conference, Haifa, Israel, August 2-5, 2022, Proceedings*, 2022.

[Kirchweger *et al.*, 2023a] Markus Kirchweger, Tomáš Peitl, and Stefan Szeider. A SAT solver's opinion on the Erdős-Faber-Lovász conjecture. In Meena Mahajan and Friedrich Slivovsky, editors, *The 26th International Conference on Theory and Applications of Satisfiability Testing (SAT 2023), July 04-08, 2023, Alghero, Italy*, LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. To appear.

[Kirchweger *et al.*, 2023b] Markus Kirchweger, Manfred Scheucher, and Stefan Szeider. SAT-based generation of planar graphs. In Meena Mahajan and Friedrich Slivovsky, editors, *The 26th International Conference on Theory and Applications of Satisfiability Testing (SAT 2023), July 04-08, 2023, Alghero, Italy*, LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. To appear.

[Kochen and Specker, 1967] Simon Kochen and Ernst Specker. The problem of hidden variables in quantum mechanics. *J. Math. Mech.*, 17(1):59–87, 1967.

[Li *et al.*, 2022] Zhengyu Li, Curtis Bright, and Vijay Ganesh. A SAT solver + computer algebra attack on the minimum Kochen–Specker problem. Technical report, School of Computer Science at the University of Windsor, November 2022. https://cbright.myweb.cs.uwindsor.ca/reports/nmi-ks-preprint.pdf.

[Marques-Silva *et al.*, 2009] João P. Marques-Silva, Inês Lynce, and Sharad Malik. Conflict-driven clause learning SAT solvers. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, pages 131–153. IOS Press, 2009.

[McCreesh *et al.*, 2020] Ciaran McCreesh, Patrick Prosser, and James Trimble. The glasgow subgraph solver: Using constraint programming to tackle hard subgraph isomorphism problem variants. In Fabio Gadducci and Timo Kehrer, editors, *Graph Transformation - 13th International Conference, ICGT 2020, Held as Part of STAF 2020, Bergen, Norway, June 25-26, 2020, Proceedings*, volume 12150 of *Lecture Notes in Computer Science*, pages 316–324. Springer Verlag, 2020.

[McKay and Piperno, 2014] Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, II. *J. Symbolic Comput.*, 60:94–112, 2014.

[McKay, 1998] Brendan D. McKay. Isomorph-free exhaustive generation. *J. Algorithms*, 26(2):306–324, 1998.

[Ohrimenko *et al.*, 2009] Olga Ohrimenko, Peter J. Stuckey, and Michael Codish. Propagation via lazy clause generation. *Constraints*, 14(3):357–391, September 2009.

[Östergård, 2014] Patric R. J. Östergård. On the minimum size of 4-uniform hypergraphs without property B. *Discr. Appl. Math.*, 163:199–204, 2014.

[Pavičić *et al.*, 2005] Mladen Pavičić, Jean-Pierre Merlet, Brendan McKay, and Norman D Megill. Kochen–Specker vectors. *J. Phys. A Math. Theor.*, 38(7):1577, 2005.

[Peres, 1991] A Peres. Two simple proofs of the Kochen-Specker theorem. *J. Phys. A Math. Theor.*, 24(4):L175, feb 1991.

[Prestwich, 2009] Steven David Prestwich. Cnf encodings. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, pages 75–97. IOS Press, 2009.

[Sinz, 2005] Carsten Sinz. Towards an optimal cnf encoding of boolean cardinality constraints. In Peter van Beek, editor, *Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings*, volume 3709 of *Lecture Notes in Computer Science*, pages 827–831. Springer Verlag, 2005.

[Uijlen and Westerbaan, 2016] Sander Uijlen and Bas Westerbaan. A Kochen-Specker system has at least 22 vectors. *New Gener. Comput.*, 34(1-2):3–23, 2016.

[Wetzler *et al.*, 2014] Nathan Wetzler, Marijn J. H. Heule, and Warren A. Hunt. DRAT-trim: Efficient checking and trimming using expressive clausal proofs. In *Theory and Applications of Satisfiability Testing – SAT 2014*, volume 8561 of *Lecture Notes in Computer Science*, pages 422–429. Springer Verlag, 2014.

[Zulkoski *et al.*, 2017] Edward Zulkoski, Curtis Bright, Albert Heinle, Ilias S. Kotsireas, Krzysztof Czarnecki, and Vijay Ganesh. Combining SAT solvers with computer algebra systems to verify combinatorial conjectures. *J. Autom. Reason.*, 58(3):313–339, 2017.