A SAT Attack on Rota's Basis Conjecture

Markus Kirchweger
Algorithms and Complexity Group, TU Wien, Autria

Manfred Scheucher
Institut für Mathematik, Technische Universität Berlin, Germany

Stefan Szeider
Algorithms and Complexity Group, TU Wien, Autria

Abstract

The SAT modulo Symmetries (SMS) is a recently introduced framework for dynamic symmetry breaking in SAT instances. It combines a CDCL SAT solver with an external lexicographic minimality checking algorithm.

We extend SMS from graphs to matroids and use it to progress on Rota's Basis Conjecture (1989), which states that one can always decompose a collection of r disjoint bases of a rank r matroid into r disjoint rainbow bases. Through SMS, we establish that the conjecture holds for all matroids of rank 4 and certain special cases of matroids of rank 5. Furthermore, we extend SMS with the facility to produce DRAT proofs. External tools can then be used to verify the validity of additional axioms produced by the lexicographic minimality check.

As a byproduct, we have utilized our framework to enumerate matroids modulo isomorphism and to support the investigation of various other problems on matroids.

2012 ACM Subject Classification Mathematics of computing → Matroids and greedoids; Mathematics of computing → Solvers; Hardware → Theorem proving and SAT solving

Keywords and phrases SAT modulo Symmetry (SMS), dynamic symmetry breaking, Rota's basis conjecture, matroid

Digital Object Identifier 10.4230/LIPIcs.SAT.2022.4

Supplementary Material *Software (Source Code)*: `https://doi.org/10.5281/zenodo.6616343` [25]
Dataset (Data for Rank 4): `https://doi.org/10.5281/zenodo.6616373` [24]

Funding M.K. and S.S. were supported by the Austrian Science Fund (FWF), project P32441, and from the Vienna Science and Technology Fund (WWTF), project ICT19-065. M.S. was supported by the DFG Grant SCHE 2214/1-1.

1 Introduction

Over the last years, SAT (solving propositional satisfiability) and CP (constraint programming) have emerged as powerful tools for finding small (counter)examples for problems from various branches of discrete mathematics, such as extremal combinatorics, graph theory, or combinatorial geometry; see, e.g., [3, 9, 12, 16, 21, 27, 28, 29, 32, 39, 40, 43, 44]. The main task is to find a combinatorial object with specific properties or determine that such an object does not exist. Since the search space typically grows extremely fast, symmetry-breaking techniques are essential in that context. Symmetry breaking tries to avoid considering several symmetric copies of the same object. The two main approaches to symmetry breaking in the context of SAT and CP are

1. static symmetry breaking, where the encoding of the desired property is enhanced by additional constraints that break the symmetry (e.g., [8, 11]), and
2. dynamic symmetry breaking, where symmetries are broken dynamically during the solver's run.

Some of the main methods that use dynamic symmetry handling are *CDCLSym* [31], *SMS* [26] (short for *SAT modulo Symmetry*), and adding symmetric versions of learned clauses to the problem (e.g., [10, 38]). All these methods have in common that they tightly integrate the dynamic symmetry breaking into a CDCL SAT solver.

SMS considers symmetries of the encoded object (a combinatorial structure) and does not precompute its symmetries. Instead, an external minimality check algorithm searches for symmetries during the CDCL solver's run. The minimality check is invoked whenever the solver decides on an element of the encoded object (e.g., whether an edge is present in a graph or not).

For SMS, we must create different minimality check algorithms for different combinatorial objects. However, this apparent drawback is also one of its strengths: SMS benefits from an efficient minimality check tailored to the combinatorial objects under consideration. Additionally, SMS supports handling a vast amount of symmetries while approaches, which need to compute all symmetries explicitly, cannot.

In this article, we extend the SMS approach from graphs to *matroids*, proposing a minimality check tailored to matroids. This allows us to search for matroids with specified properties. We exemplify the use of the extended SMS to confirm the well-known *Rota's Basis Conjecture (RBC)* from matroid theory. While Gian-Carlo Rota originally stated the conjecture in terms of vector spaces [23, Conjecture 4], it is today known in its more general form for matroids [35]. In addition, we use SMS to enumerate all matroids up to a given rank and number of elements.

1.1 Rota's Basis Conjecture

Rota's Basis Conjecture (RBC) asserts that, for every matroid of rank r with r disjoint bases $B_1, \dots, B_r$, there exist r disjoint *rainbow* bases $B'_1, \dots, B'_r$, i.e., $|B_i \cap B'_j| = 1$ holds for all $i, j \in [r]$. We give a formal definition of matroids and further terminology in Section 2.1; Figure 1 shows an example for rank 3. Even though RBC attracted the attention of many researchers, it has only been proven for up to rank 3. While the proofs for ranks 1 and 2 are trivial, Chan [6] used an elaborate case distinction for her proof, which does not directly generalize to higher ranks.

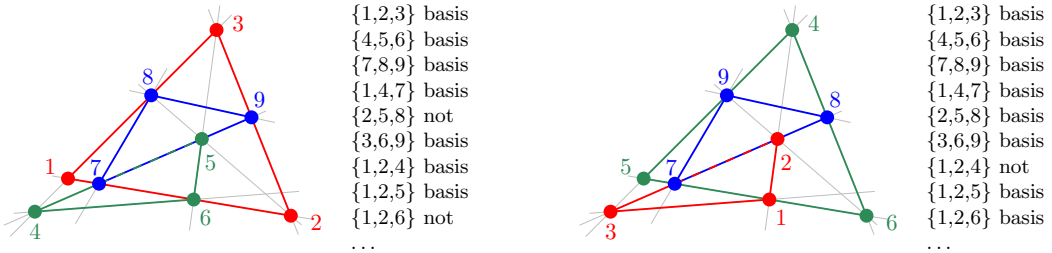

Figure 1 Two illustrations of RBC in rank 3. The elements of $B_1 = \{1, 2, 3\}$, $B_2 = \{4, 5, 6\}$, and $B_3 = \{7, 8, 9\}$ are highlighted in red, green, and blue, respectively. In this visualisation, three elements form a basis if and only if the corresponding points don't lie on a common line. In the left instance, $\{1, 4, 7\}, \{2, 5, 9\}, \{3, 6, 8\}$ are three disjoint rainbow bases. In the right instance, $\{1, 4, 7\}, \{2, 5, 8\}, \{3, 6, 9\}$ are three disjoint rainbow bases.

However, for higher ranks, RBC appears to be very difficult to confirm. Therefore, weaker versions of the conjecture and restrictions to certain subclasses of matroids have been investigated. The so-called paving matroids form an interesting subclass, for which

Geelen and Humphries [19] confirmed the conjecture by an inductive argument. Very recently, Friedman and McGuinness [15] extended the result by Geelen and Humphries to matroids with large girth. They showed that for any rank r matroid with girth $\geq r - o(\sqrt{r})$ and r disjoint bases given, one can find $r - o(\sqrt{r})$ disjoint rainbow bases. We will review some further related work in Section 1.3.

1.2 Our Contribution

We extend the SMS framework from graphs to general combinatorial structures and produce DRAT proofs, which then allows us to verify the correctness of the obtained unsatisfiability results by an independent tool such as DRAT-trim [42]. Using the extended SMS framework, we then attack RBC on matroids of small ranks. Our main result is a formal proof of the conjecture for matroids of up to rank 4 and for rank 5 matroids with girth ≥ 4. Even though we could not yet settle rank 5 entirely, we have further partial results for girth 2 and 3, and we are optimistic that we can settle the entire rank 5 case in the future. All our source code is available at [25] and the data for rank 4 (about 72 MB) is available at [24].

Moreover, we utilize our framework to enumerate all matroids modulo isomorphism for a given number of elements and rank and confirm the number of matroids given in the literature [1, 14, 30]; see Section 5.2.

In Section 3, we discuss a general dynamic symmetry-breaking strategy for object symmetries that we integrated into our SMS framework. Since the SAT encoding to verify RBC for matroids (presented in Section 4.2) has many symmetries, the dynamic symmetry breaking plays a central role in our approach.

To produce a DRAT proof, we perform the following steps: First, we run the solver Clingo [17, 18] extended with SMS and store additionally generated clauses for the symmetry breaking during solving. When Clingo concludes unsatisfiability, it exports a CNF instance containing the selection of constraints used to conclude unsatisfiability, including symmetry-breaking clauses obtained during the solving. The next step is generating a DRAT proof. We utilize the modern SAT solver CaDiCaL [4] to verify the unsatisfiability of the small instance produced by SMS and generate a DRAT proof. This DRAT proof can then be verified, for example, with DRAT-trim [42].

We provide a Python script to verify the correctness of the CNF instance generated by SMS, i.e., that all clauses are, in fact, clauses of the original instance. This provides additional certainty and allows independent third parties to verify the correctness of our computations by only checking this verification tool. In particular, the correctness of the results can be verified without inspecting our complex C-code.

1.3 Related Work

In 2012, Cheung [7] announced a computer proof of RBC for rank 3 matroids in an unpublished manuscript and that the program verifies the conjecture for rank 4. However, he did not provide specific details, and the source code appears to be unavailable today[1].

In recent years, significant steps towards an asymptotic proof of RBC have been made. For the setting where the number of bases is relaxed, Bucić et al. [5] showed that, for any r disjoint bases given, there are at least $\frac{r}{2}(1 - o(1))$ disjoint rainbow bases. This improved an earlier result by Geelen and Webb [20], who showed the existence of $\sqrt{r}(1 - o(1))$ disjoint

[1] Personal communication with Joshua E. Ducey, the supervisor of Michael Cheung.

rainbow bases. For the setting, where bases are relaxed to independent sets, Pokrovskiy [37] proved that, for any r disjoint bases $B_1, \ldots, B_r$ given, there exist $s = r(1 - o(1))$ disjoint *rainbow* independent sets $I_1, \ldots, I_s$ of cardinality $r(1 - o(1))$, i.e., $|B_i \cap I_j| \leq 1$ holds for all $i \in [r]$ and $j \in [s]$.

There is also an interesting relation between orthogonal Latin squares and rainbow bases. Onn showed that the Latin Square Conjecture by Alon and Tarsi [2] implies RBC for matroids originating from vector spaces of certain characteristics [34, Corollary 1]. The Alon–Tarsi Conjecture has been proven for all $(p+1) \times (p+1)$ Latin squares where p is a prime number [13]. In particular, RBC holds for matroids of rank $p+1$ representable over $\mathbb{R}$.

2 Preliminaries

For any positive integer n, we write $[n] = \{1, 2, \ldots, n\}$, we denote the power set of a finite set S by 2^S and the set of all k-subsets of S by $\binom{S}{k}$.

2.1 Matroids

Matroids are a well-studied and classical combinatorial structure that generalizes various concepts and notions from linear algebra, graph theory, geometry, combinatorial optimization, and other fields of mathematics in a natural way.

A *matroid* M is a pair $(E, \mathcal{I})$ where E is a finite set and $\mathcal{I} \subseteq 2^E$ fulfills the following three properties: (i) $\emptyset \in \mathcal{I}$, (ii) $\mathcal{I}$ is closed under subsets, and (iii) for $A, B \in \mathcal{I}$ and $|A| < |B|$ there exists $x \in B \setminus A$ such that $A \cup \{x\} \in \mathcal{I}$. Here, $E = E(M)$ is the *ground set* of M, and $\mathcal{I} = \mathcal{I}(M)$ is the set of *independent sets* of M. Property (iii) is called the *independent set exchange property*. The element in a one-element dependent set is called a *loop*, and the two elements in a two-element dependent set are called *parallel elements*.

An inclusion-wise maximal element of $\mathcal{I}(M)$ is called *basis*, and we denote the set of all bases by $\mathcal{B}(M)$. It is well known that all bases of a matroid M have the same cardinality $r = r(M)$, the *rank* of M. There are several well-studied cryptomorphic axiom systems for matroids; in particular, the set of bases fully characterizes a matroid.

▶ **Fact 1** (Basis exchange property (BEP)). *Let M be a matroid. For any two distinct bases B_1, B_2 of M and $e_1 \in B_1 \setminus B_2$ there exists an element $e_2 \in B_2 \setminus B_1$ such that $(B_1 \setminus \{e_1\}) \cup \{e_2\}$ is a basis of M.*

Two matroids M_1, M_2 are *isomorphic* if there is a bijection $\pi : E(M_1) \to E(M_2)$ such that $\pi(\mathcal{I}(M_1)) = \mathcal{I}(M_2)$, where $\pi(\mathcal{I}(M_1)) := \{\, \pi(I) \mid I \in \mathcal{I}(M_1) \,\}$. The *girth* $g = g(M)$ of a matroid M is the largest integer g such that every set $S \subseteq E(M)$ with $|S| < g$ is an independent set of M. Equivalently, the girth is the size of a smallest dependent set, i.e., the minimum cardinality among the sets in $2^{E(M)} \setminus \mathcal{I}(M)$. A matroid of rank r is called *paving* if it has girth $\geq r - 1$ (i.e., any $r - 1$ elements are contained in a basis) and *simple* if it has girth ≥ 3 (i.e., it does not contain loops or parallel elements). For further information on matroids, we refer the interested reader to Oxley's standard textbook [36].

2.2 Formulas and Satisfiability

A *literal* is a propositional variable or its negation. A *clause* is a disjunction of literals. A formula in *conjunctive normal form (CNF)* is a conjunction of clauses. The set of variables of a formula F is denoted by $var(F)$. A *partial assignment* is a function $\alpha : X \to \{0, 1\}$ defined on a subset $X \subseteq var(F)$. A partial assignment which assigns all propositional variables (i.e.,

$X = var(F)$) is called *assignment*. For a variable $x \notin X$ we say that α is *undefined* and write $\alpha(x) = \star$. Assignments extend to literals in the obvious way. A *model* of a CNF formula F is an assignment α defined on the variables of F such that each clause of F contains a literal that is set to true by α. F is *satisfiable* if it has a model; otherwise, it is *unsatisfiable*.

3 Dynamic Symmetry Breaking

In this section, we present a dynamic symmetry-breaking technique for object symmetries. As we will see in Section 4.2, our encoding for the considered problem has many object symmetries; hence symmetry breaking plays a central role in our investigations.

Before we define object symmetries and present our symmetry breaking for RBC, we recall variable symmetries in general. A *variable symmetry* of a propositional formula F is a permutation $\pi : var(F) \to var(F)$ such that α is a model of F iff $\alpha \circ \pi$ is a model of F. The results of this section are not limited to any specific formula F.

A well-known method for breaking variable symmetries is the LexLeader method [41]: Let $v_1 \prec \ldots \prec v_n$ be a total order of the variables of a formula F. Let $\alpha : var(F) \to \{0, 1\}$ be an assignment of F and π a variable symmetry. The LexLeader method adds clauses to the formula (so-called *symmetry breaking constraints*), such that, if $\alpha(v_1), \alpha(v_2), \ldots, \alpha(v_n)$ is lexicographically larger than $\alpha(\pi(v_1)), \alpha(\pi(v_2)), \ldots, \alpha(\pi(v_n))$, then the additional clauses guarantee that α is not a model of F. In other words, we only keep lexicographically minimal models. Two disadvantages of this approach are that typically new variables have to be introduced for the symmetry breaking constraints, and the initial size of the formula can increase significantly, especially if the number of variable symmetries is large.

CDCLSym [31] is a general dynamic symmetry-breaking framework based on the LexLeader method, which adds symmetry-breaking clauses during the solving process depending on the current partial assignment. This overcomes the issue of the large initial encoding size. Unfortunately, it is not suitable for a large number of variable symmetries because it internally handles every symmetry separately.

Kirchweger and Szeider [26] introduced SMS for searching for graphs with a specified property. SMS is based on the lexicographic order but considers only a subset of the variables. The main advantage of SMS is that it does not handle every symmetry separately and exploits the structure of the underlying combinatorial object. Whenever a variable is assigned, SMS searches for relevant symmetries by a branch-and-bound algorithm, i.e., for symmetries that give rise to clauses that are false (or unit clauses) under the partial assignment. It adds these clauses as learned clauses to the solver.

Here we want to generalize the ideas from Kirchweger and Szeider [26]. Therefore, we introduce so-called object symmetries to generalize the notion of variable symmetries. Let F be a propositional formula and let $V \subseteq var(F)$ be a fixed subset of the variables, called *object variables*. A natural choice for object variables are the variables that encode the combinatorial object under investigation. The other variables, which we call *auxiliary variables*, only play a lesser role, for example, for testing certain properties of the combinatorial object.

An *object symmetry* of a propositional formula F with the set V of object variables is a permutation $\pi : V \to V$ with the property that, for every model α of F, there exists a model α' such that $\alpha'(x) = \alpha(\pi(x))$ for every $x \in V$. We denote the set of all object symmetries by $Sym(F, V)$. Note that in the case $V = var(F)$, the object symmetries are precisely F's variable symmetries.

In Section 3.1, we generalize the SMS framework to arbitrary propositional formulas and object symmetries, formalize the meaning of relevant symmetries, describe the type of added clauses, and prove that the additional clauses do not discard lexicographically minimal solutions, i.e., the correctness of the symmetry breaking.

3.1 General Symmetry Breaking Predicates

To state the central properties of our symmetry breaking technique for object symmetries with appropriate terminology, we need to introduce some further notation.

Let $\mathcal{F}_n$ be the set of all pairs (F, V) where F is a propositional formula and $V = \{v_1, \ldots, v_n\} \subseteq var(F)$ is the set of object variables. For $(F, V) \in \mathcal{F}_n$ and two assignments α_1, α_2 of $var(F)$, we say that α_1 *is lexicographically smaller than* α_2 *with respect to* V (and write $\alpha_1 \prec_V \alpha_2$) if there exists an $i \in [n]$ such that (i) $\alpha_1(v_i) = 0$ and $\alpha_2(v_i) = 1$, and (ii) $\alpha_1(v_j) = \alpha_2(v_j)$ for all $j \in [i-1]$. Pause to note that $\prec_V$ is a partial order on the assignments of $var(F)$. In the case $V = var(F)$ it is a total order and coincides with the lexicographic order. An assignment α of a formula F is $\preceq_V$*-minimal* if $\alpha \preceq_V \alpha \circ \pi$ for all $\pi \in Sym(F, V)$.

Let $\alpha : X \subseteq var(F) \to \{0, 1\}$ be a partial assignment. An *extension* of α is an assignment $\alpha' : var(F) \to \{0, 1\}$ with $\alpha(x) = \alpha'(x)$ for all $x \in X$. We denote the set of all extensions of α by $\mathcal{X}(\alpha)$ and call α $\preceq_V$*-minimal* if $\mathcal{X}(\alpha)$ contains a $\preceq_V$*-minimal* assignment. A permutation $\pi \in Sym(F, V)$ is a *witness* of the non-$\preceq_V$-minimality of α if $\alpha' \circ \pi \prec \alpha'$ for all $\alpha' \in \mathcal{X}(\alpha)$.

For checking whether a partial assignment is $\preceq_V$-minimal, the notion of criticality and indicator indices will be of crucial importance. For a partial assignment $\alpha : X \to \{0, 1\}$, an integer $i \in [n]$ is (α, π)*-critical* if $(\alpha(v_i), \alpha(\pi(v_i))) \in \{(1, 0), (\star, 0), (1, \star)\}$. Moreover, $i \in [n]$ is an (α, π)*-indicator index* if i is (α, π)-critical and for every $j \in [i-1]$ at least one of the three cases holds: (i) $\pi(v_i) = v_i$, (ii) $\alpha(v_i) = 1$, or (iii) $\alpha(\pi(v_i)) = 0$. An (α, π)-indicator index i is *strict* if $\alpha(v_i) = 1$ and $\alpha(\pi(v_i)) = 0$. A partial assignment α is *constraining* if there is an (α, π)-indicator index for some $\pi \in Sym(F, V)$.

We are now ready to state the central proposition on object symmetries, which will allow us to reduce the search space for the SAT solver significantly. We will state how the symmetry-breaking clauses are derived during the solver's run.

▶ **Proposition 1.** *Let $(F, V) \in \mathcal{F}_n$ and let α be an assignment of $var(F)$. If there is a strict (α, π)-indicator index for some $\pi \in Sym(F, V)$ then α is not $\preceq_V$-minimal. Furthermore, if α is not $\preceq_V$-minimal then there is a strict (α, π)-indicator index for some $\pi \in Sym(F, V)$.*

Proof. Let α be a partial assignment and let i be a strict (α, π)-indicator index. To prove the first part of the statement, suppose, towards a contradiction, that α is $\preceq_V$-minimal. By definition, there is an assignment $\alpha' \in \mathcal{X}(\alpha)$ which is $\preceq_V$-minimal. First, we show by induction on $j \in [i-1]$ that $\alpha'(v_j) = \alpha'(\pi(v_j))$. We distinguish the following three cases.

1. If $\pi(v_j) = v_j$ then $\alpha'(v_j) = \alpha'(\pi(v_j))$ obviously holds.
2. If $\alpha(v_j) = 1$ then $\alpha'(v_j) = 1$. By induction hypothesis, $\alpha'(v_k) = \alpha'(\pi(v_k))$ holds for all $k \in [j-1]$. Hence also $\alpha'(\pi(v_j)) = 1$ must hold, otherwise α' is not $\preceq_V$ -minimal, so $\alpha'(v_j) = \alpha'(\pi(v_j))$.
3. If $\alpha(\pi(v_j)) = 0$ then $\alpha'(\pi(v_j)) = 0$. Again, by induction hypothesis, $\alpha'(v_k) = \alpha'(\pi(v_k))$ holds for all $k \in [j-1]$. Hence also $\alpha'(v_j) = 1$ must hold, otherwise α' is not $\preceq_V$ -minimal, so again $\alpha'(v_j) = \alpha'(\pi(v_j))$.

Altogether, we know that $\alpha'(v_j) = \alpha'(\pi(v_j))$ for all $j \in [i-1]$ and therefore α' cannot be $\preceq_V$-minimal. Consequently, α cannot be $\preceq_V$-minimal. This contradicts our initial assumption, and the first part of the statement follows.

For the second part, let α be a non-$\preceq_V$-minimal assignment. Then, by definition of $\preceq_V$-minimality, there exists a permutation $\pi \in Sym(F, V)$ such that $\alpha \circ \pi \prec_V \alpha$. By definition, there exists an index i such that $\alpha(v_i) = 1$ and $\alpha(\pi(v_i)) = 0$, and $\alpha(v_j) = \alpha(\pi(v_j))$ for all $j \in [i-1]$, hence either $\alpha(\pi(v_i)) = 0$ or $\alpha(v_i) = 1$, so i is a strict (α, π)-indicator index. ◀

▶ **Observation 2.** *Let α be a partial assignment, $\alpha' \in \mathcal{X}(\alpha)$ a $\preceq_V$-minimal assignment, and i an (α, π)-indicator index for some $\pi \in Sym(F, V)$. Then $\alpha'(v_i) = \alpha'(\pi(v_i))$.*

From Observation 2 we conclude that, if i is an (α, π)-indicator index and $\alpha(v_i) = 1$, then $\alpha'(\pi(v_i)) = 1$ for every $\preceq_V$-minimal assignment $\alpha' \in \mathcal{X}(\alpha)$. Moreover, if $\alpha(\pi(v_i)) = 0$ then we have $\alpha'(v_i) = 0$.

Next, we present how to extract a suitable clause from a partial assignment α and an (α, π)-indicator index i to avoid non-$\preceq_V$-minimal partial assignments. These clauses are added whenever the procedure finds an (α, π)-indicator index.

Let $S_1 = \{\, j \in [i-1] \mid \alpha(v_j) = 1\,,\, \pi(v_j) \neq v_j \,\}$ and let $S_2 = \{\, j \in [i-1] \mid \alpha(\pi(v_j)) = 0\,,\, \pi(v_j) \neq v_j \,\}$. Then

$$\neg v_i \vee \pi(v_i) \vee \bigvee_{j \in S_1} \neg v_j \bigvee_{j \in S_2} \pi(v_j)$$

is the resulting clause, which we add as learned clause. By Proposition 1, every $\preceq_V$-minimal partial assignment must satisfy this clause. Note that the clause is either false or a unit clause under the partial assignment α.

3.2 Integration into a CDCL-Solver

Having a procedure MinCheck to find indicator indices for any conflicting partial assignment (we stipulate the procedure tailored to our encoding in Section 4.5), we can integrate the dynamic symmetry breaking into a CDCL SAT solver as follows. Whenever the solver assigns an object variable, we use MinCheck to check whether the current partial assignment is conflicting. If the procedure returns an indicator index, we extract a clause from the partial assignment and the given index and add the new clause to the solver as a learned clause. In other words, the added clauses are part of the solver's clause-deletion policy and, therefore, can be discarded later. This allows us to add billions of clauses.

Instead of searching for indicator indices each time a variable is assigned, it is possible to do this only every f-th time for some $f \geq 1$. This allows us to balance the time spent searching indicator indices and the solving part.

4 Framework

In the following, we present a formula $F_{n,r}^{\text{matroid}}$ for all integers $n, r \geq 1$ whose models represent exactly the matroids with elements $[n]$ and rank r. The propositional variables of the formula represent the bases of the matroid, i.e., each variable indicates whether a particular r-subset T is a basis of the matroid or not. The clauses of the formulas ensure that the basis exchange property is fulfilled.

Building on $F_{n,r}^{\text{matroid}}$, we design a formula F_r^{Rota} that is satisfiable if and only if there is a counterexample to RBC for rank r. Additional clauses ensure that the modeled matroid does not contain r disjoint rainbow bases. Therefore, if a formula F_r^{Rota} was satisfiable, we could read a counterexample to the conjecture directly from the model, and vice versa. It will be crucial that, due to our symmetry breaking, only a small fraction of the clauses of the naive SAT encoding of F_r^{Rota} will be used to forbid rainbow bases.

In Sections 4.1 and 4.2, we will describe the encodings of $F_{n,r}^{\text{matroid}}$ and F_r^{Rota}, respectively. Later, we will refine our encoding of F_r^{Rota} by utilizing SMS (Section 4.4) and some lazy-encoding techniques to decrease the encoding size (Section 4.7). In Section 4.5, we present a minimality check for both formulas $F_{n,r}^{\text{matroid}}$ and F_r^{Rota} separately. Finally, we describe the verification of results generated by our framework in Section 4.8.

4.1 Encoding of Matroids

Let $\mathcal{M}_r^n$ be the set of all matroids M with $E(M) = [n]$ and rank r. A set $\mathcal{S} \subseteq \binom{[n]}{r}$ characterizes a matroid $M \in \mathcal{M}_r^n$ if $\mathcal{S}$ is nonempty and the basis exchange property holds, i.e., for any two distinct bases $B_1, B_2 \in \mathcal{S}$, there exist elements $e_1 \in B_1 \setminus B_2$ and $e_2 \in B_2 \setminus B_1$ such that $(B_1 \setminus \{e_1\}) \cup \{e_2\} \in \mathcal{S}$.

For each $T \in \binom{[n]}{r}$, we introduce a propositional variable p_T that is true if and only if $T \in \mathcal{S}$. Then, the encoding for $\mathcal{S}$ being nonempty is $\bigvee_{T \in \binom{[n]}{r}} p_T$, and the encoding for the basis exchange property is

$$\bigwedge_{B_1, B_2 \in \binom{[n]}{r} : B_1 \neq B_2} \; \bigwedge_{e_1 \in B_1 \setminus B_2} \left(\neg p_{B_1} \vee \neg p_{B_2} \vee \bigvee_{e_2 \in B_2 \setminus B_1} p_{(B_1 \setminus \{e_1\}) \cup \{e_2\}} \right).$$

4.2 Basic Encoding of RBC

Suppose that there exists a matroid M of rank r which is a counterexample to RBC, i.e., there are disjoint bases $R_1, \ldots, R_r \in \mathcal{B}(M)$ such that there are no r disjoint rainbow bases. We can assume without loss of generality that M has precisely r^2 elements, as otherwise we restrict M to $R = \bigcup_{i=1}^{r} R_i$ and obtain another matroid $M' = (E(M) \cap R, \{\, I \cap R \mid I \in \mathcal{I}(M) \,\})$ of rank r which is a counterexample to RBC. Moreover, since a matroid's elements can be relabeled arbitrarily, we may assume that $M = ([r^2], \mathcal{I})$ with $R_1, \ldots, R_r \in \mathcal{B}(M)$ and $R_i = \{(i-1)\cdot r+1, (i-1)\cdot r+2, \ldots, (i-1)\cdot r+r\}$. For example, if $r = 3$, then $R_1 = \{1,2,3\}$, $R_2 = \{4,5,6\}$, and $R_3 = \{7,8,9\}$. We denote by $\mathcal{M}_r^{\text{Rota}}$ the set of matroids in $\mathcal{M}_r^{r^2}$ that have $R_1, \ldots, R_r$ as bases but do not contain r disjoint rainbow bases. Thus $\mathcal{M}_r^{\text{Rota}} = \emptyset$ iff the RBC holds for all matroids of rank r.

A set $\mathcal{S} \subseteq \binom{[r^2]}{r}$ characterizes a matroid $M \in \mathcal{M}_r^{\text{Rota}}$ if $\mathcal{S}$ characterizes a matroid in $\mathcal{M}_r^{r^2}$ and the following two additional properties are satisfied:

(P1) $R_1, \ldots, R_r \in \mathcal{S}$.

(P2) Let $\mathcal{C}$ denote the set of all $\{C_1, \ldots, C_r\} \subseteq \binom{[r^2]}{r}$ such that $C_i \cap C_j = \emptyset$ for $i \neq j$ and $|C_i \cap R_j| = 1$ for all $i, j \in [r]$. Then for all $\{C_1, \ldots, C_r\} \in \mathcal{C}$ there exists $i \in [r]$ such that $C_i \notin \mathcal{S}$.

Note that Property (P2) ensures that the number of disjoint rainbow bases is less than r.

We encode these statements as clauses:

(P1') $\bigwedge_{i \in [r]} \; p_{R_i}$.

(P2') $\bigwedge_{\{C_1, \ldots, C_r\} \in \mathcal{C}} \; \neg p_{C_1} \vee \ldots \vee \neg p_{C_r}$.

Altogether, F_r^{Rota} consists of the clauses from $F_{r^2,r}^{\text{matroid}}$ plus the clauses (P1') and (P2').

Unfortunately, the size of the set $\mathcal{C}$ increases quickly for increasing r, and so does the number of clauses to encode property (P2). To avoid considering all possible $\{C_1, \ldots, C_r\} \in \mathcal{C}$, we apply the symmetry breaking presented in Section 3 in such a way that the clauses from property (P2) can be omitted entirely.

4.3 SMS to Enumerate Matroids modulo Isomorphism

For each matroid M with $E(M) = [n]$ and rank r, we have a model α_M of $F_{n,r}^{\text{matroid}}$ where $\alpha_M(p_T) = 1$ iff $T \in \mathcal{B}(M)$ and each model α of $F_{n,r}^{\text{matroid}}$ represents a unique matroid.

The set of object variables is $V = var(F_{n,r}^{\text{matroid}})$, and the set of object symmetries is

$$Sym(F_{n,r}^{\text{matroid}}, V) = \{\, \delta_\pi : V \to V,\, p_T \mapsto p_{\pi(T)} \mid \text{ permutation } \pi : [n] \to [n] \,\}. \tag{1}$$

By using SMS, only $\preceq_V$-minimal models are allowed. As a result, all models of $F^{\text{matroid}}_{n,r}$ given by SMS represent exactly the matroids in $\mathcal{M}^n_r$ modulo isomorphism. Note that this is not necessarily the case if we would use any auxiliary variables, i.e., there might be different $\preceq_V$-minimal models representing the same matroid.

4.4 Dynamic Symmetry Breaking and Rota's Basis Conjecture

Suppose that there exists some $M \in \mathcal{M}^{\text{Rota}}_r$, i.e., M is a counterexample to RBC for rank r. A *row permutation* is permutation $\pi : [r^2] \to [r^2]$ such that for all $i \in [r]$ there is some $j \in [r]$ such that $\pi(R_i) = R_j$. In other words, the permutation π maps all elements from a basis R_i to a basis of R_j where $i, j \in [r]$. If π is a row permutation then $\pi(M) := (E(M), \{\, \pi(I) \mid I \in \mathcal{I}(M) \,\})$ is also a counterexample to RBC for rank r, i.e., $\pi(M) \in \mathcal{M}^{\text{Rota}}_r$. Note that there are $(r!)^{r+1}$ row permutations in total. Also, note that, in general, we obtain distinct matroids $\pi(M) \neq \pi'(M)$ for distinct row permutations π, π', except if M has intrinsic symmetries.

We denote the set of all row permutations by $\mathcal{S}_r$ and consider the set of object variables $V = var(F^{\text{Rota}}_r)$. Then, the set of object symmetries for the formula F^{Rota}_r is given by

$$Sym(F^{\text{Rota}}_r, V) = \{\, \delta_\pi : V \to V,\, p_T \mapsto p_{\pi(T)} \mid \pi \in \mathcal{S}_r \,\}. \tag{2}$$

For applying the symmetry breaking technique from Section 3, we need to define a total order of the object variables V. Let $D_i = \{\, (j-1) \cdot r + i \mid j \in [r] \,\}$ for $i \in [r]$. For example, if $r = 3$, then $D_1 = \{1, 4, 7\}$, $D_2 = \{2, 5, 8\}$, and $D_3 = \{3, 6, 9\}$. To provide some intuition, it is worth noting that we can think of the elements $[r^2]$ as an $r \times r$ grid, where the R_i's are the rows, and the D_i's are the columns. The sets $D_1, \ldots, D_r$ are pairwise disjoint and, for every $i \in [r]$ and for every $j \in [r]$, the set D_i shares precisely one element with the set R_j. Hence, every element from $[r^2]$ occurs as the unique intersection of $D_i \cap R_j$ for some $i, j \in [r]$.

We define a total order $\prec$ on the object variables V via

$$p_{R_1} \prec \ldots \prec p_{R_r} \prec p_{D_1} \prec \ldots \prec p_{D_r} \prec o_1 \prec \ldots \prec o_m,$$

where $o_1, \ldots, o_m$ is a fixed labeling of the remaining variables from V.

For $M \in \mathcal{M}^{r^2}_r$ we have $\alpha_M(p_T) = 1$ if and only if $T \in \mathcal{B}(M)$. This gives us a total order on $\mathcal{M}^{r^2}_r$. For matroids $M_1, M_2 \in \mathcal{M}^{r^2}_r$ we write $M_1 \preceq M_2$ if $\alpha_{M_1} \preceq_V \alpha_{M_2}$.

As we will see, it is beneficial using the negated variables of V for the symmetry breaking in Section 3. So, lexicographically maximal partial assignments are not discarded. This will allow us to formulate Proposition 4 below, the main proposition for reducing the encoding size.

For a matroid $M \in \mathcal{M}^{r^2}_r$ we say that M is *$\preceq_V$-maximal* if α_M is $\preceq_V$-maximal.

▶ **Observation 3.** *Let $M \in \mathcal{M}^{\text{Rota}}_r$ be a $\preceq_V$-maximal matroid, then $D_r \notin \mathcal{B}(M)$.*

Proof. For the sake of contradiction, assume $M \in \mathcal{M}^{\text{Rota}}_r$ and $D_r \in \mathcal{B}(M)$. Since M is $\preceq_V$-minimal and $D_r \in \mathcal{B}(M)$, also $D_i \in \mathcal{B}(M)$ for $i \in [r-1]$ must hold. Hence, Property (P2) is violated, so M is not in $\mathcal{M}^{\text{Rota}}_r$, contradicting our assumptions. ◀

It follows by Observation 3 that F^{Rota}_r has the same $\preceq_V$-maximal models as $F^{\text{Rota}}_r \wedge \neg p_{D_r}$. Let G^{Rota}_r be a formula containing exactly the clauses of $F^{\text{Rota}}_r \wedge \neg p_{D_r}$ expect the one for encoding property (P2), i.e., for restricting the number of disjoint rainbow bases.

▶ **Proposition 4.** *Let M be a matroid with rank r with $E(M) = [r^2]$. Then, α_M is a $\preceq_V$-maximal model of G^{Rota}_r if and only if $M \in \mathcal{M}^{\text{Rota}}_r$.*

Proof. "⇒": For the sake of contradiction, assume that α_M is a $\preceq_V$-maximal model of G_r^{Rota} and $M \notin \mathcal{M}_r^{\text{Rota}}$, i.e., there are r disjoint rainbow bases $C_1, \dots, C_r$ of M. For every $x \in [r^2]$ we let $\pi(x) = i \cdot (r-1) + j$ where $i, j \in [r]$ are the unique indices such that x is the unique intersection point of $R_i \cap C_j$. Then $\pi(C_i) = D_i$ and $\pi(R_i) = R_i$ for all $i \in [r]$. Hence, $\alpha_M(p_{D_r}) = 1$ in contradiction to the assumption that α_M is a model of G_r^{Rota}.

"⇐": This direction follows immediately from Observation 3 and the fact that the set of clauses of G_r^{Rota} is a subset of $F_r^{\text{Rota}} \wedge \neg p_{D_r}$. ◀

Analogously, the following observation holds:

▶ **Observation 5.** *Let $G_{r,k}^{\text{Rota}} = G_r^{\text{Rota}} \wedge \bigwedge_{i\in[k]} p_{D_i} \bigwedge_{i\in[r]\setminus[k]} \neg p_{D_i}$ and $M \in \mathcal{M}_r^{r^2}$. Then, α_M is a $\preceq_V$-maximal model of $G_{r,k}^{\text{Rota}}$ if and only if k is the size of a largest set of disjoint rainbow bases.*

By Proposition 4, SMS does not return a model of G_r^{Rota} if and only if $\mathcal{M}_r^{\text{Rota}} = \emptyset$, since the symmetry breaking excludes matroids that are not $\preceq_V$-maximal by Proposition 1.

4.5 Minimality Check

Finally, we have to find witness permutations for a formula F and a partial assignment α. More precisely, for every partial assignment α, we want to decide whether α is constraining, i.e., there is some object symmetry $\delta \in \mathit{Sym}(F, V)$ and an (α, δ)-indicator index i.

Testing whether a (δ, α)-indicator index for a single symmetry $\delta \in \mathit{Sym}(F, V)$ exists is straightforward. We start with $i = 1$. If $(\alpha(\neg v_i), \alpha(\delta(\neg v_i))) \in \{(\star, 0), (1, \star), (1, 0)\}$ then i is an δ-indicator index. If $\alpha(v_i) = \alpha(\delta(v_i)) \neq \star$ or $v_i = \delta(v_i)$ then we increment i. If neither of the previous cases holds, then there is no (α, δ)-indicator index for our fixed δ. Since the number of object symmetries for our applications is huge, testing each object symmetry separately is not practicable. Instead, we use a branch-and-bound approach which gradually constructs a witness.

First, we present a minimality check for the formula $F_{n,r}^{\text{matroid}}$. Recall that the set of object variables is $V = \mathit{var}(F_{n,r}^{\text{matroid}})$, and the set of object symmetries are given in (1). We use the colexicographic order on $\binom{[n]}{r}$ for the symmetry breaking. Instead of directly constructing an object symmetry δ_π, we focus on the permutations $\pi : [n] \to [n]$.

Similarly to partial assignments, a partial permutation $\pi : X \to [n]$, for $X \subseteq [n]$, is an injective function. We write $\pi(x) = \star$ if $\pi(x)$ is undefined.

The idea is assigning only a few values to a partial permutation π. If the information about the partial permutation π suffices to conclude that δ_π is a witness, we can assign the remaining values arbitrarily; if we conclude that the partial permutation cannot have a corresponding indicator index, we backtrack; if we need more information about the partial permutation, we assign π for some further variables and branch over every possible assignment of the additional variables. Algorithm 1 realizes this idea, starting with $i = 1$ and $\pi(x) = \star$ for all $x \in [n]$.

▶ **Theorem 6.** *Let α be a partial assignment. If α is constraining, then Algorithm 1 returns a permutation π and an (α, π)-indicator index; otherwise, the algorithm returns nil.*

Proof. First, we prove that if the procedure returns a permutation δ_π and an index i, then i is an (α, δ_π)-indicator index. Assume the procedure returns a permutation δ_π and an index i. Line 5 guarantees that i is (α, δ_π)-critical. By Line 7, for all $i' \in [i-1]$ either $\alpha(p_{T_{i'}}) = \alpha(p_{\pi(T_{i'})}) \neq \star$ or $T_{i'} = \pi(T_{i'})$ holds, hence, at least one of the following three conditions holds: $\alpha(p_{T_{i'}}) = 1$, $\alpha(p_{\pi(T_{i'})}) = 0$, or $T_{i'} = \pi(T_{i'})$. Therefore, i is indeed an (α, δ_π)-indicator index.

Second, we prove that if α is not constraining, the procedure does not return *nil*. For the sake of contradiction, assume the procedure returns *nil* and α is constraining. Since we branch over all possible assignments and only backtrack if the partial permutation has no corresponding indicator index, we can conclude that there is no permutation δ_π with an (α, δ_π)-indicator index, hence α is not constraining in contradiction to our assumptions. ◀

Algorithm 1 Minimality check for enumerating matroids.

Input: A partial assignment α, a partial permutation π, and an index i
Output: A witness δ_π and a δ_π-indicator index or *nil*

1: **while** $i \leq$ number of r-subsets **do**
2: **for** all possible assignments of $\pi(x)$ for $x \in T_i$ with $\pi(x) = \star$ **do**
3: **if** MinCheck$(\alpha, \pi, i) \neq$ *nil* **then**
4: **return** MinCheck(α, π, i)
5: **if** $(\alpha(p_{T_i}), \alpha(p_{\pi(T_i)})) \in \{(1, \star), (1, 0), (\star, 0)\}$ **then**
6: **return** δ_π, i
7: **if** $\alpha(p_{T_i}) = \alpha(p_{\pi(T_i)}) \neq \star$ or $T_i = \pi(T_i)$ **then**
8: $i \leftarrow i + 1$
9: **continue**
10: **return** *nil*
11: **return** *nil*

Next, we focus on the minimality check for RBC. Recall the set of object symmetries from (2). Similarly to the minimality check for enumerating matroids, we construct the permutations $\pi \in \mathcal{S}_r$. Considering the elements of $[r^2]$ as an $r \times r$ grid, where the R_i's are the rows and the D_i's columns, it is easy to see that every permutation $\pi \in \mathcal{S}_r$ can be described as a permutation of the elements within the rows and a permutation of the rows itself. More precisely, every permutation $\pi \in \mathcal{S}_r$ can be described as a permutation $\pi' \in \mathcal{S}_r$ with $\pi'(R_i) = R_i$ for $i \in [r]$ and a permutation $\beta : [r] \to [r]$ of the rows.

We formalize this as follows. Let $\pi \in \mathcal{S}_r$ and $\beta : [r] \to [r]$ a permutation, then $\pi_\beta : x \mapsto \pi(x) - (\beta(i) - i) \cdot r$ for $x \in R_i$. We use the following two observations to get our final algorithm.

▶ **Observation 7.** *Let $\pi \in \mathcal{S}_r$. Then $\pi_\beta(D_i) = \pi(D_i)$ for all $i \in [r]$ and for all permutations $\beta : [r] \to [r]$.*

▶ **Observation 8.** *For each permutation $\pi \in \mathcal{S}_r$, there is some $\pi' \in \mathcal{S}_r$ with $\pi'(R_i) = R_i$ for all $i \in [r]$ and a permutation $\beta : [r] \to [r]$ such that $\pi = \pi'_\beta$.*

The minimality check for F_r^{Rota} is given by Algorithm 2. W.l.o.g., we assume that $\alpha(p_{R_i}) = 1$ for all $i \in [r]$. At the beginning, all values of the partial permutation π are undefined and $d = 1$. Then, we determine which values are mapped to D_i, with the additional restriction, if $x \in R_i$ then $\pi(x) \in R_i$ for all $i \in [r]$. If $(\alpha(\neg p_{D_i}), \alpha(\neg p_{\pi(D_i)})) \in \{(1, \star), (1, 0), (\star, 0)\}$ we have found an indicator index. If $\alpha(p_{D_i}) = \alpha(p_{\pi(D_i)}) \neq \star$ or $D_i = \pi(D_i)$, we can continue. Otherwise, we backtrack. If we reach Line 2, the partial permutation is fully defined and $\pi(R_i) = R_i$ for $i \in [r]$.

Again, instead of testing the solution for each β separately, we use a branch-and-bound approach and assign values to the permutation β if needed to get the truth value of $\pi(v)$ for some variable.

Algorithm 2 Minimality check for RBC.

Input: A partial assignment α, a partial permutation π, and a recursion depth d
Output: A witness δ_π and a δ_π-indicator index or *nil*

```
1: if d > r then
2:     for β permutation of [r] do                ▷ partial permutation π is fully defined
3:         if CheckSingleSymmetry(α, δ_{π_β}) ≠ nil then
4:             return CheckSingleSymmetry(α, δ_{π_β})
5:     return nil
6: for all possible assignments of π(x) for x ∈ D_d such that π(x) ∈ R_i if x ∈ R_i do
7:     if (α(¬p_{D_d}), α(¬p_{π(D_d)})) ∈ {(1, ⋆), (1, 0), (⋆, 0)} then
8:         return δ_π, d + r
9:     if α(p_{D_d}) = α(p_{π(D_d)}) ≠ ⋆ or D_d = π(D_d) then
10:        if MinCheck(α, π, d + 1) ≠ nil then
11:            return MinCheck(α, π, d + 1)
12: return nil
```

▶ **Theorem 9.** *Let α be a partial assignment of $var(F_r^{\mathrm{Rota}})$ with $\alpha(p_{R_i}) = 1$ for all $i \in [r]$. If α is constraining, then Algorithm 2 returns a permutation π and a (α, π)-indicator index; otherwise the algorithm returns nil.*

Proof. We proceed similarly to the proof of Theorem 6. First, we prove that if the procedure returns a permutation δ_π and an index i, that i is an (α, δ_π)-indicator index. Then, we prove that if α is not constraining, the procedure does not return *nil*.

- Assume the procedure returns an index i and a permutation δ_π. We distinguish the following two cases.
  1. $i > 2r$: In this case, the index was found by CheckSingleSymmetry, and therefore i is indeed an indicator index.
  2. $i \leq 2r$: This case is analog to the proof of Theorem 6.
- For the sake of contradiction, assume that the procedure returns *nil*, and there is an (α, δ_π)-indicator index j. By Observation 8, there are π' with $\pi'(R_i) = R_i$ and β such that $\pi'_\beta = \pi$. We distinguish the following two cases.
  1. $j \leq 2r$: We branch over all possible assignments such that $\pi''(R_i) = R_i$ for all $i \in [r]$ and only backtrack if the partial permutation has no corresponding indicator index. By Observation 7, the index j is an $(\alpha, \delta_{\pi'_\beta})$-indicator index if and only if it is an $(\alpha, \delta_{\pi'})$-indicator index for $j \leq 2r$. Since the procedure returns *nil*, the index j is not an $(\alpha, \delta_{\pi'})$-indicator index, contradicting to it being an $(\alpha, \delta_{\pi'_\beta})$-indicator index.
  2. $j > 2r$: By Observation 7, it follows that $\pi'_\beta(D_d) = \pi'(D_d)$ for $d \in [r]$. Therefore, Line 2 is reached with the permutation π', and so CheckSingleSymmetry$(\alpha, \delta_{\pi'_\beta})$ returns *nil*, contradicting that j is an $(\alpha, \delta_{\pi'_\beta})$-indicator index.

This completes the proof of Theorem 9. ◀

In practice, instead of terminating the algorithm immediately after finding an indicator index, we add the corresponding clause to the solver, backtrack in MinCheck if the index is not critical, and search for further indicator indices.

4.6 An Example for the Dynamic Symmetry Breaking for RBC

Let us consider the left-hand side example of RBC depicted in Figure 1. If we exchange the red and the green colors and relabel the points as $1 \mapsto 5, 2 \mapsto 6, 3 \mapsto 4, 4 \mapsto 3, 5 \mapsto 2, 6 \mapsto 1, 7 \mapsto 7, 8 \mapsto 9, 9 \mapsto 8$, we obtain precisely the right-hand side example from Figure 1. Table 1 summarises the truth values for the first variables ordered according to our order on the object variables given in Section 4.4.

Table 1 Comparison between original and permuted matroid.

T	{1,2,3}	{4,5,6}	{7,8,9}	{1,4,7}	{2,5,8}	{3,6,9}	...
$\alpha(p_T)$	1	1	1	1	0	1	...
$\pi(T)$	{5,6,4}	{3,2,1}	{7,9,8}	{5,3,7}	{6,2,9}	{4,1,8}	...
$\alpha(p_{\pi(T)})$	1	1	1	1	1	1	...

Since the first 4 variables coincide and the fifth variable differs, we see that the left-hand side example of Figure 1 is lexicographically smaller (w.r.t. to our order of the object variables) than the right-hand side example and therefore cannot be a lexicographically maximal matroid.

If we – at any time of the computations – obtain a (partial) matroid which has such a behavior, our dynamic symmetry breaking adds a clause that prevents that such a situation can occur again.

4.7 Lazy Encoding for Basis Exchange Property

The encoding size for the basis exchange property, described in Section 4.2, grows rapidly for increasing r. For example, for $r = 4$ the number of potential bases is 1640, while the number for $r = 5$ is 53130, which results in roughly $9.8 \cdot 10^6$ clauses for $r = 4$ and $1.1 \cdot 10^{10}$ for $r = 5$ for encoding the basis exchange property.

Instead of adding all the clauses at once, we only add a few clauses depending on the solver's current partial assignment α and remove clauses. More precisely, whenever a variable p_{T_1} is set to true during the solving process, we add the following clauses:

For all variables $p_{T_2} \in var(F_r^{\text{Rota}}) \setminus \{p_{T_1}\}$ with $\alpha(p_{T_2}) = 1$ and $e_1 \in T_1 \setminus T_2$, we add

$$\neg p_{T_1} \vee \neg p_{T_2} \bigvee_{e_2 \in T_2 \setminus T_1} p_{(T_1 \setminus \{e_1\}) \cup \{e_2\}}.$$

For all variables $p_{T_2} \in var(F_r^{\text{Rota}}) \setminus \{p_{T_1}\}$ with $\alpha(p_{T_2}) = 1$ and $e_2 \in T_2 \setminus T_1$, we add

$$\neg p_{T_2} \vee \neg p_{T_1} \bigvee_{e_1 \in T_1 \setminus T_2} p_{(T_2 \setminus \{e_2\}) \cup \{e_1\}}.$$

We don't add a clause if it is already satisfied.

If the solver removes the assignment of p_{T_1}, we remove all these clauses. Although the literals $\neg p_{T_1}$ and $\neg p_{T_2}$ are false under the current partial assignment, we do not omit the literals so that the clauses can be added to the solver without any concerns.

4.8 DRAT Proofs for RBC

To confirm our results, we produce DRAT proofs (short for Deletion Resolution Asymmetric Tautology), which are sequences of addition and deletion steps of RAT clauses [22].

In principle, SMS and the generation of a DRAT proof can be accomplished at once, within a single run of a SAT solver. To allow the use of two different solvers, one for SMS and one for DRAT proof generation, we proceed as follows. First, we generate clauses for showing unsatisfiability. More precisely, we store our initial encoding plus all used symmetry breaking clauses and clauses for encoding the basis exchange property. Then, we feed these clauses to a SAT solver supporting DRAT proofs.

The verification of the result consists of two parts. First, we verify our DRAT proof with DRAT-trim. Second, we verify that all dynamically added clauses for symmetry breaking and the basis exchange property are correct. Therefore, we store some additional data to each generated clause indicating the reason for the clause being added and store some information to ease the validation of the correctness of the clause.

The clauses for encoding the basis exchange property have the following form: $\neg p_{T_1} \vee \neg p_{T_2} \bigvee_{e_2 \in T_2 \setminus T_1} p_{(T_1 \setminus \{e_1\}) \cup \{e_2\}}$ where $e_1 \in T_1 \setminus T_2$. So, the two sets T_1, T_2 are implicitly given by the negated variables. Additionally, we store the element e_1 used for the specific clause. Given the sets T_1 and T_2 and an element e_1, we can check if all positive literals correspond to sets $(T_1 \setminus \{e_1\}) \cup \{e_2\}$ where $e_2 \in T_2 \setminus T_1$ and the clause indeed contains all positive literals.

For the symmetry-breaking clauses, the additional information is the permutation π of $[r^2]$. We order the literals of the clause in special way to ease the validation. Recall the form of symmetry breaking clauses from the end of Section 3.1:

$$\neg v_i \vee \pi(v_i) \vee \bigvee_{j \in S_1} \neg v_j \bigvee_{j \in S_2} \pi(v_j).$$

First, we add the literals of the form $\neg v_j$ or $\pi(v_j)$ ordered by j. At the end, we add the two literals $\neg v_i$ and $\pi(v_i)$, i.e., the last two literals correspond to the indicator index. This allows us to verify the clause in the following way, starting with $j = 1$ and $p = 1$, where p means that the first $p - 1$ literals in the clause are already checked: (i) if $\pi(v_j) = v_j$ we increase j by 1, (ii) if only two literals left then $i = j$,i.e., they must be $\neg v_j$ and $\pi(v_j)$, (iii) otherwise the literal on position p in the clause must be either $\neg v_j$ or $\pi(v_j)$ and we increase j and p by 1.

If cases (ii) and (iii) are not violated, we know that the clause has indeed the correct form. So, this procedure allows us two verify the symmetry breaking clauses given a permutation.

5 Experiments

In this section, we describe our experimental setup, some implementation details and report our results. As the SAT solver to handle the SMS procedure, we use Clingo [17, 18], an ASP solver containing a CDCL SAT solver. Clingo comes with a C-interface that supports rapid prototyping for developing custom propagators. We use this interface to integrate our implementation of MinCheck into the solver. Since Clingo does not support DRAT proofs, we use CaDiCaL [4] to produce a proof from the initial encoding and the clauses generated by SMS and the basis exchange property.

We use Clingo 5.5.0 and CaDiCaL version sc2020. All tests are executed with a single thread with at most 100 GB RAM.

5.1 Results for RBC

We report all running times in *CPU time* and give the sum of individual running times if the solving was conducted in several parts. With SMS, we can verify RBC for rank 4 in about 5 minutes. SMS produces 4154226 clauses in total. Then CaDiCaL takes 7 minutes to

produce a DRAT proof given the clauses. The time for verifying the clauses is 26 seconds, and DRAT-trim takes 4 minutes to check the DRAT proof. Furthermore, DRAT-trim provides an unsatisfiable core with only 137949 clauses. An unsatisfiable core is a subset of the clauses which is still unsatisfiable.

Since we cannot yet confirm RBC for all matroids for rank 5, we use an encoding to ensure that the girth is at least a certain value. This allows us to restrict the search space.

We encode that the girth must be at least g in the following way:

$$\bigwedge_{I \in \binom{[r^2]}{g-1}} \bigvee_{T \in \binom{[r^2]}{r}, I \subseteq T} p_T.$$

Note that each potential matroid $M \in \mathcal{M}_r^{\text{Rota}}$ has girth ≥ 2 because each element $e \in E(M)$ is in R_i for some $i \in [r]$. Furthermore, Observation 5 allows us to search for matroids in $\mathcal{M}_r^{\text{Rota}}$ with exactly k disjoint rainbow bases by solving $G_{r,k}^{\text{Rota}}$ with SMS. If SMS concludes unsatisfiability for all $k \in [r-1]$, then we know that there are always r disjoint rainbow bases; hence RBC holds for the given rank.

Table 2 summarises our results for rank 5. For $(g,k) \in \{(2,3), (2,4), (3,4)\}$, the solver does not terminate after several days. For all other configurations, SMS concludes unsatisfiability. We provide the time spent in Clingo and CaDiCaL, respectively, the number of generated clauses and the number of clauses of an unsatisfiable core given by the DRAT-trim. For $(g,k) \in \{(3,3), (4,4)\}$, SMS concludes unsatisfiability, but we have not generated or verified the proof yet.

Table 2 Results for rank 5. The girth is given by g and the exact number of disjoint rainbow bases by k. Each table entry consists of two lines. The first line has the format X/Y where X is the solving time for SMS and Y is the time used for producing the DRAT-proof. The second line has the format A/B where A gives the number of clauses of an unsatisfiable core and B gives the number of generated clauses.

	$k=1$	$k=2$	$k=3$	$k=4$
$g \geq 2$	13 h / 7 s $6.2 \cdot 10^3$/ $5.4 \cdot 10^6$	5h / 5min $8.2 \cdot 10^4$/ $1.1 \cdot 10^8$	t.o.	t.o.
$g \geq 3$	9h / 6s $2.8 \cdot 10^4$/ $4.2 \cdot 10^6$	6h / 5min $7.9 \cdot 10^4$/ $9.9 \cdot 10^7$	248h –	t.o.
$g \geq 4$	15min / 15s $7.1 \cdot 10^2$/ $2.0 \cdot 10^7$	20min / 8s $1.7 \cdot 10^4$/ $6.4 \cdot 10^6$	2h / 8min $2.9 \cdot 10^5$/ $3.7 \cdot 10^7$	208h –
$g \geq 5$	14min / 1s $4.7 \cdot 10^1$/ $2.1 \cdot 10^6$	1min / 1s $1.1 \cdot 10^2$/ $2.4 \cdot 10^6$	48s / 1s $8.4 \cdot 10^2$/ $2.2 \cdot 10^6$	1h / 1min $3.5 \cdot 10^4$/ $2.7 \cdot 10^7$

Interestingly, the times in the first column are higher than in the second. The reason might be that most of the r-subsets are not bases; therefore, there might be lots of indicator indices produced during the solving process. In fact, for $(g,k) \in \{(2,1), (3,1)\}$, we use a slightly different configuration because otherwise, DRAT-trim is not able to verify the results within a few hours due to the high number of clauses. Instead of adding several clauses in a single call of MinCheck, we terminate the check immediately after finding the first indicator index. This slightly increases the running time of SMS but allows us to verify the results. This also explains the lower number of clauses for $(g,k) \in \{(2,1), (3,1)\}$.

5.2 Enumerating Matroids

For enumerating all matroids with a given rank r and n elements modulo isomorphism, we use the formula $F_{n,r}^{\text{matroid}}$ from Section 4.1. Clingo supports enumerating all models for a given formula. Our results coincide with the numbers given by Acketa [1], Dukes [14], and

Mayhew and Royle [30]. It is worth noting that, while rank 2 matroids are well-studied and an explicit formula for their number is available (cf. A58681 in the OEIS [33]), for higher ranks, no explicit formula is known. Table 3 summarises the currently known numbers, all of which we can confirm with our framework within only a few hours of CPU time.

Table 3 The number of matroids with rank r and n elements.

$r\backslash n$	1	2	3	4	5	6	7	8	9	10	11	12	...	OEIS
1	1	2	3	4	5	6	7	8	9	10	11	12		n
2		1	3	7	13	23	37	58	87	128	183	259		A058682
3			1	4	13	38	108	325	1275	10037	298491	?		A058693
4				1	5	23	108	940	190214	?	?	?		A336704
5					1	6	37	325	190214	?	?	?		
6						1	7	58	1275	?	?	?		
7							1	8	87	10037	?	?		
8								1	9	128	298491	?		
9									1	10	183	?		
10										1	11	259		
11											1	12		
12												1		

6 Conclusion

We extended the SMS approach from graphs to matroids and presented the first work attacking matroid problems with SAT. We utilized SMS to verify Rota's Basis Conjecture for matroids of bounded rank, confirming it for rank 4 completely and for matroids of rank 5 and girth at least 4. Besides the SAT attack on Rota's Basis Conjecture, we enumerated all matroids up to 9 elements modulo isomorphism. Our framework can also be easily adapted to search for and enumerate matroids with particular properties. We also extended the SMS framework to produce DRAT proofs.

In the future, we plan to integrate other solvers such as CaDiCaL into SMS to be used in the first phase. This might be one of the key steps towards settling the remaining cases of Rota's Basis Conjecture for rank 5, that is, girth 2 and girth 3.

In contrast to other approaches, SMS can deal with a massive amount of symmetries, even if there are exponentially many. Consequently, we look forward to using SMS for solving various other combinatorial problems with lots of symmetries and, in particular, to provide a powerful and robust framework for many other structures besides graphs and matroids.

References

1 Dragan Acketa. The catalogue of all nonisomorphic matroids on at most 8 elements. *Special Issue, University of Novi Sad Institute of Mathematics Faculty of Science*, 1, 1983.

2 Noga Alon and Michael Tarsi. Colorings and orientations of graphs. *Combinatorica*, 12(2):125–134, 1992. doi:10.1007/BF01204715.

3 Martin Balko and Pavel Valtr. A SAT attack on the Erdős-Szekeres conjecture. *European Journal of Combinatorics*, 66:13–23, 2017. doi:10.1016/j.ejc.2017.06.010.

4 Armin Biere. CaDiCaL at the SAT Race 2019. In *Proc. of SAT Race 2019 – Solver and Benchmark Descriptions*, volume B-2019-1 of *Department of Computer Science Series*, pages 8–9. University of Helsinki, 2019. URL: http://researchportal.helsinki.fi/en/publications/proceedings-of-sat-race-2019-solver-and-benchmark-descriptions.

5 Matija Bucić, Matthew Kwan, Alexey Pokrovskiy, and Benny Sudakov. Halfway to Rota's basis conjecture. *Int. Math. Res. Not.*, 2020. doi:10.1093/imrn/rnaa004.

6 Wendy Chan. An exchange property of matroid. *Discrete Math.*, 146(1):299–302, 1995. doi:10.1016/0012-365X(94)00071-3.

7 Micheal S. Cheung. Computational proof of Rota's basis conjecture for matroids of rank 4. Unpublished manuscript, available at Joshua E. Ducey's website http://educ.jmu.edu/~duceyje/undergrad/2012/mike.pdf, 2012.

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

9 Karl Däubel, Sven Jäger, Torsten Mütze, and Manfred Scheucher. On orthogonal symmetric chain decompositions. *Electron. J. Combin.*, 26(3):Article Number P3.64, 32, 2019. doi:10.37236/8531.

10 Jo Devriendt, Bart Bogaerts, and Maurice Bruynooghe. Symmetric explanation learning: Effective dynamic symmetry handling for SAT. In *Theory and Applications of Satisfiability Testing - SAT 2017 - 20th International Conference*, volume 10491 of *Lecture Notes in Computer Science*, pages 83–100. Springer Verlag, 2017. doi:10.1007/978-3-319-66263-3_6.

11 Jo Devriendt, Bart Bogaerts, Maurice Bruynooghe, and Marc Denecker. Improved static symmetry breaking for SAT. In *Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference*, volume 9710 of *Lecture Notes in Computer Science*, pages 104–122. Springer Verlag, 2016. doi:10.1007/978-3-319-40970-2_8.

12 Michael R. Dransfield, Lengning Liu, Victor W. Marek, and Miroslaw Truszczyński. Satisfiability and computing van der Waerden numbers. *Electron. J. Combin.*, 11(1):Article Number R41, 15, 2004. doi:10.37236/1794.

13 Arthur A. Drisko. On the number of even and odd Latin squares of order $p+1$. *Adv. Math.*, 128(1):20–35, 1997. doi:10.1006/aima.1997.1623.

14 Mark Dukes. On the number of matroids on a finite set. *Séminaire Lotharingien de Combinatoire*, 51:Article B51g, 12, 2004. URL: https://www.mat.univie.ac.at/~slc/wpapers/s51dukes.html.

15 Benjamin Friedman and Sean McGuinness. Girth conditions and Rota's basis conjecture, 2020. arXiv:1908.01216.

16 Hiroshi Fujita. A new lower bound for the Ramsey number $R(4,8)$. arXiv:1212.1328, 2012.

17 Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Max Ostrowski, Torsten Schaub, and Philipp Wanko. Theory solving made easy with Clingo 5. In *Technical Communications of the 32nd International Conference on Logic Programming (ICLP 2016)*, volume 52 of *OASICS*, pages 2:1–2:15. Dagstuhl, 2016. doi:10.4230/OASIcs.ICLP.2016.2.

18 Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Clingo = ASP + control: Preliminary report, 2014. arXiv:1405.3694.

19 Jim Geelen and Peter J. Humphries. Rota's basis conjecture for paving matroids. *SIAM J. Discrete Math.*, 20(4):1042–1045, 2006. doi:10.1137/060655596.

20 Jim Geelen and Kerri Webb. On Rota's basis conjecture. *SIAM J. Discrete Math.*, 21(3):802–804, 2007. doi:10.1137/060666494.

21 P. R. Herwig, Marijn J. H. Heule, P. M. van Lambalgen, and H. van Maaren. A new method to construct lower bounds for van der Waerden numbers. *Electron. J. Combin.*, 14(1):Article Number R6, 18, 2007. doi:10.37236/925.

22 Marijn J. H. Heule. The DRAT format and DRAT-trim checker, 2016. arXiv:1610.06229.

23 Rosa Huang and Gian-Carlo Rota. On the relations of various conjectures on Latin squares and straightening coefficients. *Discrete Math.*, 128(1-3):225–236, 1994. doi:10.1016/0012-365X(94)90114-7.

24 Markus Kirchweger, Manfred Scheucher, and Stefan Szeider. A SAT Attack on Rota's Basis Conjecture: Supplemental data for rank 4. doi:10.5281/zenodo.6616373.

25 Markus Kirchweger, Manfred Scheucher, and Stefan Szeider. A SAT Attack on Rota's Basis Conjecture: Supplemental source code. doi:10.5281/zenodo.6616343.

26 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, pages 39:1–39:17. Dagstuhl, 2021. doi:10.4230/LIPIcs.CP.2021.34.

27 Boris Konev and Alexei Lisitsa. A SAT attack on the Erdős discrepancy conjecture. In *Theory and Applications of Satisfiability Testing - SAT 2014 - 17th International Conference*, volume 8561 of *Lecture Notes in Computer Science*, pages 219–226. Springer Verlag, 2014. doi:10.1007/978-3-319-09284-3_17.

28 Michal Kouril and Jerome L. Paul. The van der Waerden number $W(2, 6)$ is 1132. *Experiment. Math.*, 17(1):53–61, 2008. URL: http://projecteuclid.org/euclid.em/1227031896.

29 Filip Marić. Fast formal proof of the Erdős–Szekeres conjecture for convex polygons with at most 6 points. *Journal of Automated Reasoning*, 62:301–329, 2019. doi:10.1007/s10817-017-9423-7.

30 Dillon Mayhew and Gordon F. Royle. Matroids with nine elements. *J. Combin. Theory Ser. B*, 98(2):415–431, 2008. doi:10.1016/j.jctb.2007.07.005.

31 Hakan Metin, Souheib Baarir, Maximilien Colange, and Fabrice Kordon. CDCLSym: Introducing effective symmetry breaking in SAT solving. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 99–114. Springer Verlag, 2018. doi:10.1007/978-3-319-89960-2_6.

32 Torsten Mütze and Manfred Scheucher. On L-shaped point set embeddings of trees: first non-embeddable examples. *J. Graph Algorithms Appl.*, 24(3):343–369, 2020. doi:10.7155/jgaa.00537.

33 OEIS Foundation Inc. The On-Line Encyclopedia of Integer Sequences. Published electronically at http://oeis.org.

34 Shmuel Onn. A colorful determinantal identity, a conjecture of Rota, and Latin squares. *Amer. Math. Monthly*, 104(2):156–159, 1997. doi:10.2307/2974985.

35 Open Problem Garden. Rota's basis conjecture, 2009. URL: http://www.openproblemgarden.org/op/rotas_basis_conjecture.

36 James Oxley. *Matroid theory*, volume 21 of *Oxford Graduate Texts in Mathematics*. Oxford University Press, second edition, 2011. doi:10.1093/acprof:oso/9780198566946.001.0001.

37 Alexey Pokrovskiy. Rota's basis conjecture holds asymptotically, 2020. arXiv:2008.06045.

38 Bas Schaafsma, Marijn Heule, and Hans van Maaren. Dynamic symmetry breaking by simulating Zykov contraction. In *Theory and Applications of Satisfiability Testing - SAT 2009*, volume 5584 of *Lecture Notes in Computer Science*, pages 223–236. Springer Verlag, 2009. doi:10.1007/978-3-642-02777-2_22.

39 Manfred Scheucher. Two disjoint 5-holes in point sets. *Comput. Geom.*, 91:101670, 2020. doi:10.1016/j.comgeo.2020.101670.

40 Manfred Scheucher, Hendrik Schrezenmaier, and Raphael Steiner. A note on universal point sets for planar graphs. *J. Graph Algorithms Appl.*, 24(3):247–267, 2020. doi:10.7155/jgaa.00529.

41 Toby Walsh. General symmetry breaking constraints. In *Principles and Practice of Constraint Programming - CP 2006, 12th International Conference, CP 2006*, volume 4204 of *Lecture Notes in Computer Science*, pages 650–664. Springer Verlag, 2006. doi:10.1007/11889205_46.

42 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. doi:10.1007/978-3-319-09284-3_31.

43 O. Zaikin, S. Kochemazov, and A. A. Semenov. SAT-based search for systems of diagonal Latin squares in volunteer computing project SAT@home. In *39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO 2016)*, pages 277–281, 2016. doi:10.1109/MIPRO.2016.7522152.

44 I. Zinovik, D. Kroening, and Y. Chebiryak. Computing binary combinatorial Gray codes via exhaustive search with SAT solvers. *Institute of Electrical and Electronics Engineers. Transactions on Information Theory*, 54(4):1819–1823, 2008. doi:10.1109/TIT.2008.917695.