# ON THE GENERATION OF RANK 3 SIMPLE MATROIDS WITH AN APPLICATION TO TERAO'S FREENESS CONJECTURE

MOHAMED BARAKAT, REIMER BEHRENDS, CHRISTOPHER JEFFERSON, LUKAS KÜHNE, AND MARTIN LEUNER

ABSTRACT. In this paper we describe a parallel algorithm for generating all non-isomorphic rank 3 simple matroids with a given multiplicity vector. We apply our implementation in the HPC version of GAP to generate all rank 3 simple matroids with at most 14 atoms and a splitting characteristic polynomial. We have stored the resulting matroids alongside with various useful invariants in a publicly available, ArangoDB-powered database. As a byproduct we show that the smallest divisionally free rank 3 arrangement which is not inductively free has 14 hyperplanes and exists in all characteristics distinct from 2 and 5. Another database query proves that Terao's freeness conjecture is true for rank 3 arrangements with 14 hyperplanes in any characteristic.

## 1. INTRODUCTION

In computational mathematics one often encounters the problem of scanning (finite but) large sets of certain objects. Here are two typical scenarios:

- Searching for a counter-example of an open conjecture among these objects.
- Building a database of such objects alongside with some of their invariants.

A database is particularly useful when the questions asked are relational, i.e., involve more than one object (see Remark 2.11). Recognized patterns and questions which a database answers affirmatively may lead to working hypotheses (see Example 1.1) or even proofs by inspection (see Theorem 1.4).

In any such scenario there is no need to simultaneously hold the entire set in RAM. It is hence important to quickly *iterate* over such sets in a memory efficient way rather than to enumerate them.

The central idea is to represent each such set $T$ as the set of leaves of a rooted tree $T_\bullet$ and to describe an iterator of $T$ as the so-called leaf-iterator $\ell(t)$ of a tree-iterator $t$ (cf. Definition B.1). In other words, we embed $T$ in the bigger set of vertices $V(T_\bullet)$ as the set of leaves. We then say that $T_\bullet$ **classifies** $T$. The internal vertices of the tree $T_\bullet$ are usually of different nature than the elements of $T$. Their sole purpose is to encode common pre-stages of the leaves.

In Section 3 we will describe how to use tree iterators to generate all non-isomorphic rank 3 simple matroids with up to 14 atoms and integrally splitting characteristic polynomial.

A simple matroid $M$ of rank 3 on $n$ labeled points corresponds to a bipartite graph $G_M$ (cf. Remark 3.2). We denote by $(m_2, \ldots, m_{n-1})$ the **multiplicity vector** of $M$ where $m_k$ is the number of coatoms of multiplicity $k$, i.e., the degree in the bipartite graph corresponding to $M$ (cf. Definition 3.3). The multiplicity vector determines the characteristic polynomial

of $M$:

$$(*) \qquad \frac{\chi_M(t)}{t-1} = t^2 - (n-1)t + (b_2 - (n-1)) \quad \text{with} \quad b_2 := \sum_{k=2}^{n-1} m_k(k-1).$$

In fact, two rank 3 simple matroids (or more generally, two paving matroids) have the same multiplicity vector $(m_2, \ldots, m_{n-1})$ iff their Tutte polynomials coincide [Bry72].

After extending the notions of inductive and divisional freeness from arrangements to matroids (see Definitions 2.8 and 2.9) we get the following table of cardinalities[1] of certain classes of non-isomorphic rank 3 simple matroids. A matroid is called *Tutte-unique* or *T-unique* if it is determined up to isomorphism by its Tutte polynomial[2] (see [dMN05] for a survey on T-unique matroids). The content of the table can be reconstructed using the database [BK19].

| number of atoms | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| rank 3, simple matroids | | | | | | | | | | | | |
| simple matroids | 1 | 2 | 4 | 9 | 23 | 68 | 383 | 5 249 | 232 928 | 28 872 972 | ? | ? |
| integral roots | 1 | 1 | 2 | 3 | 7 | 7 | 17 | 35 | 163 | 867 | 30 724 | 783 280 |
| divisionally free | 1 | 1 | 2 | 3 | 6 | 7 | 15 | 33 | 147 | 857 | 28 287 | 781 795 |
| inductively free | 1 | 1 | 2 | 3 | 6 | 7 | 15 | 33 | 147 | 839 | 27 931 | 750 305 |
| supersolvable | 1 | 1 | 2 | 3 | 5 | 7 | 11 | 20 | 41 | 118 | 518 | 4 820 |
| representable, rank 3, simple matroids | | | | | | | | | | | | |
| rep. integral roots | 1 | 1 | 2 | 3 | 7 | 7 | 17 | 30 | 86 | 208 | 999 | 1 574 |
| rep. divisionally free | 1 | 1 | 2 | 3 | 6 | 7 | 15 | 28 | 75 | 198 | 631 | 1 401 |
| rep. inductively free | 1 | 1 | 2 | 3 | 6 | 7 | 15 | 28 | 75 | 198 | 631 | 1 400 |
| rep. supersolvable | 1 | 1 | 2 | 3 | 5 | 7 | 11 | 20 | 35 | 82 | 223 | 649 |
| Tutte-unique, rank 3, simple matroids | | | | | | | | | | | | |
| T.-u. integral roots | 1 | 1 | 2 | 3 | 7 | 5 | 11 | 10 | 17 | 17 | 18 | 23 |
| T.-u. divisionally free | 1 | 1 | 2 | 3 | 6 | 5 | 9 | 10 | 14 | 16 | 17 | 21 |
| T.-u. inductively free | 1 | 1 | 2 | 3 | 6 | 5 | 9 | 10 | 14 | 16 | 17 | 21 |
| T.-u. supersolvable | 1 | 1 | 2 | 3 | 5 | 5 | 8 | 10 | 12 | 14 | 15 | 19 |
| representable, Tutte-unique, rank 3, simple matroids | | | | | | | | | | | | |
| rep. T.-u. int. roots | 1 | 1 | 2 | 3 | 7 | 5 | 11 | 10 | 16 | 17 | 17 | 22 |
| rep. T.-u. div. free | 1 | 1 | 2 | 3 | 6 | 5 | 9 | 10 | 13 | 16 | 16 | 20 |
| rep. T.-u. ind. free | 1 | 1 | 2 | 3 | 6 | 5 | 9 | 10 | 13 | 16 | 16 | 20 |
| rep. T.-u. supersolv. | 1 | 1 | 2 | 3 | 5 | 5 | 8 | 10 | 12 | 14 | 15 | 19 |

**Table 1.** Cardinalities of certain classes of non-isomorphic rank 3 simple matroids.

The total number of rank 3 simple matroids with $n \leq 12$ (unlabeled) atoms[3] is taken from [MMIB12b]. This number also coincides with minus one plus the number of linear geometries with $n \leq 12$ (unlabeled) points[4] and has been determined earlier in [BB99].

Using our algorithm in HPC-GAP we directly computed all 815107 rank 3 simple matroids with integrally splitting characteristic polynomial with up to $n = 14$ atoms, stored them in the database [BK19], and verified the previous counting for $n \leq 11$ in [MMIB12a].

**Example 1.1.** The database [BK19] enables us to ask questions on the role of the Tutte polynomial such as:

---

[1]Apart from the number of simple matroids, we were unable to find any of the sequences in the above table in the OEIS database.

[2]The Tutte polynomial of all rank 3 integrally split matroids with up to 13 atoms was computed using the GAP package alcove [Leu19].

[3]http://oeis.org/A058731

[4]http://oeis.org/A001200

- Is being divisionally or inductively free a property determined by the Tutte polynomial?

  We answer this question negatively in Example 1.2.
- Is the corresponding question (for rank 3 matroids) true for the stricter notion of supersolvability?

  Answer: In this case, we found that for any pair of rank 3 matroids $M_1$, $M_2$ up to 14 atoms having the same Tutte polynomial $M_1$ is supersolvable iff $M_2$ is supersolvable. Hence, we formulate this question as a working hypotheses found through inspection of our database.

For such questions we need to construct all matroids with the corresponding number of atoms first, demonstrating the usefulness of a database.

**Example 1.2.** Consider the rank 3 matroids $M_1$ and $M_2$ of size 11 given below by the adjacency lists $A_1$ and $A_2$ of their corresponding bipartite graph respectively.

$A_1 := \{\{1,2,3,4\},\{1,5,6,7\},\{1,8,9,10\},\{2,5,8,11\},\{3,6,9,11\},\{2,6,10\},\{2,7,9\},\{3,5,10\},\{4,5,9\},\{4,7,11\},$
$\qquad \{1,11\},\{3,7\},\{3,8\},\{4,6\},\{4,8\},\{4,10\},\{6,8\},\{7,8\},\{7,10\},\{10,11\}\},$

$A_2 := \{\{1,2,3,4\},\{1,5,6,7\},\{2,5,8,9\},\{3,6,8,10\},\{4,7,9,10\},\{1,8,11\},\{2,7,11\},\{3,9,11\},\{4,6,11\},\{5,10,11\},$
$\qquad \{1,9\},\{1,10\},\{2,6\},\{2,10\},\{3,5\},\{3,7\},\{4,5\},\{4,8\},\{6,9\},\{7,8\}\}.$

The matroids $M_1$ and $M_2$ are representable over $\mathbb{Q}$ and $\mathbb{Q}(\sqrt{5})$, respectively. Their representation matrices are given by

$$R_1 := \begin{pmatrix} 1 & \cdot & 1 & 1 & \cdot & 1 & 1 & \cdot & 1 & 1 & \cdot \\ \cdot & 1 & 1 & \frac{1}{2} & \cdot & \cdot & \cdot & 1 & \frac{1}{2} & 1 & 1 \\ \cdot & \cdot & \cdot & \cdot & 1 & 1 & \frac{1}{2} & 1 & \frac{1}{2} & 1 & -1 \end{pmatrix},$$

$$R_2 := \begin{pmatrix} 1 & \cdot & 1 & 1 & \cdot & 1 & 1 & \cdot & \cdot & 1 & 1 \\ \cdot & 1 & 1 & \varphi+1 & \cdot & \cdot & \cdot & 1 & 1 & -\varphi & -\varphi \\ \cdot & \cdot & \cdot & \cdot & 1 & 1 & \varphi & -1 & -\varphi+1 & \varphi+1 & \varphi \end{pmatrix},$$

where $\varphi = \frac{1+\sqrt{5}}{2}$ denotes the golden ratio. Their multiplicity vectors agree and are given by $(m_k) = (m_2, m_3, m_4) = (10, 5, 5)$. Hence, their Tutte polynomials also agree:

$$T_{M_1}(x,y) = T_{M_2}(x,y) = y^8 + 3y^7 + 6y^6 + 10y^5 + 15y^4 + x^3 + 5xy^2 + 21y^3 + 8x^2 + 15xy + 23y^2 + 16x + 16y.$$

The characteristic polynomials of $M_1$ and $M_2$ factor completely over the integers

$$\chi_{M_1}(t) = \chi_{M_2}(t) = (t-1)(t-5)^2.$$

Using the database we found that $M_1$ is inductively free and hence divisionally free whereas $M_2$ is not even divisionally free. We checked with GAP that any representation of $M_2$ is a free arrangement. Both are not supersolvable.

The database also shows that for rank 3 matroids this example is minimal with respect to the number of elements.

**Example 1.3.** Inspecting the table (resp. the database) we can also answer the following *Question:* What is the smallest number of atoms of a representable rank 3 matroid which is divisionally free but not inductively free?[5]
*Answer:* Among the rank 3 matroids with up to 14 atoms there is a unique representable matroid $M$ with 14 atoms which is divisionally free but not inductively:

$$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 2a-1 & 2a & 0 & 0 & 0 & 0 & 1 & -2a+2 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & -2a+1 & -a+1 & a & 1 & a & 2a-1 & 1 \end{pmatrix},$$

---

[5]It is already know that such a matroid exists, namely the rank 3 reflection arrangement $\mathcal{A}(G_{24})$ (with 21 hyperplanes) of the exceptional complex reflection group $W = G_{24}$ is recursively free [Mü17] but not inductively free [HR15]. Hence, an addition of $\mathcal{A}(G_{24})$ is easily seen to be divisionally free but not inductively free. Therefore, the sequences of representable divisionally free and inductively free matroids differ at $n = 22$ at the latest.

where $a$ satisfies the equality $2a^2 - 2a + 1 = 0$ and the inequation $(3a - 1)(a + 1) \neq 0$. In particular $M$ is representable in any characteristic distinct from 2 and 5.

Its characteristic polynomial is $\chi_M(t) = (t - 1)(t - 6)(t - 7)$. The restriction $M''$ of $M$ to its third atom (resp. hyperplane) has characteristic polynomial $\chi_M(t) = (t - 1)(t - 6)$ which shows that any arrangement representing $M$ is divisionally free (cf. Definition 2.9). Furthermore, the Tutte polynomial of $M$ is

$$y^{11} + 3y^{10} + 6y^9 + 10y^8 + 15y^7 + 21y^6 + 28y^5 + 2xy^3 + 36y^4 + x^3 + 10xy^2 + 43y^3 + 11x^2 + 24xy + 43y^2 + 30x + 30y.$$

A central notion in the study of hyperplane arrangements is freeness. A central arrangement of hyperplanes $\mathcal{A}$ is called *free* if the derivation module $D(\mathcal{A})$ is a free module over the polynomial ring. An important open question in this field is Terao's conjecture which asserts that the freeness of an arrangement over a field $k$ only depends on its underlying matroid and the characteristic of $k$. It is known that Terao's conjecture holds for arrangements with up to 12 hyperplanes in characteristic 0 (cf. [FV14, ACKN16]). Recently, Dimca, Ibadula, and Macinic confirmed Terao's conjecture for arrangements in $\mathbb{C}^3$ with up to 13 hyperplanes [DIM17].

Inspecting our database we obtain the following result:

**Theorem 1.4.** *Terao's freeness conjecture is true for rank 3 arrangements with 14 hyperplanes in any characteristic.*

This article is organized as follows: In Section 2 we recall the notion of a matroid and introduce several subclasses of rank 3 simple matroids. In Section 3 with discuss the Algorithm used to construct tree-iterators generating all non-isomorphic rank 3 simple matroids with up to $n = 14$ atoms having an integrally splitting characteristic polynomial. In Section 4 we briefly point out how to use Gröbner bases to compute the moduli space of representations (over some unspecified field $\mathbb{F}$) of a matroid as an affine variety over $\operatorname{Spec} \mathbb{Z}$. In Section 5 we finally prove Theorem 1.4. In Appendix A we collect some terminology about rooted trees. In Appendix B we define recursive and tree-iterators and introduce algorithms to parallely evaluate them. Appendix C summarizes the merits of the high performance computing (HPC) version of GAP, which we used to implement the above mentioned algorithms. We conclude by giving some timings in Appendix D to demonstrate the significance of our parallelized algorithms in the generation of (certain classes) of rank 3 simple matroids.

## 2. SIMPLE MATROIDS

2.1. **Basic definitions.** Finite simple matroids have many equivalent descriptions. For our purposes we prefer the one describing the lattice of flats.

**Definition 2.1.** A **matroid** $M = (E, \mathcal{F})$ consists of a finite ground set $E$ and a collection $\mathcal{F}$ of subsets of $E$, called **flats** (of $M$), satisfying the following properties:

(a) The ground set $E$ is a flat;
(b) The intersection $F_1 \cap F_2$ is a flat, if $F_1$ and $F_2$ are flats;
(c) If $F$ is a flat, then any element in $E \setminus F$ is contained in exactly one flat covering $F$.

Here, a flat is said to **cover** another flat $F$ if it is minimal among the flats properly containing $F$. A matroid is called **simple** if

(d) it is loopless, i.e., $\emptyset$ is a flat;

(e) it contains no parallel elements, i.e., the singletons are flats, which are called **atoms**.

For a matroid $M = (E, \mathcal{F})$ and $S \subseteq E$ we denote by $\mathrm{rk}(S)$ the **rank of** $S$ which is the maximal length of chains of flats in $\mathcal{F}$ all contained in $S$. The **rank of the matroid** $M$ is defined to be $rk(E)$. A subset $S \subseteq E$ is called **independent** if $|S| = rk(S)$ and otherwise **dependent**. A maximal independent set is called a **basis of** $M$. Hence, the cardinality of any basis equals the rank of the matroid.

*Remark* 2.2 (Basis Extension Theorem). Any independent subset of a matroid can be extended to a basis.

The flats form a poset[6] $\mathcal{F}$ by inclusion. Dually, a **coatom** is a maximal element in $\mathcal{F} \backslash \{E\}$. An isomorphism between the matroids $(E, \mathcal{F})$ and $(E', \mathcal{F}')$ is a bijective map $E \to E'$ which induces an isomorphism $\mathcal{F} \to \mathcal{F}'$ of posets.

Originally, matroids were introduced as an abstraction of the notion of linear (in)dependence in linear algebra.

**Example 2.3.** A **central arrangement** over a field $\mathbb{F}$ is a finite set $\mathcal{A}$ of $(n-1)$-dimensional subspaces of an $n$-dimensional $\mathbb{F}$-vector space $V$. The intersection lattice $L(\mathcal{A})$ is the set of intersections of subsets of $\mathcal{A}$, partially ordered by reverse inclusion, where the empty (set-theoretic) intersection is defined as $V$. The arrangement $\mathcal{A}$ is called **essential** if $\{0\} \in L(\mathcal{A})$.

The pair $(\mathcal{A}, L(\mathcal{A}))$ is a matroid of rank $n - \dim \bigcap_{H \in \mathcal{A}} H$, i.e., of rank $n$ iff $\mathcal{A}$ is essential. We call such a pair a **vector matroid** over $\mathbb{F}$.

This example motivates the following definition:

**Definition 2.4.** A matroid is called **representable over the field** $\mathbb{F}$ if it is isomorphic to a vector matroid over $\mathbb{F}$. A matroid is called **representable** if it is representable over some field $\mathbb{F}$.

The following matroid invariant and its specialization play an important role in our study of simple rank 3 matroids.

**Definition 2.5.** The **Tutte Polynomial** $T_M(x, y)$ of a matroid $M = (E, \mathcal{F})$ is defined by

$$T_M(x, y) := \sum_{S \in \mathcal{P}(E)} (x - 1)^{r(M) - r(S)} (y - 1)^{|S| - r(S)}.$$

A matroid is called **Tutte-unique**, if it is determined by its Tutte polynomial, i.e. any matroid with the same Tutte polynomial is isomorphic to the given one. An important evaluation of the Tutte polynomial is the **characteristic polynomial**

$$\chi_M(t) := (-1)^{r(M)} T_M(1 - t, 0) = \sum_{S \in \mathcal{P}(E)} (-1)^{|S|} t^{r(M) - r(S)}.$$

There is a simpler definition of the monic characteristic polynomial of a *simple* matroid using the **Möbius function** $\mu : \mathcal{F} \to \mathbb{Z}$, which is defined as follows:

- $\mu(\emptyset) = 1$;
- $\sum_{F \supset F' \in \mathcal{F}} \mu(F') = 0$ for all $F \in \mathcal{F} \setminus \{\emptyset\}$.

One can then show that

$$\chi_M(t) = \sum_{F \in \mathcal{F}} \mu(F) t^{r(M) - r(F)}.$$

---

[6]The poset of flats is a geometric lattice, i.e., a finite atomic semimodular lattice. Conversely, finite atomic semimodular lattices give rise to matroids.

The main application of this article is the enumeration of matroids whose characteristic polynomial completely factors over the integers. We will denote the class of rank $r$ matroids with **integrally splitting characteristic polynomial** by $\mathcal{ISM}_r$. Such a factorization of $\chi_M(t)$ is often implied by stronger combinatorial or (in the representable case) algebraic/geometric properties. The only known converse statement is that $\chi_M(t)$ factors completely over the integers for a graphic or cographic matroid $M$ induced by a planar graph $G$ if and only if $G$ is chordal as shown in [DK98] for graphic and in [KR11] for cographic matroids respectively. In both cases, the splitting of $\chi_M(t)$ over $\mathbb{Z}$ even implies that $M$ is supersolvable. However, it is still safe to say that these rather small classes of matroids are not yet well understood when $r \geq 3$.

2.2. **Rank** $3$ **simple matroids.** For simplicity we will restrict ourselves to the case of rank $3$ matroids in the following definitions. The smallest class we will consider is that of supersolvable matroids introduced by Stanley in [Sta72]. In the rank $3$ case the definition can be given as follows:

**Definition 2.6.** A matroid $M = (E, \mathcal{F})$ of rank $3$ is **supersolvable** if there exists a flat $F_0 \in \mathcal{F}$ of rank $2$ such that every intersection with other flats of rank $2$ is non-trivial. In this case the characteristic polynomial splits over the integers with roots

$$\chi_M(t) = (t - 1)\,(t - (|F_0| - 1))\,(t - (|E| - |F_0|)).$$

Define $\mathcal{SSM}_3$ to be the class of all supersolvable rank $3$ matroids.

To introduce the next combinatorial classes of matroids we need the notions of deletion and reduced contraction of a matroid with respect to an element $H$ of the ground set $E$.

**Definition 2.7.** Let $M = (E, \mathcal{F})$ be a matroid and $H \in E$. Define the **deletion of** $H$ to be the matroid $M' := M \setminus H := (E', \mathcal{F}')$ where

$$E' := E \setminus H := E \setminus \{H\},$$
$$\mathcal{F}' := \mathcal{F} \setminus H := \{F \setminus \{H\} \mid F \in \mathcal{F}\}.$$

The **reduced contraction**[7] **of** $H$ is the matroid $M'' := M^H := (E'', \mathcal{F}'')$ where

$$\mathcal{F}'' := \mathcal{F}^H := \{F \in \mathcal{F} \mid \{H\} \subseteq F\},$$

and its atoms $E'' = E^H$ are identified with the flats of rank $1$ in $\mathcal{F}^H$. If $\{H\}$ is a flat in $M$ then $M^H$ is a simple matroid. In particular, if $M$ is simple then so are $M \setminus H$ and $M^H$.

The following two classes stem from the notion of freeness of hyperplane arrangements.

**Definition 2.8.** We define the class $\mathcal{IFM}_3$ of **inductively free rank** $3$ **matroids** to be the smallest class of rank $3$ simple matroids containing

- the boolean matroid $M_3 := (\{1, 2, 3\}, \mathcal{P}(\{1, 2, 3\}))$ and
- $M = (E, \mathcal{F})$ with $|E| > 3$ if there exists an $H \in E$ such that $\chi_{M^H}(t)|\chi_M(t)$ and $M \setminus H \in \mathcal{IFM}_3$.

Recently, Abe introduced a larger class of combinatorially free arrangements in [Abe16].

**Definition 2.9.** The class $\mathcal{DFM}_3$ of **divisionally free rank** $3$ **matroids** is the smallest class of rank $3$ simple matroids containing

- the boolean matroid $M_3 := (\{1, 2, 3\}, \mathcal{P}(\{1, 2, 3\}))$ and
- $M = (E, \mathcal{F})$ with $|E| > 3$ if there exists an $H \in E$ such that $\chi_{M^H}(t)|\chi_M(t)$.

---

[7]This definitions mimics the usual notion of restriction for hyperplane arrangements. Note that it differs from the matroid-theoretic contraction since it does not contain loops and parallel elements.

*Remark* 2.10. The following strict inclusions hold

$$\mathcal{SSM}_3 \subsetneq \mathcal{IFM}_3 \subsetneq \mathcal{DFM}_3 \subsetneq \mathcal{ISM}_3,$$

where the first strict inclusion is shown in [JT84] and the second inclusion in [Abe16] (for strictness of the inclusion in rank 3 cf. Example 1.3 and for vector matroids of rank at least 4 cf. loc. cit.). The last inclusion holds by the definition of divisional freeness and since $\chi_M(t) = (t-1)(t-(|E|-1))$ for any simple matroid $M = (E, \mathcal{F})$ of rank 2 (for strictness see the table in Table 1, for example).

*Remark* 2.11. Due to the recursive nature of the definition of inductive freeness, a database containing the rank 3 simple matroids with up to $n$ atoms is extremely useful when deciding the inductive freeness of those with $n + 1$ atoms. This is how we determined the subclass $\mathcal{IFM}_3$ in our database.

## 3. GENERATING RANK 3 MATROIDS WITH INTEGRALLY SPLITTING CHARACTERISTIC POLYNOMIALS

Since we will focus on the rank 3 case we prefer to describe them as special instances of bipartite graphs. And as already mentioned in the introduction, the description of tree iterators $T_\bullet$ generating rank 3 simple matroids will rely on the language of bipartite graphs.

**Definition 3.1** ([Oxl11]). A 2-**partition** of a finite set $E$ is a set $\mathcal{E}$ of subsets of $E$ such that

    (a) $|F| \geq 2$ for all $F \in E$;
    (b) $\{\mathrm{Pot}_2(F) \mid F \in \mathcal{E}\}$ is a partition of $\mathrm{Pot}_2(E)$,

where $\mathrm{Pot}_2(E)$ denotes the set of two-element subsets of $E$. We call the elements of $E$ atoms and those of $\mathcal{E}$ coatoms.

*Remark* 3.2. Let $(E, \mathcal{E})$ be a 2-partition.

- $\bigcup \mathcal{E} = E$.
- $|F \cap F'| \leq 1$ for all $F, F' \in \mathcal{E}$ with $F \neq F'$.
- The union $E \cup \mathcal{E}$ defines the vertices of a bipartite graph with adjacency given by membership. We call bipartite graphs admitting such a description **matroidal**. Connecting, as in Figure 1, the atoms with an initial element and the coatoms with a terminal element we obtain a geometric lattice of flats of a rank 3 simple matroid. Hence, there is a bijective correspondence between rank 3 simple matroids with ground set $E$ and 2-partitions of $E$.
- $\sum_{F \in \mathcal{E}} \binom{|F|}{2} = \binom{|E|}{2}$.

In words, each pair of atoms is contained in exactly one coatom. Hence, the left hand side of the last equation counts the number of pairs of atoms which are joined by the coatoms. This count must be equal to the number of all pairs of atoms which is the right hand side of the equation.

Remark 3.2 gives rise to the following definition:

**Definition 3.3.** For $n \in \mathbb{N}$ we call $(m_k) := (m_k)_{k=2,\ldots,n-1} := (m_2, \ldots, m_{n-1})$ a **multiplicity vector** of size $n$ if $\sum_{k=2}^{n-1} m_k \binom{k}{2} = \binom{n}{2}$.

An example of such a matroidal bipartite graph corresponding to the rank 3 braid arrangement $\mathcal{A}_3$ is given in Figure 1. It has the multiplicity vector $(m_2, m_3) = (3, 4)$ and the characteristic polynomial $\chi_{\mathcal{A}_3}(t) = (t-1)(t-2)(t-3)$.
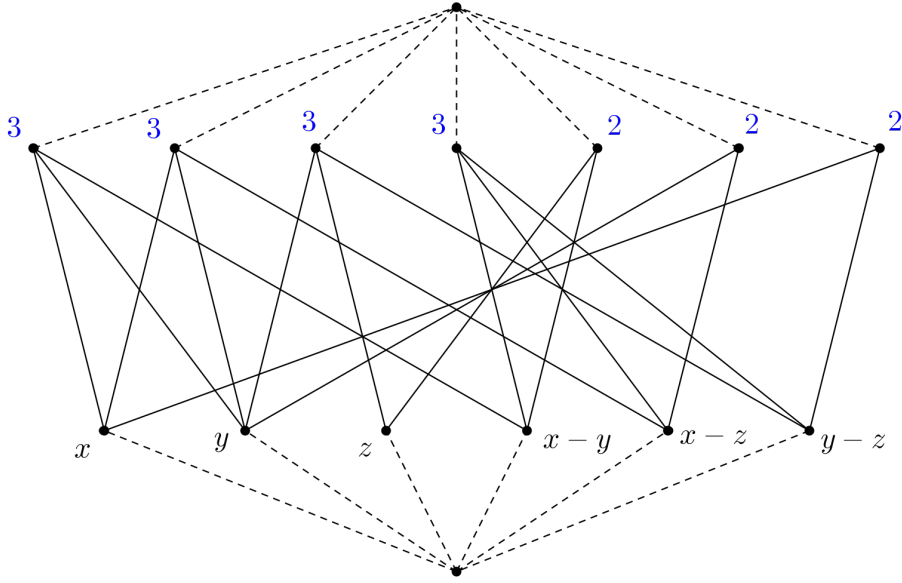
**Figure 1.** The bipartite graph of the $\mathcal{A}_3$ braid arrangement. The linear forms depicted next to the atoms are a possible representations of the matroid. The numbers denoted in blue are the multiplicities of the coatoms.

To generate the multiplicity vectors of all simple rank 3 matroids of fixed size $n$ we can naively iterate over all vectors in $\{1, \ldots, n-1\}^{n-2}$ satisfying the equation in Definition 3.3. Furthermore, we are only considering those multiplicity vectors $(m_k)$ such that the corresponding characteristic polynomial as in (*) factors completely over the integers.

In what follows we will use the language of rooted trees which will be summarized in Appendix A.

We will now describe a rooted tree $T_\bullet^{(m_k)}$ classifying the set $T^{(m_k)}$ of simple matroids of rank 3 on $n$ labeled atoms[8], for $n \in \mathbb{N}_{>0}$ and fixed multiplicity vector $(m_k)$: We set $T_0^{(m_k)} := \{*\}$ a singleton. For $i \in \mathbb{N}_{>0}$ the set $T_i^{(m_k)}$ is an approximation of $T^{(m_k)}$: If $G_M \in T^{(m_k)}$ is a bipartite graph, then its image in $T_i^{(m_k)}$ is the full subgraph containing the $\sum_{\ell \geq k} m_\ell$ coatoms of cardinality at least $k = n - i$ together with their adjacent atoms. All maps $T_{i-1}^{(m_k)} \xleftarrow{\varphi} T_i^{(m_k)}$ are evident.

The family of these trees $T_\bullet^{(m_k)}$ forms a forest which can be turned into a tree $\widetilde{T}_\bullet$ in the obvious way.

The non-isomorphic simple matroids of rank 3 on $n$ unlabeled atoms[9] are then classified by $\widetilde{T}_\bullet / S_n$. Since the multiplicity vector of a rank 3 matroid $M$ determines the characteristic polynomial $\chi_M$ by (*) we can restrict to the subtree $T_\bullet \subset \widetilde{T}_\bullet$ consisting of the sub-forest of those $T^{(m_k)}$ with **integrally splitting characteristic polynomial**, i.e., where the characteristic polynomial has only integral roots.

**Definition 3.4.** We call a set $A = \{A_i\}$ of subsets of $\{1, \ldots, n\}$ an **admissible partial 2-partition** of level $k_0 \in \{1, \ldots, n-1\}$ for a multiplicity vector $(m_k) = (m_2, \ldots, m_{n-1})$ of size $n$ if

- $|\{i \mid |A_i| = k\}| = m_k$ for all $k$ with $n - 1 \geq k > k_0$;
- $|A_i \cap A_j| \leq 1$ for all $i < j$.

---

[8]Cf. (http://oeis.org/A058720, for $k = 3$).

[9]Cf. (http://oeis.org/A058731).

To generate all simple rank 3 simple matroids of a fixed multiplicity vector $(m_k)$ we propose an algorithm that takes as input an admissible partial 2-partition $A$ of some level $k_0$ and iterates over all possible extensions to admissible partial 2-partition of the next non-trivial smaller level $k_1$ with $k_1 < k_0$ as a tree-iterator. In this computation we only consider lexicographically minimal extensions with respect to the stabilizer of $A$ under the action of the symmetric group $S_n$ to avoid iterating over isomorphic matroids multiple times. The details of this procedure are given in Algorithm 1.

Finally, to build the tree-iterator of all rank 3 simple matroids with $n$ atoms and multiplicity vector $(m_k)$ (as bipartite graphs) we apply **IteratorFromState** to the initial state

$$s^{(m_k)} := \big(n, (m_k), k_0 := \max\{k \mid m_k > 0\}, A := ()\big).$$

We have implemented the four algorithms

(1) **IteratorFromState** (Algorithm 1)
(2) **ParallelyEvaluateRecursiveIterator** (Algorithm 2);
(3) **EvaluateRecursiveIterator** (Algorithm 3)
(4) **LeafIterator** (Algorithm 4)

in the HPC version[10] of GAP 4.9.2 [GAP18]. The last three of them are independent of the specific recursive iterator defined by **IteratorFromState** (Algorithm 1) and can be viewed as part of a general parallelization scheme for recursive iterators. Their detailed description is therefore deferred to Appendix B.

Algorithm 4 is executed in the main thread with an initial tree-iterator of all rank 3 matroids corresponding to a given multiplicity vector as input. This initial tree-iterator is constructed using Algorithm 1. Algorithm 4 then initializes a global FIFO $L$ of leaves and invokes Algorithm 2. The latter creates a shared priority queue $P$, launches as much workers (threads) as specified by the user, triggers Algorithm 3 in each of them, and then terminates.

The shared priority queue stores the list of tree-iterators still to be searched along with their priority, which in our case is the depth at which they were created. Tree-iterators can be added to the priority queue, and when a thread requests a new tree-iterator to work on, the priority queue returns one of the tree-iterators with highest priority currently in the queue.

Each thread then searches the highest priority tree-iterator given by the priority queue $P$ and moves the resulting tree-iterator with priority increased by one back to $P$ or the resulting matroids into the FIFO $L$ of leaves. The latter is constantly monitored by Algorithm 4 and the included matroids (the leaves) are read and stored into the database [BK19]. In particular, our use of a priority queue avoids the need for a central process supervising the workers.

Implementations of priority queues exist both for shared memory and distributed operation. We have chosen to use a simple shared memory implementation, as contention for our workloads is very low, so we do not have to worry about the priority queue becoming a serialization bottleneck.

---

[10]Since version 4.9.1 GAP can be compiled with the option `--enable-hpcgap`.

---

**Algorithm 1:** IteratorFromState

**Input:** state $s$ consisting of
- a number $n$ of atoms
- a multiplicity vector $(m_k) = (m_2, \ldots, m_{n-1})$ of size $n$     `// cf. Section 1`
- an integer $n - 1 \geq k_0 \geq 2$ with $m_{k_0} > 0$
- an admissible partial 2-partition $A$ of level $k_0$ for $(m_k)$    `// cf. Definition 3.4`

**Output:** tree-iterator iter for which **Next**(iter) returns one of the following:
- **IteratorFromState**(state satisfying above specifications for $k_1$ defined in line 3),
- adjacency list, or
- `fail`

**IteratorFromState** *(s := (n, (m_k), k_0, A), S)*

1    Initialize an iterator iter and equip it with

2    • an empty list $H$ to store the produced admissible partial 2-partitions,

3    • an integer $k_1 := \max(\{1\} \cup \{k' < k_0 \mid m_{k'} > 0\}) \geq 1$, and

4    • a function **Next** as defined in line 5

5    **Next** *(iter)*

       `/* find the next block of coatoms of multiplicity k₀:      */`

6      **if** *next $A' = \{A'_1, \ldots, A'_{m_{k_0}}\}$ exists with*     `// find m_{k_0} new coatoms`

7      • *$A \cup A'$ admissible partial 2-partition of level $k_1$*

       `/* the following line guarantees the generation of pairwise`
          `non-isomorphic bipartite graphs, the justification will be`
          `provided in Remark 3.5                                   */`

8      • *the lexicographically minimal element $A''$ in the orbit of $A'$ under $\mathrm{Stab}_{S_n}(A)$ is not contained in $H$*

       `/* Lines 6,7,8 can again be realized by an iterator which`
          `returns the next A' or fail if no such A' exits.        */`

      **then**

9        save $A''$ in $H$

10     $A'' := A \cup A''$     `// augment the current partial 2-partition`

11     **if** $k_1 \geq 2$ **then**

12        $s' := (n, (m_k), k_1, A'')$     `// define the new state`

          `/* return IteratorFromState applied to the new state s' */`

13        **return IteratorFromState**$(s')$

14     **else**

15        **return** $A''$     `// return the complete adjacency list`

16     **else**

17        **return** `fail`

     **return** iter

---

*Remark* 3.5. Line 8 in Algorithm 1 ensures that the iterator iter instantiated by the state $s$ does not create two isomorphic adjacency lists $A''$ and $A''_2$ with a common sublist $A$. Furthermore, the lexicographically minimal element of the orbit of[11] $A \cup A'$ under $\mathrm{Stab}_{S_n}(A)$ is nothing but $A \cup A''$, namely the union of $A$ and the lexicographically minimal element $A''$ of the orbit of $A'$ under $\mathrm{Stab}_{S_n}(A)$ (considered in line 8). This is due to the fact that sets in $A'$ are of different cardinality than those in $A$.

---

[11]$A \cup A'$ is considered in line 7

*Remark* 3.6. A rank 3 simple matroid $M$ (of size $n$) is *weakly atom balanced on dependent flats*, i.e., every atom is contained in at most $\frac{n-1}{2}$ rank 2 flats of cardinality at least 3 (these are all dependent in $M$).

*Proof.* Let $M$ be a simple rank 3 matroid of size $n$ and consider a fixed atom $k$. Let $F_1, \ldots, F_l$ be the rank 2 flats of size at least 3 containing the atom $k$. This implies $|F_i \setminus \{k\}| \geq 2$ for all $1 \leq i \leq l$. Furthermore, by definition of a simple matroid it holds that $F_i \cap F_j = \{k\}$ for all pairs $1 \leq i < j \leq l$. This means that the flats $F_1, \ldots, F_l$ contain each at least two atoms of the $n - 1$ atoms which are different from $k$ and moreover these additional atoms are all pairwise distinct. This immediately yields $l \leq \frac{n-1}{2}$ which proves the claim. $\square$

*Remark* 3.7. The computationally difficult part of Algorithm 1 is to find admissible completions of partial 2-partitions in Line 6. To speed up this part of the computation we use the restriction imposed by Remark 3.6. This means that we can discard atoms in computations of multiplicity $k_0 \geq 3$ if they are already contained in $\frac{n-1}{2}$ flats. This speeds up the generation of rank 3 matroids drastically.

*Remark* 3.8. To calculate lexicographically minimal elements of orbits we use the `Ferret` and `Images` packages, by the third author:

- `Ferret` is a reimplementation of Jeffrey Leon's Partition Backtrack Algorithm [Leo91], with a number of extensions [JPW18].
- `Images` provides algorithms which, given a permutation group $G$ on a set $\Omega$ and a set $S \subseteq \Omega$, find the lexicographically minimal image of $S$ under $G$, or a canonical image of the orbit of $S$ under $G$. `Images` uses the algorithms of Jefferson et al. [JJPW18].

For this project, both `Ferret` and `Images` were extended to be compatible with HPC-GAP.

*Remark* 3.9. The computations to generate all rank 3 simple matroids with integrally splitting roots terminated on all 695 possible multiplicities vectors except for the two vectors $(m_3, m_4, m_5) = (21, 3, 1)$ and $(m_2, m_3, m_4, m_5, m_6) = (1, 23, 1, 0, 1)$. The latter multiplicity vector is in any case not relevant for Terao's conjecture as any matroid with this multiplicity vector would not be flat balanced (cf. Definition 5.1). In Proposition 3.10, we prove that there are no matroids with one of the above multiplicities vectors. Hence, these computations which did not terminate yet do not impose any restrictions on Theorem 1.4 or Table 1.

**Proposition 3.10.** *Let $v_1$ and $v_2$ be the multiplicity vectors $(m_3, m_4, m_5) = (21, 3, 1)$ and $(m_2, m_3, m_4, m_5, m_6) = (1, 23, 1, 0, 1)$ respectively. Then, there exists no simple rank 3 matroid of size 14 having either one of these two multiplicities vectors as multiplicities of its coatoms.*

*Proof.* Given an admissible partial 2-partition $A$ and an atom $e$ we denote by $d_A(e)$ the *deficiency* of $e$ in $A$ which is the number of atoms that are not contained in a common coatom with $e$ in $A$.

For both multiplicity vectors we start our parallel matroid generation algorithm but terminate after completing all levels of size greater than 3. In this way we obtain the admissible partial 2-partitions $A_{1,1} \ldots A_{1,22}$ and $A_{2,1}, A_{2,2}$ for $v_1$ and $v_2$ respectively which are given in Table 2 below. We will argue based on the parity of their deficiencies that all of them cannot be completed to a matroid with the remaining coatoms of size 3 (and one coatom of size 2 in the case of $v_2$) which completes the proof.

First consider the case of the multiplicity vector $v_1$. In this case we can only add coatoms of size 3 to the lists $A_{1,1} \ldots A_{1,22}$ to obtain a matroid. Consider the step that we add the coatom $C := \{e_1, e_2, e_3\}$ to a list $A$ and obtain a new list $A'$. Then we have $d_{A'}(e_i) = d_A(e_i) - 2$ for $1 \le i \le 3$ and the remaining deficiencies remain constant. In particular, the parity of all deficiencies is constant in this step. Since a matroid $M$ has $d_M(e) = 0$ for all atoms $e$, the deficiencies $d_{A_{1,i}}(e)$ need to be even for all atoms $e$. However, by inspection of Table 2 we find that for each partial 2-partitions $A_{1,1} \ldots A_{1,22}$ there is an atom with an odd deficiency which proves that there exists no matroid with multiplicity vector $v_1$.

In the case of the multiplicity vector $v_2$ we need to add coatoms of size 3 and exactly one coatom of size 2 to the admissible partial 2-partitions $A_{2,1}, A_{2,2}$. An analogous argument as in the first case shows that number of atoms with odd deficiency of the lists $A_{2,1}, A_{2,2}$ must be exactly two. Again by inspection of Table 2 we find that the lists $A_{2,1}, A_{2,2}$ have one and zero atoms with odd deficiency respectively which proves that there exists no matroid with multiplicity vector $v_2$.

| | admissible partial 2-partition | atoms with odd deficiencies |
|---|---|---|
| $A_{1,1}$ | $[[1,2,3,4,5],[1,6,7,8],[1,9,10,11],[1,12,13,14]]$ | $2,3,4,5$ |
| $A_{1,2}$ | $[[1,2,3,4,5],[1,6,7,8],[1,9,10,11],[2,6,9,12]]$ | $1,3,4,5,6,9$ |
| $A_{1,3}$ | $[[1,2,3,4,5],[1,6,7,8],[1,9,10,11],[2,6,12,13]]$ | $1,3,4,5,6$ |
| $A_{1,4}$ | $[[1,2,3,4,5],[1,6,7,8],[1,9,10,11],[2,12,13,14]]$ | $1,3,4,5$ |
| $A_{1,5}$ | $[[1,2,3,4,5],[1,6,7,8],[1,9,10,11],[6,9,12,13]]$ | $1,2,3,4,5,6,9$ |
| $A_{1,6}$ | $[[1,2,3,4,5],[1,6,7,8],[1,9,10,11],[6,12,13,14]]$ | $1,2,3,4,5,6$ |
| $A_{1,7}$ | $[[1,2,3,4,5],[1,6,7,8],[2,6,9,10],[3,6,11,12]]$ | $4,5$ |
| $A_{1,8}$ | $[[1,2,3,4,5],[1,6,7,8],[2,6,9,10],[3,7,9,11]]$ | $4,5,6,7,9$ |
| $A_{1,9}$ | $[[1,2,3,4,5],[1,6,7,8],[2,6,9,10],[3,7,11,12]]$ | $4,5,6,7$ |
| $A_{1,10}$ | $[[1,2,3,4,5],[1,6,7,8],[2,6,9,10],[3,11,12,13]]$ | $4,5,6$ |
| $A_{1,11}$ | $[[1,2,3,4,5],[1,6,7,8],[2,6,9,10],[6,11,12,13]]$ | $3,4,5$ |
| $A_{1,12}$ | $[[1,2,3,4,5],[1,6,7,8],[2,6,9,10],[7,9,11,12]]$ | $3,4,5,6,7,9$ |
| $A_{1,13}$ | $[[1,2,3,4,5],[1,6,7,8],[2,6,9,10],[7,11,12,13]]$ | $3,4,5,6,7$ |
| $A_{1,14}$ | $[[1,2,3,4,5],[1,6,7,8],[2,6,9,10],[11,12,13,14]]$ | $3,4,5,6$ |
| $A_{1,15}$ | $[[1,2,3,4,5],[1,6,7,8],[2,9,10,11],[3,12,13,14]]$ | $4,5$ |
| $A_{1,16}$ | $[[1,2,3,4,5],[1,6,7,8],[2,9,10,11],[6,9,12,13]]$ | $3,4,5,6,9$ |
| $A_{1,17}$ | $[[1,2,3,4,5],[1,6,7,8],[2,9,10,11],[6,12,13,14]]$ | $3,4,5,6$ |
| $A_{1,18}$ | $[[1,2,3,4,5],[1,6,7,8],[6,9,10,11],[6,12,13,14]]$ | $2,3,4,5$ |
| $A_{1,19}$ | $[[1,2,3,4,5],[1,6,7,8],[6,9,10,11],[7,9,12,13]]$ | $2,3,4,5,6,7,9$ |
| $A_{1,20}$ | $[[1,2,3,4,5],[1,6,7,8],[6,9,10,11],[7,12,13,14]]$ | $2,3,4,5,6,7$ |
| $A_{1,21}$ | $[[1,2,3,4,5],[1,6,7,8],[6,9,10,11],[9,12,13,14]]$ | $2,3,4,5,6,9$ |
| $A_{1,22}$ | $[[1,2,3,4,5],[6,7,8,9],[6,10,11,12],[7,10,13,14]]$ | $1,2,3,4,5,6,7,10$ |
| $A_{2,1}$ | $[[1,2,3,4,5,6],[1,7,8,9]]$ | $1$ |
| $A_{2,2}$ | $[[1,2,3,4,5,6],[7,8,9,10]]$ | |

**Table 2.** The partial 2-partitions considered in the proof of Proposition 3.10.

$\square$

## 4. HOW TO DECIDE REPRESENTABILITY OF A MATROID?

The Basis Extension Theorem for matroids (cf. Remark 2.2) implies that the (possibly empty) space $\mathcal{R}(M)$ of *all* representations (over some unspecified field $\mathbb{F}$) of a matroid $M = (E, \mathcal{F})$ is an *affine*, namely an affine subvariety $V(I') = \mathbb{A}_{\mathbb{Z}}^{rn+1}$, where $r$ the rank of $M$ and $n$ its number of atoms.

More precisely, let $\mathbb{A}_{\mathbb{Z}}^{rn+1} := \operatorname{Spec} R[d]$, where $R := \mathbb{Z}[a_{ij} \mid i = 1, \ldots, r, \; j = 1, \ldots, n]$. To describe the ideal $I'$ set $A := (a_{ij}) \in R^{r \times n}$. For a subset $S \subset E$ denote by $A_S$ the submatrix of $A$ with columns in $S$. Further, let $\mathcal{B}(M) = \{B_1, \ldots, B_b\}$ be the set of bases

of $M$. Then

$$I' = \langle \det(A_D) \mid D \subset E \text{ dependent}, |D| = r \rangle + \left\langle 1 - d \prod_{B \in \mathcal{B}(M)} \det(A_B) \right\rangle \trianglelefteq R[d].$$

It follows that the $M$ is representable (over some field $\mathbb{F}$) if and only if $1 \notin I'$. This ideal membership problem can be decided by computing a Gröbner basis of $I'$. This is basically the algorithm suggested in [Oxl11].

If the ideal $I'$ is a maximal ideal in $R[d]$ the moduli space of representations $\operatorname{Spec} R[d]/I'$ of the matroid $M$ contains only one point. In this case, the matroid $M$ has only a unique representation (up to equivalence) and we call $M$ *uniquely representable over* $\mathbb{Z}$.

However, it is computationally more efficient to represent $\mathcal{R}(M)$ as a quasi-affine set $V(I) \setminus V(J) \subset \mathbb{A}^{rn} = \operatorname{Spec} R$, where $J$ is a principal ideal. Denote by $J_S := \langle \det(A_S) \rangle$ the principal ideal generated by the maximal minor corresponding to $S$, provided $|S| = r$. Then

$$I = \sum \{ J_D \mid D \subset E \text{ dependent}, |D| = r \},$$
$$J = \prod \{ J_B \mid B \in \mathcal{B}(M) \}.$$

In particular, $J$ is a principal ideal. It follows that $M$ is representable (over some field $\mathbb{F}$) iff $\det(A_S) \notin \sqrt{I}$ for all $S \subset E$ basis. The ideal $I$ can be replaced by the saturation

$$\widetilde{I} := I : \left( \prod_{B \subset E \text{ basis}} \det(A_B) \right)^\infty = I : \det(A_{B_1})^\infty : \cdots : \det(A_{B_b})^\infty.$$

Then $M$ is representable iff $1 \notin \widetilde{I}$.

We used a more efficient approach which does not involve working over $\mathbb{A}_{\mathbb{Z}}^{rn+1}$ but fixes certain values of the matrix $A$ to $0$ or $1$ as described in [Oxl11, p. 184]. Firstly, we choose a basis $B \in \mathcal{B}(M)$ and fix the corresponding submatrix $A_B$ to be the unit matrix. Without loss of generality, we can for simplicity assume $B = \{1, \ldots, r\}$. Secondly, we consider the fundamental circuits with respect to this basis $B$, i. e. for each $k \in E \setminus B$ let $C(k, B)$ be the unique circuit of the matroid $M$ contained in $B \cup k$. The entries of $A$ in the column $k \in E \setminus B$ which do not appear in $C(k, B)$ can be fixed to $0$. Lastly, the first entry in every column and the first entry in every row of $A$ can be taken as $1$ by column and row scaling respectively. We have added this algorithm to `alcove` [Leu19].

For another approach to the rational moduli space cf. [Cun11].

## 5. PROOF OF THEOREM 1.4

Before proving Theorem 1.4 we need the following definition.

**Definition 5.1.** Let $M$ be a simple matroid of rank $3$ and assume $\chi_M(t) = (t-1)(t-a)(t-b)$ for some integers $a, b \in \mathbb{Z}$ such that $a \leq b$.

- We call $M$ **atom balanced** if any atom is contained in at most $a$-many flats.
- We call $M$ **flat balanced** if any flat contains strictly less than $a$-many atoms.
- If $M$ is both atom and flat balanced we call it **strongly balanced**.

The importance of balancedness in our context stems from the next proposition.

**Proposition 5.2.** *Let $M$ be a simple matroid of rank $3$ and assume $\chi_M(t) = (t-1)(t-a)(t-b)$ for some integers $a, b \in \mathbb{Z}$ such that $a \leq b$. If $M$ is* not *strongly balanced then the freeness of any arrangement of hyperplanes representing $M$ can be decided combinatorially. These representations therefore satisfy Terao's freeness conjecture.*

*Proof.* To begin assume that $M$ is not atom balanced for some atom $A$ which is contained in $n_{M,A}$ many flats with $n_{M,A} > a$. Then, Theorem 1.1 and Corollary 1.2 in [Abe14] show that any representation of $M$ is free if and only if $n_{M,A} \in \{a + 1, b + 1\}$.

Instead assume that $M$ is not flat balanced. In this case, Lemma 2.10 in [ACKN16] shows that $M$ cannot be atom balanced either which finishes the proof by the first part.   □

Now we have all ingredients to prove Theorem 1.4.

*Proof of Theorem 1.4.* It suffices to check Terao's conjecture for all representations of matroids of size 14 which do not fall into any of the following classes of arrangements for which Terao's conjecture is known to be true:

- If the characteristic polynomial of the arrangement does not factor over the integers the arrangement is combinatorially non-free by Terao's Factorization Theorem [Ter81].
- Representations of non-strongly balanced simple rank 3 matroids satisfy Terao's conjecture by Proposition 5.2.
- Any representation of an inductively free matroid is a free arrangement.
- If a matroid has a unique representation over the integers[12] it trivially satisfies Terao's conjecture.

The query of the database [BK19] of integrally split, rank 3 matroids of size 14 which are strongly balanced, not inductively free, and representable but not uniquely representable over the integers does not return any matroid. This completes the proof of Terao's conjecture for arrangements of rank 3 with 14 hyperplane.   □

*Remark* 5.3. The situation of matroids of size 14 is surprisingly simple in that respect. This is not the case for matroids of smaller size since there are examples which avoid all of the above classes and exhibit a non-trivial moduli space of representations (among them the example of a free but not rigid arrangement of size 13 described in [ACKN16]). We will describe their moduli spaces over $\operatorname{Spec} \mathbb{Z}$ and the free locus therein in a subsequent article which will establish Terao's conjecture for arrangements of rank 3 with up to 14 hyperplanes in any characteristic.

---

[12]i.e., the moduli space $\operatorname{Spec} R/\widetilde{I} \to \operatorname{Spec} \mathbb{Z}$ of representations is $\operatorname{Spec} \mathbb{F}_p \to \operatorname{Spec} \mathbb{Z}$, a singleton.

## Appendix A. Rooted trees

A finite **rooted forest** (or set of **rooted trees**) can be understood as a finite **sequential inverse system** (in the category) of finite sets, i.e., a sequence of the form

$$T_\bullet : T_0 \xleftarrow{\varphi_1} T_1 \xleftarrow{\varphi_2} T_2 \xleftarrow{\varphi_3} \cdots \xleftarrow{\varphi_d} T_d,$$

where $T_i$ is the finite set of **vertices** of **depth** $i$. We call $d$ then **depth** of $T_\bullet$. In particular, $T_0$ is the set of **roots**. The (inverse) limit $T := \lim T_\bullet$ can be naturally identified with the set of **leaves** of $T_\bullet$, i.e., the set of non-images in $T_\bullet$. As mentioned in the introduction we then say that $T_\bullet$ **classifies** $T$.

**Convention.** Without loss of generality[13] we will henceforth assume $T_\bullet$ to be a rooted tree of depth $d$, i.e., $T_0 = \{*\}$ a singleton.

If all maps in the inverse system are surjective then the natural map $T_d \leftarrow T$ (which is part of the limit datum) is bijective and the set leaves $T = T_d$. In this case all leaves have the same depth $n$ and we call $T_\bullet$ **uniform (of depth** $d$**)**.

More generally, we call a tree $T_\bullet$ **locally uniform** if each vertex that has a leaf as a child only has leaves as children, i.e., if for each vertex $v$ of depth $i$ the following holds: $\varphi_i^{-1}(v) \cap T \neq \emptyset \implies \varphi_i^{-1}(v) \subset T$.

Many inequivalent representations of such rooted trees classifying the same set $T$ might exist: Examples A.1 and A.2 are inequivalent families of rooted trees $T_\bullet^{(n)}$ (indexed by a natural number $n$) classifying the same family of sets $T^n$ of cardinality $C_n$, the $n$-th Catalan number.

**Example A.1** (Matched parentheses)**.** For $i \in \mathbb{N}$ denote by $T_i$ the set containing $i+1$ pairs of correctly matched parentheses:

$$T_0 := \{()\}, T_1 := \{(()), ()()\}, T_2 := \{()(()), (()()), ((())), (())(), ()()()\}, \ldots$$

Define $T_{i-1} \xleftarrow{\varphi_i} T_i$ to be the map removing the left most[14] pair of parentheses containing no other ones. For a fixed $n \in \mathbb{N}$ the sequence $T_\bullet : T_0 \xleftarrow{\varphi_1} T_1 \xleftarrow{\varphi_2} T_2 \xleftarrow{\varphi_3} \cdots \xleftarrow{\varphi_{n-1}} T_{n-1}$ is a finite rooted tree of uniform depth $n-1$. The cardinality of the set of leaves $\lim T_\bullet = T_{n-1}$ is the $n$-th Catalan number[15] $C_n = \binom{2n}{n} - \binom{2n}{n+1} = \frac{1}{n+1}\binom{2n}{n}$.

**Example A.2** (Magma evaluation)**.** For $n \in \mathbb{N}_{>0}$ denote by $T^{(n)}$ the set of all possible ways to evaluate the product of the sorted list of free generators of a free magma $M_n = \langle a_0, \ldots, a_n \rangle$ of rank $n+1$:

| $n$ | 1 | 2 | 3 |
|---|---|---|---|
| $M_n$ | $\langle a, b \rangle$ | $\langle a, b, c \rangle$ | $\langle a, b, c, d \rangle$ |
| $T^{(n)}$ | $\{ab\}$ | $\{(ab)c, a(bc)\}$ | $\{((ab)c)d, (a(bc))d, (ab)(cd), a((bc)d), a(b(cd))\}$ |

---

[13]A forest of rooted trees can be understood as a single rooted tree by adding a constant map $T_{-1} := \{*\} \leftarrow T_0$ and then increase all indices by 1.

[14]or right most, ...

[15]Cf. (http://oeis.org/A000108).

The set $T_i^{(n)}$ for $i \in \mathbb{N}$ arises from $T^{(n)}$ by deleting all pairs of parentheses of depth higher than $i$. The maps $T_{i-1}^{(n)} \xleftarrow{\varphi_i} T_i^{(n)}$ are evident.

| $n$ | 1 | 2 | 3 |
|---|---|---|---|
| $T_0^{(n)}$ | $\{ab\}$ | $\{abc\}$ | $\{abcd\}$ |
| $T_1^{(n)}$ | | $\{(ab)c, a(bc)\}$ | $\{(abc)d, (ab)(cd), a(bcd)\}$ |
| $T_2^{(n)}$ | | | $\{((ab)c)d, (a(bc))d, a((bc)d), a(b(cd))\}$ |

The gray entries in the above table are the internal nodes of the rooted tree $T_\bullet^{(n)}$. The latter is not locally uniform for $n \geq 3$. The set of leaves $\lim T_\bullet^{(n)}$ coincides with $T^{(n)}$, by construction. The cardinality of $T^{(n)}$ is again the $n$-th Catalan number $C_n$.

In the following example the sets of leaves are themselves sets of rooted trees. We hope this does not cause confusion.

**Example A.3** (Phylogenetic trees with labeled leaves). A phylogenetic tree is a labeled rooted tree. A phylogenetic tree with $n \in \mathbb{N}_{>0}$ leaves corresponds to a total partition of $n$. Let $T^{(n)}$ be the set of phylogenetic trees with $n$ (labeled) leaves.[16]

| $n$ | 1 | 2 | 3 |
|---|---|---|---|
| $T^{(n)}$ | $\big\{\{1\}\big\}$ | $\big\{\{\{1\}, \{2\}\}\big\}$ | $\big\{\{\{1\}, \{2\}, \{3\}\}; \{\{1\}, \{\{2\}, \{3\}\}\}; \{\{2\}, \{\{1\}, \{3\}\}\}; \{\{3\}, \{\{1\}, \{2\}\}\}\big\}$ |

Truncating a phylogenetic tree at depth $i$ means to contract all edges below depth $i$ and multi-label the new leaves at depth $i$ by all their child leaves. For $i \in \mathbb{N}$ denote by $T_i^{(n)}$ the set of all truncations of trees in $T^{(n)}$ at depth $i$. Again, all maps $T_{i-1}^{(n)} \xleftarrow{\varphi} T_i^{(n)}$ are evident.

| $n$ | 1 | 2 | 3 |
|---|---|---|---|
| $T_0^{(n)}$ | $\big\{\{1\}\big\}$ | $\big\{\{1, 2\}\big\}$ | $\big\{\{1, 2, 3\}\big\}$ |
| $T_1^{(n)}$ | | $\big\{\{\{1\}, \{2\}\}\big\}$ | $\big\{\{\{1\}, \{2\}, \{3\}\}; \{\{1\}, \{2, 3\}\}; \{\{2\}, \{1, 3\}\}; \{\{3\}, \{1, 2\}\}\big\}$ |
| $T_2^{(n)}$ | | | $\big\{\{\{1\}, \{\{2\}, \{3\}\}\}; \{\{2\}, \{\{1\}, \{3\}\}\}; \{\{3\}, \{\{1\}, \{2\}\}\}\big\}$ |

The rooted tree $T_\bullet^{(n)}$ is not locally uniform for $n \geq 3$. The set of leaves $\lim T_\bullet^{(n)}$ coincides with $T^{(n)}$, by construction.

Factoring out symmetries of rooted trees again yields rooted trees:

*Remark* A.4 (Rooted trees of group orbits). Let $G$ be a group. A rooted tree $T_\bullet$ is called a **rooted $G$-tree** if each $T_i$ is a $G$-set and all maps $\varphi_i$ are $G$-equivariant. A rooted $G$-tree $\lim T_\bullet$ induces a rooted tree of orbits $T_\bullet/G$. Furthermore $\lim(T_\bullet/G) = \lim(T_\bullet)/G$, naturally.

**Example A.5** (Phylogenetic trees with nonlabeled leaves). Applying Remark A.4 to the previous Example A.3 yields a rooted tree classifying phylogenetic trees with unlabeled leaves. More precisely, the action of $S_n$ on $\{1, \ldots, n\}$ turns the rooted tree $T_\bullet$ in Example A.3 into a rooted $S_n$-tree. The rooted tree of orbits $T_\bullet/S_n$ then classifies $T/S_n$ which is the set of phylogenetic trees with unlabeled leaves.[17]

Our primary family of examples of rooted tree was discussed in Section 2. They have rank 3 matroids as their set of leaves.

---

[16]Cf. (http://oeis.org/A000311).

[17]Cf. (http://oeis.org/A000669).

APPENDIX B. RECURSIVE ITERATORS, TREE-ITERATORS: DEFINITIONS AND PARALLELIZED EVALUATION

**Definition B.1.** Let $T$ be a set.

- A **recursive iterator** $t$ **in** $T$ is an iterator which upon popping produces a **descendent** $\text{Next}(t)$ which is either
  (a) a new recursive iterator in $T$,
  (b) an element of $T$, or
  (c) $\texttt{fail} \notin T$.
  If the pop result $\text{Next}(t)$ is $\texttt{fail}$ then any subsequent pop result of $t$ remains $\texttt{fail}$.
- A **full evaluation** of a recursive iterator recursively pops all recursive iterators until each of them pops $\texttt{fail}$.
- If $t$ is a recursive iterator then the subset of elements $T(t) \subset T$ produced upon full evaluation is called the **set of leaves** of $t$.
- A recursive iterator is called **locally uniform** if every descendent either pops recursive iterators or leaves, exclusively (if not $\texttt{fail}$).
- A recursive iterator $t$ in $T$ is called a **tree-iterator** if upon full evaluation each element of $T(t)$ is the pop result of exactly one descendent of $t$.

Algorithm 2 gets as input a recursive iterator, a number $n$ of workers, and a global FIFO $L$. It initializes a shared priority queue[18] $P$, adds $P$ as the only job with priority 0, triggers $n$ workers (running in threads) each executing Algorithm 3. If a worker produces a leaf it writes it to the FIFO $L$.

We use a priority queue to schedule units of work in our algorithm. Unlike regular queues, having a computable priority (which can be drawn from any partially or totally ordered set) to govern order of processing allows us to easily define for each job when it is to be executed in relation to other jobs without requiring a more complex scheduler, as long as we can compute a priority as a function from the set of jobs to the set of priorities that encodes the necessary relation between jobs. As discussed in Section 3, we use a simple shared memory implementation for our shared priority queue.

---

**Algorithm 2:** ParallelyEvaluateRecursiveIterator

**Input:**
- A recursive iterator $t$
- a number $n \in \mathbb{N}_{>0}$ of workers
- a global FIFO $L = ()$, accessible by the subprocesses of the workers

**Output:** no return value; the side effect is to fill the FIFO $L$ with the leaves in $T(t)$

**ParallelyEvaluateRecursiveIterator** *(t, n, L)*

1     Initialize a farm $w$ of $n$ workers $w_1, \ldots, w_n$
2     Initialize a *shared* priority queue $P$ of iterators and set $P = ()$
3     Initialize a *shared* counter $j$ of jobs in process and pending and set $j = 1$
4     Initialize a *shared* semaphore $s \geq 0$ and set $s = 0$
5     $P := ((t, 0))$
6     **for** $i = 1, \ldots, n$ **do**
7        $\texttt{EvaluateRecursiveIterator}(n, L, P, s, j)$ within worker $w_i$
8     $\texttt{SignalSemaphore}(s)$
9     **return** *none*

---

[18]The priority queue $P$ stores recursive iterators. We define their priority in $P$ to be their nesting level with respect to the first recursive iterator $t$.

Algorithm 3 is the one executed by each worker. It gets the global state consisting of the number $n$ of workers, the FIFO $L$ of leaves, the priority queue $P$, the semaphore $s$, and the counter $j$ of jobs in process or pending. A semaphore is a globally shared variable with nonnegative integers as admissible values, which we use to tell workers when to start looking for jobs to process. The command $\texttt{SignalSemaphore}(s)$ increases $s$ by 1. The command $\texttt{WaitSemaphore}(s)$ halts until $s > 0$ and then decreases $s$ by 1.

---

**Algorithm 3:** EvaluateRecursiveIterator

**Input:**
- a number $n \in \mathbb{N}_{>0}$ of all workers
- a global FIFO $L = ()$, accessible by the other $n - 1$ workers
- a shared priority queue $P$
- a shared semaphore $s$
- a shared counter $j$ of jobs in process or pending

**Output:** no return value; the side effect is to evaluate the recursive iterators in the priority queue which get processed by this worker and save the eventually leaves in the FIFO $L$

**EvaluateRecursiveIterator** *(n, L, P, s, j)*

```
1    while true do
2        WaitSemaphore(s)                    // wait until the semaphore s > 0
3        if P = () then                      // if the priority queue is empty
4            return none                              // terminate the worker
5        (tᵢ, pₜᵢ) := Pop(P)          // get the highest priority job from P
6        rᵢ := Next(tᵢ)                      // pop the recursive iterator tᵢ
7        if rᵢ ∈ T then                         // the result rᵢ is a leaf
8            Add(L, rᵢ)               // add the leaf rᵢ to the FIFO L of leaves
9            Add(P, (tᵢ, pₜᵢ))     // return the recursive iterator tᵢ back to P
10       elif rᵢ ≠ fail then           // the result rᵢ is a recursive iterator
11           Add(P, (tᵢ, pₜᵢ))     // return the recursive iterator tᵢ back to P
12           SignalSemaphore(s)              // increase the semaphore by 1
13           Add(P, (rᵢ, pₜᵢ + 1))    // add the new recursive operator rᵢ to P
14           SignalSemaphore(s)              // increase the semaphore by 1
15           j := j + 1                  // increase the job counter j by 1
16       else                                // the result rᵢ is fail
17           j := j − 1                  // decrease the job counter j by 1
18       if j = 0 then  // no recursive iterator is in in process or pending
19           Add(L, fail)              // add fail to the FIFO L of leaves
20           for i = 1, . . . , n do                        // for each worker
21               SignalSemaphore(s)          // increase the semaphore by 1
                 /* the first worker who realizes that there are no jobs
                    left writes fail in the FIFO L of leaves and increases
                    the semaphore by n to enable all workers to bypass
                    line 2, reach line 4 and terminate                    */
```

Here the subscripts are: $(t_i, p_{t_i}) := \texttt{Pop}(P)$, $r_i := \texttt{Next}(t_i)$, $\texttt{Add}(L, r_i)$, $\texttt{Add}(P, (t_i, p_{t_i}))$, $\texttt{Add}(P, (r_i, p_{t_i} + 1))$.

---

Algorithm 3 could be refined for *locally uniform* recursive iterators as follows: Whenever a recursive iterator starts to evaluate leaves then do not add it back to the priority queue (line 9) but evaluate it fully (by repeating lines 6 and 8).

In Algorithms 2 and 3 the FIFO $L$ can be equipped with a capacity $k$. Once this capacity is reached line 8 of Algorithm 3 will automatically pause the worker until some other process, e.g. Algorithm 4, pops the FIFO $L$.

Algorithm 4 turns a tree-iterator $t$ in $T$ into a single iterator $\ell(t)$ which enumerates $T(t)$. Whenever $t$ is a tree-iterator then we call $\ell(t)$ the associated **leaf-iterator**.

---

**Algorithm 4:** LeafIterator (Leaf-iterator of a tree-iterator)

**Input:**
- A tree-iterator $t$
- a number $n \in \mathbb{N}_{>0}$ of workers

**Output:** The associated leaf-iterator $\ell(t)$

**LeafIterator** *(t, n, k)*

1    Initialize a FIFO $L := ()$
2    Trigger **ParallelyEvaluateRecursiveIterator**$(t, n, L)$
3    Initialize the leaf-iterator $\ell$:
4        Define $\texttt{IsDone}(\ell)$ to check if first entry of $L$ is $\texttt{fail}$[a]
5        Define $\texttt{Next}(\ell)$ to return the first entry of $L$ which is an element of $T(t)$
6    **return** $\ell$

---

[a]Recall, $\texttt{fail} \notin T$.

## APPENDIX C. WHY HPC-GAP?

We list some advantages of our implementation HPC-GAP:

 (a) More threads can be added on the fly; they simply start to pull jobs from the priority queue (if non-empty);
 (b) One can even notify single threads to terminate once they finish evaluating a recursive-iterator;
 (c) HPC-GAP supports global shared memory and therefore allows us to use a simple and efficient shared memory implementation for priority queues, as described in Section B;
 (d) HPC-GAP allows for objects to be moved efficiently from one thread to another by reassigning ownership of those objects to the new thread, rather than inefficiently performing a full structural copy or using serialization.

The most obvious drawback of our implementation is the following: The state of evaluation of a recursive iterator is defined by the priority queue (residing in a shared region) and by the iterators that are being evaluated in the threads. So if a thread dies or hangs[19] while evaluating a recursive-iterator then the latter (which was adopted by the thread from the priority queue) with all its leaves (e.g., matroids) are lost. In particular, it is impossible to terminate the running HPC-GAP process without losing the state of evaluation.

A second drawback is that it is currently impossible to use a distributed computational model since in our implementation the state of evaluation of a recursive iterator can only be defined and managed by a single HPC-GAP process.

One way to avoid these drawbacks is to store the state of evaluation into a (temporary) database. In particular *all* yet non-fully evaluated recursive-iterators should be stored in the database, while those in process should be marked as such using a unique fingerprint of the evaluating process. This allows a distributed access on the one side. On the other side an

---

[19]either manually terminated or due to an instability of HPC-GAP, which rarely happens in the current version

iterator with a deadlock can be manually (or maybe even automatically by a watchdog) be freed for evaluation by other threads searching for jobs.

Our implementation performs best for recursive-iterators where the evaluation time of each produced iterator is considerably longer than the organizational overhead in HPC-GAP caused by by redefining regions, etcetera.

## APPENDIX D. TIMINGS

It is worth noting that $97\%$ of the $404$ tree iterators of the different multiplicity vectors for $n = 13$ atoms can be evaluated in less than a day of CPU time. For $n = 14$ the corresponding number are still $93\%$ of $695$.

*Remark* D.1. While processing all relevant multiplicity vectors is an "embarrassingly parallel" problem, the reader may have noticed that the parallel evaluation of a single tree-iterator corresponding to one such multiplicity vector is much more involved.

The gain of the parallelized evaluation of tree iterators of rank $3$ matroids with given multiplicity vector depends on the number $n$ of atoms. The longest CPU time of an evaluation of a tree iterator with $n = 13$ atoms was that of the one with multiplicity vector $(m_3, m_4) = (18, 4)$ which took $16.2$ CPU days but finished in $5.59$ days using $8$ workers, a factor of $2.9$. The gain for $n = 14$ was more significant. The multiplicity vector with the largest number of matroids is $(m_2, m_3, m_4, m_5) = (14, 9, 5, 2)$. It generates $168352$ matroids ($45$ of them are representable) in about $22.8$ hours of CPU time but finished in $112$ minutes using $24$ workers, a factor of $12.2$. The longest CPU time of an evaluation of a tree iterator with $n = 14$ atoms was that of the one with multiplicity vector $(m_2, m_3, m_4, m_5) = (3, 18, 4, 1)$ which took $495.7$ CPU days but finished in $74.3$ days using $8$ workers, a factor of $6.7$.

## REFERENCES

[Abe14]    Takuro Abe, *Roots of characteristic polynomials and intersection points of line arrangements*, J. Singul. **8** (2014), 100–116. MR 3395241 14

[Abe16]    Takuro Abe, *Divisionally free arrangements of hyperplanes*, Invent. Math. **204** (2016), no. 1, 317–346. MR 3480558 6, 7

[ACKN16]   T. Abe, M. Cuntz, H. Kawanoue, and T. Nozawa, *Non-recursive freeness and non-rigidity*, Discrete Math. **339** (2016), no. 5, 1430–1449. MR 3475556 4, 14

[BB99]     Anton Betten and Dieter Betten, *Linear spaces with at most 12 points*, J. Combin. Des. **7** (1999), no. 2, 119–145. MR 1670277 2

[BK19]     Mohamed Barakat and Lukas Kühne, `matroids_split_public` – *a database collection for rank* $3$ *integrally split simple matroids*, 2014–2019, (https://github.com/homalg-project/MatroidGeneration/blob/master/README.md). 2, 9, 14

[Bry72]    Thomas H. Brylawski, *A decomposition for combinatorial geometries*, Trans. Amer. Math. Soc. **171** (1972), 235–282. MR 0309764 2

[Cun11]    Michael Cuntz, *Minimal fields of definition for simplicial arrangements in the real projective plane*, Innov. Incidence Geom. **12** (2011), 12. MR 2942717 13

[DIM17]    A. Dimca, D. Ibadula, and A. Macinic, *Freeness and near freeness are combinatorial for line arrangements in small degrees*, ArXiv e-prints (2017). 4

[DK98]     F. M. Dong and K. M. Koh, *Non-chordal graphs having integral-root chromatic polynomials*, Bull. Inst. Combin. Appl. **22** (1998), 67–77. MR 1489869 6

[dMN05]    Anna de Mier and Marc Noy, *On matroids determined by their Tutte polynomials*, Discrete Math. **302** (2005), no. 1-3, 52–76. MR 2179636 2

[DW89]     Andreas W. M. Dress and Walter Wenzel, *Geometric algebra for combinatorial geometries*, Adv. Math. **77** (1989), no. 1, 1–36. MR 1014071 4

[FV14]     Daniele Faenzi and Jean Vallès, *Logarithmic bundles and line arrangements, an approach via the standard construction*, J. Lond. Math. Soc. (2) **90** (2014), no. 3, 675–694. MR 3291795 4

[GAP18]    The GAP Group, *GAP – Groups, Algorithms, and Programming, Version 4.9.2*, 2018, (http://www.gap-system.org). 9

[HR15]     Torsten Hoge and Gerhard Röhrle, *On inductively free reflection arrangements*, J. Reine Angew. Math. **701** (2015), 205–220. MR 3331732 3

[JJPW18]   Christopher Jefferson, Eliza Jonauskyte, Markus Pfeiffer, and Rebecca Waldecker, *Minimal and canonical images*, Journal of Algebra (2018). 11

[JPW18]    Christopher Jefferson, Markus Pfeiffer, and Rebecca Waldecker, *New refiners for permutation group search*, Journal of Symbolic Computation (2018). 11

[JT84]     Michel Jambu and Hiroaki Terao, *Free arrangements of hyperplanes and supersolvable lattices*, Adv. in Math. **52** (1984), no. 3, 248–258. MR 744859 7

[KR11]     Joseph P. S. Kung and Gordon F. Royle, *Graphs whose flow polynomials have only integral roots*, European J. Combin. **32** (2011), no. 6, 831–840. MR 2821554 6

[Leo91]    Jeffrey S. Leon, *Permutation group algorithms based on partitions. I. Theory and algorithms*, J. Symbolic Comput. **12** (1991), no. 4-5, 533–583, Computational group theory, Part 2. MR 1146516 11

[Leu19]    Martin Leuner, `alcove` – *algebraic combinatorics package for* GAP, 2013–2019, (`https://github.com/martin-leuner/alcove`). 2, 13

[MMIB12a]  Yoshitake Matsumoto, Sonoko Moriyama, Hiroshi Imai, and David Bremner, *Database of matroids*, 2012, (`http://www-imai.is.s.u-tokyo.ac.jp/~ymatsu/matroid/index.html`). 2

[MMIB12b]  Yoshitake Matsumoto, Sonoko Moriyama, Hiroshi Imai, and David Bremner, *Matroid enumeration for incidence geometry*, Discrete Comput. Geom. **47** (2012), no. 1, 17–43. MR 2886089 2

[Mü17]     Paul Mücksch, *Recursively free reflection arrangements*, J. Algebra **474** (2017), 24–48. MR 3595783 3

[Oxl11]    James Oxley, *Matroid theory*, second ed., Oxford Graduate Texts in Mathematics, vol. 21, Oxford University Press, Oxford, 2011. MR 2849819 7, 13

[Sta72]    R. P. Stanley, *Supersolvable lattices*, Algebra Universalis **2** (1972), 197–217. MR 0309815 6

[Ter81]    Hiroaki Terao, *Generalized exponents of a free arrangement of hyperplanes and Shepherd-Todd-Brieskorn formula*, Invent. Math. **63** (1981), no. 1, 159–179. MR 608532 14

DEPARTMENT OF MATHEMATICS, UNIVERSITY OF SIEGEN, 57068 SIEGEN, GERMANY
*E-mail address*: mohamed.barakat@uni-siegen.de

DEPARTMENT OF MATHEMATICS, UNIVERSITY OF KAISERSLAUTERN, 67653 KAISERSLAUTERN, GERMANY
*E-mail address*: behrends@gmail.com

SCHOOL OF COMPUTER SCIENCE, UNIVERSITY OF ST ANDREWS, KY16 9SX ST ANDREWS, UNITED KINGDOM
*E-mail address*: aj21@st-andrews.ac.uk

EINSTEIN INSTITUTE OF MATHEMATICS, THE HEBREW UNIVERSITY OF JERUSALEM, 9190401 JERUSALEM, ISRAEL
*E-mail address*: lf.kuehne@gmail.com

LEHRSTUHL B FÜR MATHEMATIK, RWTH AACHEN UNIVERSITY, GERMANY
*E-mail address*: martin.leuner@rwth-aachen.de