

**Max-Planck-Institut
für Mathematik
in den Naturwissenschaften
Leipzig**

**On the Generation of Rank 3 Simple
Matroids with an Application to
Terao's Freeness Conjecture**

by

*Mohamed Barakat, Reimer Behrends,
Christopher Jefferson, Lukas Kühne, and
Martin Leuner*

Preprint no.: 88

2020



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

MOHAMED BARAKAT, REIMER BEHRENDTS, 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 an integrally 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 (cf. Remark 2.11). Recognized patterns and questions which a database answers affirmatively may lead to working hypotheses or even proofs by inspection (cf. Theorem 1.3).

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 (cf. Appendix A). In other words, we embed T as the set of leaves in the bigger set of vertices $V(T_\bullet)$. 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. To iterate over the vertices of the rooted tree T_\bullet we introduce the data structure of a tree-iterator t (cf. Definition B.1).

In this article we will describe how to use tree-iterators to classify all non-isomorphic simple rank 3 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, \dots, m_{n-1}) the **multiplicity vector** of M where m_k is

2010 *Mathematics Subject Classification.* 05B35, 52C35, 32S22, 68R05, 68W10.

Key words and phrases. rank 3 simple matroids, integrally splitting characteristic polynomial, Terao's freeness conjecture, recursive iterator, tree-iterator, leaf-iterator, iterator of leaves of rooted tree, priority queue, parallel evaluation of recursive iterator, noSQL database, ArangoDB.

This work is a contribution to Project II.1 of SFB-TRR 195 'Symbolic Tools in Mathematics and their Application' funded by Deutsche Forschungsgemeinschaft (DFG). The fourth author was supported by ERC StG 716424 - CAsE, a Minerva Fellowship of the Max Planck Society and the Studienstiftung des deutschen Volkes.

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 :

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

In fact, two simple rank 3 matroids (or more generally, two paving matroids) have the same multiplicity vector (m_2, \dots, 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 cardinalities¹ of certain classes of non-isomorphic simple rank 3 matroids. A matroid is called *Tutte-unique* or *T-unique* if it is determined up to isomorphism by its Tutte polynomial² (see [dMN05] for a survey on T-unique matroids). The content of the table can be reconstructed using the database [BK19b].

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	?	?
integrally splitting $\chi_M(t)$	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. & int. split. $\chi_M(t)$	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. & int. split. $\chi_M(t)$	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. split. $\chi_M(t)$	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. & supersolvable	1	1	2	3	5	5	8	10	12	14	15	19

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

The total number of simple rank 3 matroids with $n \leq 12$ (unlabeled) atoms³ is taken from [MMIB12b]. This number also coincides with the number of linear geometries minus one with $n \leq 12$ (unlabeled) points⁴ and has been determined earlier in [BB99].

Using our algorithm in HPC-GAP we directly computed all 815107 simple rank 3 matroids with integrally splitting characteristic polynomial with up to $n = 14$ atoms and stored

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

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

³<http://oeis.org/A058731>

⁴<http://oeis.org/A001200>

them in the database [BK19b]. Subsequently, we verified our counting by comparing it against the matroids with integrally splitting characteristic for $n \leq 11$ in [MMIB12a]⁵.

1.1. Applications of the Database. Inspecting the database [BK19b] enables us to investigate questions like:

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

We answer this question negatively in Example 1.1.

(b) What is the smallest number of atoms of a representable rank 3 matroid which is divisionally free but not inductively free?⁶

We answer this question in Example 1.2.

(c) Does the database confirm Terai's freeness conjecture for further classes of arrangements?

Indeed, this is Theorem 1.3.

Some of these questions require the construction of *all* matroids with the corresponding number of atoms first, demonstrating the usefulness of a database.

Example 1.1. 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.

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

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

The matroids M_1 and M_2 are representable over \mathbb{Q} and $\mathbb{Q}(\sqrt{5})$, respectively. They admit the following representation matrices, respectively:

$$R_1 := \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & \frac{1}{2} & 0 & 0 & 0 & 1 & \frac{1}{2} & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & \frac{1}{2} & 1 & \frac{1}{2} & 1 & -1 \end{pmatrix},$$

$$R_2 := \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & \varphi + 1 & 0 & 0 & 0 & 1 & 1 & -\varphi & -\varphi \\ 0 & 0 & 0 & 0 & 1 & 1 & \varphi & -1 & -\varphi + 1 & \varphi + 1 & \varphi \end{pmatrix},$$

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:

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

Both M_1 and M_2 have an integrally splitting characteristic polynomial:

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

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.

⁵We wrote a short program to compute the characteristic polynomial of these matroids as the matroids come without precomputed properties in [MMIB12a].

⁶It is already known 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.

Finally, the corresponding question (a) for the stricter notion of supersolvability is confirmed by the database and already proven for rank 3 matroids in [Abe17, Proposition 4.2]. The proof is formulated for arrangements but works without changes for matroids.

Example 1.2. By inspecting the database we found that 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 free. It can be represented by the following matrix:

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

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 [BK19c].

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 [DIM19].

Inspecting our database we obtain the following result:

Theorem 1.3. *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 simple rank 3 matroids. In Section 3 we discuss the Algorithm used to construct a tree-iterator classifying all non-isomorphic simple rank 3 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 $\text{Spec } \mathbb{Z}$. In Section 5 we finally prove Theorem 1.3. In Appendix A we collect some terminology about rooted trees. In Appendix B we define recursive and tree-iterators and in Appendix C we introduce algorithms to evaluate them in parallel. Appendix D 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 E to demonstrate the significance of our parallelized algorithms in the generation of (certain classes) of simple rank 3 matroids.

ACKNOWLEDGMENTS

We would like to thank Rudi Pendavingh for pointing us to the paper [DW89] and providing us with his SageMath code to compute the Dress-Wenzel condition for representability of matroids established in loc. cit. Using his code we could avoid computing the empty moduli spaces of ca. 400.000 non-representable matroids with 14 atoms. We would also like to thank Markus Lange-Hegermann and Sebastian Posur for their comments on an earlier version of this paper. Our thanks goes also to Sebastian Gutsche and Fabian Zickgraf

who helped us to setup the VM on which the public database is running. Last but not least, we are indebted to the anonymous referees for their careful reading and for the valuable suggestions which helped us improve the exposition.

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 $r(S)$ 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 $r(E)$. A subset $S \subseteq E$ is called **independent** if $|S| = r(S)$ and otherwise **dependent**. A maximal independent set is called a **basis of M** .

Remark 2.2 (Basis Extension Theorem). Any independent subset of a matroid can be extended to a basis. Hence, the cardinality of any basis equals the rank of the matroid.

The flats form a poset⁷ \mathcal{F} by inclusion. Dually, a **coatom** is a maximal element in $\mathcal{F} \setminus \{E\}$. An isomorphism between the matroids (E, \mathcal{F}) and (E', \mathcal{F}') is a bijective map $E \rightarrow E'$ which induces an isomorphism $\mathcal{F} \rightarrow \mathcal{F}'$ of posets.

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

Definition 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 **lattice of flats** $L(\mathcal{A})$ 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 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

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

⁷The poset of flats is a geometric lattice, i.e., a finite atomic semimodular lattice. Conversely, finite atomic semimodular lattices give rise to matroids.

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**

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

The main application of this article is the enumeration of matroids with an integrally splitting characteristic polynomial. 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 for a graphic or cographic matroid M induced by a planar graph G the matroid has an integrally splitting characteristic polynomial if and only if G is chordal as shown in [DK98] for graphic and in [KR11] for cographic matroids, respectively. In both cases, the fact that the characteristic polynomial is integrally splitting even implies that M is supersolvable. However, it is still safe to say that the rather small class of matroids in \mathcal{ISM}_r is not yet well understood when $r \geq 3$.

2.2. Simple Rank 3 Matroids. We will restrict ourselves to the case of simple rank 3 matroids in the following definitions. It is worth pointing out at this point that a rank 3 matroid is simple if and only if it is **paving** which in general means that any circuit is at least as large as the rank of the matroid. 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-empty. In this case the characteristic polynomial is integrally splitting 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 . The deletion just removes the element H from the matroid and for a representable matroid the reduced contraction is the matroid that arises by intersecting all hyperplanes with H :

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

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

The **reduced contraction**⁸ 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 theory of free hyperplane arrangements. The first one generalizes Terao's notion of inductively free hyperplane arrangements to matroids [Ter80]. Intuitively, a matroid M is inductively free if there is a pair of a deletion and reduced contraction with respect to one atom both of which are inductively free and

⁸This definition 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.

the characteristic polynomial of the reduced contraction divides the one of M . As the start of the induction one defines all rank 2 matroids and all Boolean matroids to be inductively free.

Definition 2.8. We define the class \mathcal{IFM}_3 of **inductively free rank 3 matroids** to be the smallest class of simple rank 3 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]. This class is defined in a similar fashion just without the assumption on the deletion of a matroid.

Definition 2.9. The class \mathcal{DFM}_3 of **divisionally free rank 3 matroids** is the smallest class of simple rank 3 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 $\chi_{M^H}(t) | \chi_M(t)$.

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 1.2 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 Table 1, for example).

Remark 2.11. Due to the recursive nature of the definition of inductive freeness, a database containing the simple rank 3 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 simple rank 3 matroids will rely on the language of bipartite graphs. Our description is a special case of m -partitions which describes general paving matroids of rank $m+1$ [Oxl11, Proposition 2.1.24].

Definition 3.1. A (proper) **2-partition** of a finite set E is a set \mathcal{E} of non-empty (proper) subsets of E , called blocks, such that

- each block contains at least 2 elements;
- each pair of elements is contained in exactly one block.

Condition (b) means that $\{\binom{F}{2} \mid F \in \mathcal{E}\}$ is a partition of $\binom{E}{2} := \{\{a, b\} \subseteq E \mid a \neq b\}$.

Remark 3.2. Let \mathcal{E} be a 2-partition of E . Then

- $\bigcup \mathcal{E} = E$.
- $|F \cap F'| \leq 1$ for all $F, F' \in \mathcal{E}$ with $F \neq F'$.
- $\sum_{F \in \mathcal{E}} \binom{|F|}{2} = \binom{|E|}{2}$.
- 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**, if the

2-partition is proper. Connecting, as in Figure 1, the elements of E with an initial element and the blocks with a terminal element we obtain a geometric lattice of flats of a simple rank 3 matroid. Hence, there is a bijective correspondence between simple rank 3 matroids with ground set E and proper 2-partitions of E . Therefore, we will henceforth call the elements of E **atoms** and those of \mathcal{E} **coatoms**.

Since each pair of atoms is contained in exactly one coatom, 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.

We divide the matroid generation by only considering those with a fixed number of coatoms of each size at a time. To this end, we define size vectors of coatoms which satisfy the condition in Remark 3.2 as multiplicity vectors:

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

Each 2-partition \mathcal{E} gives rise to an **associated multiplicity vector** $m(\mathcal{E}) = (m_k)$ with $m_k = |\{F \in \mathcal{E} : |F| = k\}|$.

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 $\chi_{\mathcal{A}_3}(t) = (t-1)(t-2)(t-3)$.

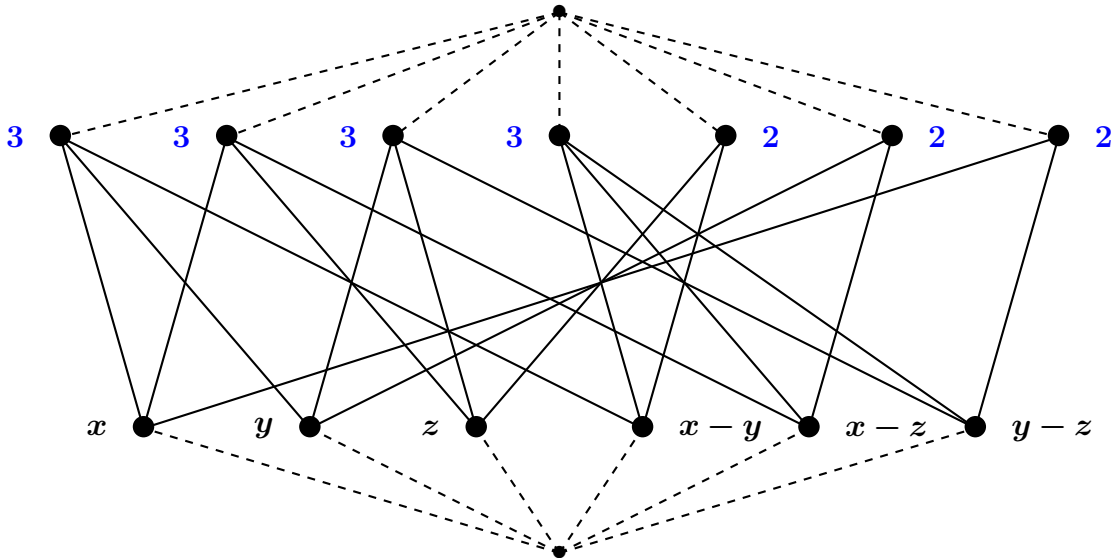


Figure 1. The lattice of flats of the \mathcal{A}_3 braid arrangement with atoms in the bottom row and coatoms in the top row. The linear forms depicted next to the atoms are a possible representations of the matroid. The numbers denoted in blue are the multiplicities of the coatoms.

To generate the multiplicity vectors of all simple rank 3 matroids of fixed size n we can naively iterate over all vectors in $\{0, \dots, n\}^{n-2}$ satisfying the equation in Definition 3.3. Additionally, we can assume that any matroid has at least as many coatoms as atoms by a theorem of de Bruijn and Erdős [dBE48]. Finally, we are only considering those multiplicity vectors (m_k) such that the corresponding characteristic polynomial as in (*) is integrally splitting.

Let T denote the set of isomorphism classes of simple rank 3 matroids on n unlabeled atoms⁹. In what follows we will embed T as the set of leaves in a rooted tree T_\bullet . To make

⁹Cf. (<http://oeis.org/A058731>).

this precise we will use the language of rooted trees and tree-iterators which we summarized in Appendix A and Appendix B, respectively.

We start by describing a rooted tree \tilde{T}_\bullet with an action of the symmetric group $\text{Sym}(n)$, such that the quotient tree $T_\bullet := \tilde{T}_\bullet / \text{Sym}(n)$ (in the sense of Remark A.4) classifies the set T . The set of leaves $\tilde{T} := \lim \tilde{T}_\bullet$ is then the set of isomorphism classes of simple rank 3 matroids on n labeled atoms.¹⁰

We subdivide this problem by describing a subtree $\tilde{T}_\bullet^{(m_k)} \subseteq \tilde{T}_\bullet$ such that $T_\bullet^{(m_k)} := \tilde{T}_\bullet^{(m_k)} / \text{Sym}(n)$ classifies the set $T^{(m_k)}$ of all non-isomorphic simple rank 3 matroids with given multiplicity vector (m_k) . Our goal is to build a locally uniform tree-iterator $t^{(m_k)}$ having the tree $T_\bullet^{(m_k)}$ as its tree of relevant leaves in the language of Remark B.3.

In order to describe the larger tree $T_\bullet^{t^{(m_k)}}$ associated to $t^{(m_k)}$ (in the sense of Remark B.2) we define the theoretically possible pre-stages of 2-partitions with associated multiplicity vector (m_k) in the following sense:

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

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

The rooted tree $T_\bullet^{t^{(m_k)}}$ can now be described as follows: We set $T_0^{t^{(m_k)}} := \{*\}$ and for $1 \leq i \leq n-2$ let $T_i^{t^{(m_k)}}$ consist of all admissible partial 2-partitions of level $k_0 = n-i-1$ modulo the action of $\text{Sym}(n)$. All maps $T_i^{t^{(m_k)}} \leftarrow T_{i+1}^{t^{(m_k)}}$ are evident and surjective. The rooted tree $T^{t^{(m_k)}}$ differs from its subtree $T^{(m_k)}$ by the possible dead ends, i.e., by those admissible partial 2-partitions that cannot be completed to a proper 2-partition.

To describe the iterator $t^{(m_k)}$ 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 $k_1 < k_0$. In this computation we only consider lexicographically minimal extensions with respect to the stabilizer of A under the action of the symmetric group $\text{Sym}(n)$ to avoid iterating over isomorphic matroids multiple times. The details of this procedure are given in Algorithm 1 (**IteratorFromState**).

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

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

For the proof of Theorem 1.3 we are only interested in those matroids with an **integrally splitting characteristic polynomial** (see Section 5). Since the multiplicity vector of a rank 3 matroid M determines the characteristic polynomial χ_M by (*) we only consider tree-iterators $t^{(m_k)}$ such that the corresponding characteristic polynomial defined by (*) is integrally splitting.

¹⁰Cf. (<http://oeis.org/A058720>, for $k = 3$).

Algorithm 1: IteratorFromState

Input: state s consisting of

- a number n of atoms
- a multiplicity vector $(m_k) = (m_2, \dots, m_{n-1})$ 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 $\text{Next}(\text{iter})$ 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  $\text{iter}$  and equip it with
2   • an empty list  $APP$  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 ( $\text{iter}$ )
6   /* find the next block of coatoms of multiplicity  $k_0$ : */
7   if next  $A' = \{A'_1, \dots, A'_{m_{k_0}}\}$  exists with // find  $m_{k_0}$  new coatoms
8     •  $A \cup A'$  admissible partial 2-partition of level  $k_1$ 
9     /* the following line guarantees the generation of pairwise
10      non-isomorphic bipartite graphs, the justification will be
11      provided in Remark 3.5 */
12     • the lexicographically minimal element  $A''$  in the orbit of  $A'$  under
13      $\text{Stab}_{\text{Sym}(n)}(A)$  is not contained in  $APP$ 
14     /* Lines 6,7,8 can again be realized by an iterator which
15      returns the next  $A'$  or fail if no such  $A'$  exists. */
16     then
17       save  $A''$  in  $APP$ 
18        $A'' := A \cup A''$  // augment the current partial 2-partition
19       if  $k_1 \geq 2$  then
20          $s' := (n, (m_k), k_1, A'')$  // define the new state
21         /* return IteratorFromState applied to the new state  $s'$  */
22         return IteratorFromState( $s'$ )
23       else
24         return  $A''$  // return the complete adjacency list
25     else
26       return fail
27 return  $\text{iter}$ 

```

We have implemented Algorithm 1 as part of the GAP-package MatroidGeneration [BK20]. We show in Appendix C how to evaluate recursive iterators in parallel. We applied Algorithm 1 to all multiplicity vectors with an integrally splitting characteristic polynomial and stored the resulting matroids in the database [BK19b] using the GAP-package ArangoDBInterface [BK19a].

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

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 $A \cup A'$ under $\text{Stab}_{\text{Sym}(n)}(A)$ is nothing but $A \cup A''$, namely the union of A and the lexicographically minimal element A'' of the orbit of A' under $\text{Stab}_{\text{Sym}(n)}(A)$ (considered in line 8). This is due to the fact that sets in A' are of different cardinality than those in A .

Remark 3.6. A simple rank 3 matroid M (of size n) is *weakly atom balanced on dependent coatoms*, i.e., every atom is contained in at most $\frac{n-1}{2}$ coatoms 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, \dots, F_ℓ be the coatoms of size at least 3 containing the atom k . This implies $|F_i \setminus \{k\}| \geq 2$ for all $1 \leq i \leq \ell$. Furthermore, by definition of a simple matroid it holds that $F_i \cap F_j = \{k\}$ for all pairs $1 \leq i < j \leq \ell$. This means that the coatoms F_1, \dots, 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 $\ell \leq \frac{n-1}{2}$ 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. Naïvely, one needs to loop over all subsets of $\{1, \dots, n\}$ of a given size and discard all those which contain a pair of atoms which is already contained in a previous coatom. To speed up this part of the computation we use the following methods:

- Following Remark 3.6 we can discard atoms in computations of multiplicity $k_0 \geq 3$ if they are already contained in $\frac{n-1}{2}$ coatoms.
- We can assume that within one level the coatoms are ordered lexicographically. Thus, we discard all subsets of $\{1, \dots, n\}$ that are lexicographically smaller than the last coatom of the same size before calling the search algorithm.
- If at any step a transposition $(e_1 e_2)$ is in the stabilizer of all previous coatoms we treat the atoms e_1 and e_2 as equivalent at this step of the algorithm. This means that we loop over subsets of $\{1, \dots, n\}$ modulo all such equivalences instead of the entire set $\{1, \dots, n\}$. To ensure we do not miss relevant subsets we also need to consider subsets of these representatives of smaller sizes and fill up the resulting subsets to the correct size with atoms that are not contained in any coatom yet. We can choose the first unused atoms in the lexicographic order for this matter.

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 [JPW19].
- `Images` provides algorithms which, given a permutation group G on a set Ω 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. [JJPW19].

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

Remark 3.9. In the database [BK19b] we store lexicographically minimal elements of the list of coatoms computed by the `GAP` package `Images`. This specific form enables the lookup of an arbitrary matroid in the database by computing its uniquely defined minimal image under the action of the symmetric group. We have also used this lookup procedure to compute the inductive freeness property since this property depends by definition on the inductive freeness of matroids of smaller size (cf. Remark 2.11).

Remark 3.10. The computations to generate all simple rank 3 matroids with integrally splitting characteristic polynomial 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 coatom balanced (cf. Definition 5.1). In Proposition 3.11, we prove that there are no matroids with one of the above multiplicities vectors. Hence, these computations which did not terminate do not impose any restrictions on Theorem 1.3 or Table 1.

Proposition 3.11. *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 v_1 or v_2 as its associated multiplicity vector.*

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 investigate the admissible partial 2-partitions that contain all coatoms of size greater than 3. 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.

Consider a step in which we add the coatom $C := \{e_1, e_2, e_3\}$ to a list of coatoms A and obtain a new list A' . Then we have $d_{A'}(e_i) = d_A(e_i) - 2$ for $1 \leq i \leq 3$ and the remaining deficiencies remain constant. In particular, the parity of all deficiencies is constant in this step.

In the case of the multiplicity vector v_1 we can without loss of generality assume that all admissible partial 2-partitions with all coatoms of size greater than 3 contain the coatom $[1, \dots, 5]$ and the atom 4 is not contained in any coatom of size 4. Thus, we have $d_A(4) = 13 - 4 = 9$ for all such lists A . Since this number is odd but the deficiency of any matroid is 0 the above discussion proves that all such partial list of coatoms of v_1 can not be completed to a matroid.

To prove the remaining statement regarding the multiplicity vector v_2 , we start our parallel matroid generation algorithm but terminate after completing all levels of size greater than 3. We obtain the two partial admissible 2-partitions

$$A_1 := [[1, 2, 3, 4, 5, 6], [1, 7, 8, 9]], \quad A_2 := [[1, 2, 3, 4, 5, 6], [7, 8, 9, 10]].$$

Now, we need to add coatoms of size 3 and exactly one coatom of size 2 to the admissible partial 2-partitions A_1, A_2 . An analogous argument as in the first case shows that number of atoms with odd deficiency of the lists A_1, A_2 must be exactly two. Computing deficiencies of the lists A_1 and A_2 yields that 1 is only atom with an odd deficiency in A_1 whereas all atoms in A_2 have an even deficiency. Thus, there exists no matroid with multiplicity vector v_2 . \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 variety*, namely an affine subvariety $V(I') = \mathbb{A}_{\mathbb{Z}}^{rn+1}$, where r is the rank of M and n its number of atoms.

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

of M . Then

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

It follows that 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 $\text{Spec } R[d]/I'$ of the matroid M contains only one point. In this case, the matroid M has 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 $V(I) \setminus V(J) \subseteq \mathbb{A}_{\mathbb{Z}}^n = \text{Spec } R$, 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

$$I = \sum \{J_D \mid D \subseteq E \text{ dependent}, |D| = r\},$$

$$J = \prod \{J_B \mid B \in \mathcal{B}(M)\}.$$

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 $S \subseteq E$ basis. The ideal I can be replaced by the saturation

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

Then M is representable iff $1 \notin \tilde{I}$. For the Gröbner basis computations over \mathbb{Z} we used SINGULAR [DGPS19] from within the homalg project [hom20].

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 assume $B = \{1, \dots, r\}$. 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 nonzero entry in every column and the first nonzero 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 1.3

If a matroid has an atom which is contained in many coatoms or conversely a coatom which contains many atoms any realization satisfies Terao's conjecture. This statement will be a crucial ingredient in the proof of Theorem 1.3. To formalize it we make 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 each atom is contained in at most a -many coatoms.
- We call M **coatom balanced** if each coatom contains strictly less than a -many atoms.
- If M is both atom and coatom 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 coatoms 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 coatom 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. \square

Now we have all ingredients to prove Theorem 1.3.

Proof of Theorem 1.3. 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 is not integrally splitting 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 [Ter80].
- If a matroid has a unique representation over the integers¹² it trivially satisfies Terao's conjecture.

Querying the database [BK19b] there are

- 783280 rank 3 matroids of size 14 with integrally splitting characteristic polynomial,
- 1574 thereof are representable over the integers,
- 174 thereof are not inductively free,
- 64 thereof are strongly balanced.

All of these 64 matroids are only representable over \mathbb{F}_5 with a unique representation, and are hence irrelevant for the Terao conjecture. This completes the proof of Terao's conjecture for rank 3 arrangements with 14 hyperplanes in any characteristic. \square

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 $\text{Spec } \mathbb{Z}$ and the free locus therein in a subsequent article which will establish Terao's conjecture for rank 3 arrangements with up to 14 hyperplanes in any characteristic.

¹²i.e., the moduli space $\text{Spec } R/\tilde{I} \rightarrow \text{Spec } \mathbb{Z}$ of representations is $\text{Spec } \mathbb{F}_p \rightarrow \text{Spec } \mathbb{Z}$, a singleton.

APPENDIX A. ROOTED TREES

In this Appendix we discuss rooted trees and give simple examples for their use as a tool to iterate over desired sets. Instead of the classical definition of rooted trees we use an alternative mathematical model of rooted trees in which one can easily interpret the data structure of tree-iterators and their evaluations which we introduce in Appendix B. Expressed in this model, the (parallelized) evaluation of tree-iterators (Algorithm 2) can then be understood as a limiting process.

For the design of our algorithms we represent a finite **rooted forest** (or set of **rooted trees**) as a finite 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**. We denote the set of **leaves** of T_\bullet by $T := \lim T_\bullet$, which is the set of non-images in T_\bullet .¹³ As mentioned in the introduction we then say that T_\bullet **classifies** T .

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.

Convention. So without loss of generality 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: $\varphi_i^{-1}(v) \cap T \neq \emptyset \implies \varphi_i^{-1}(v) \subseteq T$.

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:

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

Define $T_{i-1} \xleftarrow{\varphi^i} T_i$ to be the map removing the left most¹⁴ 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 number¹⁵ $C_n = \binom{2n}{n} - \binom{2n}{n+1} = \frac{1}{n+1} \binom{2n}{n}$.

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 $M_n = \langle a_0, \dots, a_n \rangle$ of rank $n + 1$:

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

¹³The notation $\lim T_\bullet$ can be justified as follows: T_\bullet is a sequential inverse system in the category finite sets with the set of leaves as its limiting object.

¹⁴or right most, ...

¹⁵Cf. (<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
$T^{(n)}$	$\{\{1\}\}$	$\{\{\{1\}, \{2\}\}\}$	$\{\{\{1\}, \{2\}, \{3\}\}; \{\{1\}, \{\{2\}, \{3\}\}\}; \{\{2\}, \{\{1\}, \{3\}\}\}; \{\{3\}, \{\{1\}, \{2\}\}\}\}$

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

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 φ_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 $\text{Sym}(n)$ on $\{1, \dots, n\}$ turns the rooted tree T_\bullet in Example A.3 into a rooted $\text{Sym}(n)$ -tree. The rooted tree of orbits $T_\bullet/\text{Sym}(n)$ then classifies $T/\text{Sym}(n)$ which is the set of phylogenetic trees with unlabeled leaves.¹⁷

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

¹⁶Cf. (<http://oeis.org/A000311>).

¹⁷Cf. (<http://oeis.org/A000669>).

APPENDIX B. RECURSIVE ITERATORS AND TREE-ITERATORS

In this appendix we introduce the data structure of so-called tree-iterators, which we use to recursively iterate over the vertices of a rooted tree. This data structure is a central ingredient of all algorithms.

Definition B.1. Let T be a set.

- A **recursive iterator** t **within** T is an iterator which upon popping produces either $\text{Next}(t) = \text{fail} \notin T$ or a **child** $\text{Next}(t)$ which is either
 - (a) a new recursive iterator within T , or
 - (b) an element of T .

If the pop result $\text{Next}(t)$ is fail then any subsequent pop result of t remains fail . We call T the **ambient set** of t .

- A **full evaluation** of a recursive iterator recursively pops all recursive iterators until each of them pops fail .
- If t is a recursive iterator then the subset of elements $T(t) \subseteq T$ produced upon full evaluation is called the **set of leaves of t in T** . We say that t **classifies** $T(t) \subseteq T$.
- A recursive iterator is called **locally uniform** if every descendant either pops recursive iterators or leaves, exclusively (if not fail).
- A recursive iterator t within T is called a **tree-iterator** if upon full evaluation each element of $T(t) \subseteq T$ is the pop result of exactly one descendant of t .

In order to iterate over a tree T_\bullet with set of leaves $T = \lim T_\bullet$ it is somewhere between convenient and almost unavoidable to construct a tree-iterator t within T which might iterate over a larger tree T_\bullet^t having a set of leaves $T^t = \lim T_\bullet^t$ which is larger than $T(t)$, i.e., with dead ends being all tree-iterators which are descendants of t but have no own descendants. In our application to the classification of simple rank 3 matroids the dead ends are the admissible partial 2-partitions which cannot be completed to a proper 2-partition (cf. Definition 3.4).

Remark B.2. A tree-iterator t within T or any of its descendants can be understood as a vertex of the rooted tree

$$T_\bullet^t : T_0^t := \{t\} \xleftarrow{\varphi_1^t} T_1^t \xleftarrow{\varphi_2^t} T_2^t \xleftarrow{\varphi_3^t} \dots \xleftarrow{\varphi_d^t} T_d^t,$$

inductively described as follows: Let t' be any descendant of t interpreted as an element $t' \in T_i^t$. If $\text{Next}(t') = \text{fail}$ then t' has no (further) preimages under φ_{i+1}^t . Otherwise each evaluation $\text{Next}(t') \in T_{i+1}^t$, which is a preimage of t' under φ_{i+1}^t . We call T_\bullet^t the **tree associated to t** . Its set of leaves $T^t := \lim T_\bullet^t$ is the union of $T(t) \subseteq T$ and the set of all tree-iterators which are descendants of t but have no own descendants.

Remark B.3. Given a tree-iterator t within T with corresponding tree T_\bullet^t as in Remark B.2 we define the subtree

$$T_\bullet : T_0 \xleftarrow{\varphi^1} T_1 \xleftarrow{\varphi^2} T_2 \xleftarrow{\varphi^3} \dots \xleftarrow{\varphi^d} T_d.$$

with $\lim T_\bullet = T(t) \subseteq T$, i.e., the subtree $T_\bullet \subseteq T_\bullet^t$ consisting of the leaves in $T(t)$ and all their predecessors. We call T_\bullet the **tree of relevant leaves of t** .

In order to iterate over a tree T_\bullet with set of leaves $T = \lim T_\bullet$ we use the freedom to construct a tree-iterator t within T having T_\bullet as its tree of relevant leaves, even though its associated tree T_\bullet^t might be considerably larger.

APPENDIX C. PARALLEL EVALUATION OF RECURSIVE ITERATORS

In this Appendix we describe the three algorithms

- **ParallelyEvaluateRecursiveIterator** (Algorithm 2),

- **EvaluateRecursiveIterator** (Algorithm 3),
- **LeafIterator** (Algorithm 4),

which constitute our general parallelization scheme for recursive iterators. They are independent of any specific recursive iterator (e.g., the one defined by **IteratorFromState** in Algorithm 1). Furthermore, the recursive iterators can be implemented in classical sequential code, i.e., this organization requires no pre-knowledge in parallel programming in order to implement a recursive iterator and evaluate it in parallel. We have implemented the three algorithms in the High-Performance-Computing (HPC) version¹⁸ of GAP 4.9.2 [GAP18] as part of the GAP-package `ParallelizedIterators` [BBK19].

The combination of these three algorithms takes a recursive iterator t (within T) as input and returns an iterator $\ell(t)$ which iterates over the set of leaves $T(t) \subseteq T$. We call $\ell(t)$ the **leaf-iterator** associated to t . If t is a tree-iterator then $\ell(t)$ produces no duplicates.

We now briefly explain the role of each of the three algorithms and the way they interact: Algorithm 4 is executed in the main thread with a recursive iterator as input. In the main application of this paper the input is the tree-iterator $t^{(m_k)}$ of all rank 3 matroids of a given multiplicity vector (m_k) , 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 many workers (threads) as specified by the user, triggers Algorithm 3 in each of them, and then terminates.

The shared¹⁹ priority queue stores the list of recursive iterators still to be searched along with their priority, which in our case is the depth at which they were created. The instance of Algorithm 3 running in each thread asks for the highest priority iterator t' in the priority queue P and evaluates $t'' := \text{Next}(t')$. If t'' is an element of T then t'' is added to the FIFO L of leaves and t' is returned to P with the same priority. If t'' is again an iterator then t' and t'' are returned to P ; t' is returned with the same priority and t'' with the priority of t' increased by one. Finally if $t'' = \text{fail}$ then nothing is done. After any of the three actions the instance of Algorithm 3 starts over again. In particular, our use of a priority queue avoids the need for a central process supervising the workers.

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  $w_1, \dots, w_n$ 
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  $i = 1, \dots, n$  do
7 |   | EvaluateRecursiveIterator( $n, L, P, s, j$ ) within worker  $w_i$ 
8 | SignalSemaphore( $s$ )
9 | return none

```

¹⁸Since version 4.9.1 GAP can be compiled with the option `--enable-hpcgap`.

¹⁹Implementations of priority queues exist both for shared memory and distributed operation. We have chosen

Algorithm 2 gets as input a recursive iterator, a number n of workers, and a global FIFO L . It initializes a shared priority queue 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 .

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 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
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 */

```

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

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.

with nonnegative integers as admissible values, which we use to tell workers when to start looking for jobs to process. The command `SignalSemaphore(s)` increases s by 1. The command `WaitSemaphore(s)` halts until $s > 0$ and then decreases s by 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 recursive iterator t within T into a single iterator $\ell(t)$ which enumerates $T(t) \subseteq T$.

Algorithm 4: LeafIterator (Leaf-iterator of a recursive iterator)

Input:

- A recursive 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 IsDone( $\ell$ ) to check if first entry of  $L$  is faila
5 |   Define Next( $\ell$ ) to return the first entry of  $L$  which is an element of  $T(t)$ 
6 | return  $\ell$ 

```

^aRecall, `fail` $\notin T$.

APPENDIX D. WHY HPC-GAP?

We list some advantages of our implementation in 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 hangs²⁰ 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.

²⁰either manually terminated or due to an instability of HPC-GAP, which rarely happens in the current version

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 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 redefining regions, etc.

APPENDIX E. 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 E.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 [7](#)
- [Abe17] Takuro Abe, *Restrictions of free arrangements and the division theorem*, Perspectives in Lie theory, Springer INdAM Ser., vol. 19, Springer, Cham, 2017, pp. 389–401. MR 3751135 [4](#)
- [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](#)
- [BBK19] Mohamed Barakat, Reimer Behrends, and Lukas Kühne, *ParallelizedIterators – Parallely evaluate recursive iterators*, 2017–2019, (<https://github.com/homalg-project/ParallelizedIterators>). [18](#)
- [BK19a] Mohamed Barakat and Lukas Kühne, *ArangoDBInterface – An interface to ArangoDB*, 2018–2019, (<https://github.com/homalg-project/ArangoDBInterface>). [10](#)
- [BK19b] Mohamed Barakat and Lukas Kühne, *matroids_split_public – a database collection for rank 3 integrally split simple matroids*, 2019, (<https://github.com/homalg-project/MatroidGeneration/blob/master/README.md>). [2](#), [3](#), [10](#), [11](#), [14](#)

- [BK19c] Mohamed Barakat and Lukas Kühne, *Smallest divisionally free not inductively free arrangement*, 2019, (<https://github.com/homalg-project/CapHomalgNotebooks/blob/master/DivFreeNotIndFree.ipynb>). 4
- [BK20] Mohamed Barakat and Lukas Kühne, *MatroidGeneration – Generate low-rank matroids*, 2017–2020, (<https://github.com/homalg-project/MatroidGeneration>). 10
- [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
- [dBE48] N. G. de Bruijn and P. Erdős, *On a combinatorial problem*, Nederl. Akad. Wetensch., Proc. **51** (1948), 1277–1279 = Indagationes Math. 10, 421–423 (1948). MR 28289 8
- [DGPS19] Wolfram Decker, Gert-Martin Greuel, Gerhard Pfister, and Hans Schönemann, *SINGULAR 4-1-2 — A computer algebra system for polynomial computations*, <http://www.singular.uni-kl.de>, 2019. 13
- [DIM19] Alexandru Dimca, Denis Ibadula, and Anca Măcinic, *Freeness for 13 lines arrangements is combinatorial*, Discrete Mathematics **342** (2019), no. 8, 2445 – 2453. 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>). 18
- [hom20] homalg project authors, *The homalg project – Algorithmic Homological Algebra*, (<http://homalg-project.github.io>), 2003–2020. 13
- [HR15] Torsten Hoge and Gerhard Röhrle, *On inductively free reflection arrangements*, J. Reine Angew. Math. **701** (2015), 205–220. MR 3331732 3
- [JJPW19] Christopher Jefferson, Eliza Jonauskite, Markus Pfeiffer, and Rebecca Waldecker, *Minimal and canonical images*, J. Algebra **521** (2019), 481–506. MR 3906181 11
- [JPW19] Christopher Jefferson, Markus Pfeiffer, and Rebecca Waldecker, *New refiners for permutation group search*, J. Symbolic Comput. **92** (2019), 70–92. MR 3907348 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>). 3
- [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
- [OT92] Peter Orlik and Hiroaki Terao, *Arrangements of hyperplanes*, Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], vol. 300, Springer-Verlag, Berlin, 1992. MR MR1217488 (94e:52014)
- [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

- [Ter80] Hiroaki Terao, *Arrangements of hyperplanes and their freeness. I*, J. Fac. Sci. Univ. Tokyo Sect. IA Math. **27** (1980), no. 2, 293–312. MR 586451 [6](#), [14](#)
- [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
Email address: mohamed.barakat@uni-siegen.de

DEPARTMENT OF MATHEMATICS, UNIVERSITY OF KAISERSLAUTERN, 67653 KAISERSLAUTERN, GERMANY
Email address: behrends@gmail.com

SCHOOL OF COMPUTER SCIENCE, UNIVERSITY OF ST ANDREWS, KY16 9SX ST ANDREWS, UNITED KINGDOM
Email address: caj21@st-andrews.ac.uk

EINSTEIN INSTITUTE OF MATHEMATICS, THE HEBREW UNIVERSITY OF JERUSALEM, GIV'AT RAM, JERUSALEM, 91904, ISRAEL

MAX PLANCK INSTITUTE FOR MATHEMATICS IN THE SCIENCES, INSELSTR. 22, 04103, LEIPZIG, GERMANY
Email address: lukas.kuhne@mis.mpg.de

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