Discrete & Computational Geometry (2023) 70:1356–1377
https://doi.org/10.1007/s00454-023-00557-2

Computing Characteristic Polynomials of Hyperplane Arrangements with Symmetries

Taylor Brysiewicz[1] · Holger Eble[2] · Lukas Kühne[3]

Received: 25 June 2021 / Revised: 23 February 2023 / Accepted: 25 March 2023 /
Published online: 7 November 2023

Abstract
We introduce a new algorithm computing the characteristic polynomials of hyperplane arrangements which exploits their underlying symmetry groups. Our algorithm counts the chambers of an arrangement as a byproduct of computing its characteristic polynomial. We showcase our `julia` implementation, based on `OSCAR`, on examples coming from hyperplane arrangements with applications to physics and computer science.

Keywords Hyperplane arrangement · Chambers · Algorithm · Symmetry · Resonance arrangement · Separability

Mathematics Subject Classification 52C35 · 52B15

1 Introduction

The problem of enumerating chambers of hyperplane arrangements is a challenge in computational discrete geometry [20, 29, 34, 42]. A well-known approach to this problem is through the computation of characteristic polynomials [1, 21, 30, 38, 43, 47].

Editor in Charge: János Pach

Taylor Brysiewicz
tbrysiew@uwo.ca

Holger Eble
eble@math.tu-berlin.de

Lukas Kühne
lkuehne@math.uni-bielefeld.de

[1] Department of Mathematics, Western University, London, Canada

[2] Chair of Discrete Mathematics/Geometry, Technische Universität Berlin, Berlin, Germany

[3] Fakultät für Mathematik, Universität Bielefeld, Bielefeld, Germany

Table 1 Our timings on examples from Sect. 6

$\mathcal{A}$	$\lvert\mathrm{Aut}(\mathcal{A})\rvert$	$d = 3$ (s)	4 (s)	5 (s)	6	7 (s)	8	9
$\mathcal{T}_d$	$(d+1)!2^d$	0.005	0.013	0.041	0.28 s	33.17	8.16 h	
$\mathcal{R}_d$	$(d+1)!$	0.004	0.011	0.035	0.12 s	2.89	19.8 min	$\sim 10\,\mathrm{day}^+$
$\mathcal{C}_{2d}$	$(2d)!2^{2d}$	0.015	0.039	0.085	0.183 s	0.42	1.158 s	4.50 s
$\mathcal{P}_d$	$d!$	0.003	0.013	6.398	$\sim 8\,\mathrm{day}^+$			
$\mathcal{D}_d$	$(d)!2^{d-1}$	0.002	0.005	0.018	0.049 s	0.54	1.9 min	$\sim 8\,\mathrm{day}^+$
$\mathrm{Disc}_{4,n}$	$n!$	–	0.0003	0.0047	0.055	0.71 s	7.62	41.14 s

Computations ran on a single thread (`Intel Core i7-8700`) except for $\mathcal{R}_9$ which ran on 42 threads (`Intel Xeon E7-8867`)

We develop a novel for computing characteristic polynomials which takes advantage of the combinatorial symmetries of an arrangement. While most arrangements admit few combinatorial symmetries [39], most arrangements of interest do [19, 40, 48].

We implemented our algorithm in `julia` [3] and published it as the package `CountingChambers.jl`.[1] Our implementation relies heavily on the cornerstones of the new computer algebra system `OSCAR` [51] for group theory computations (`GAP` [50]) and the ability to work over number fields (`Hecke` and `Nemo` [17]).

While other algorithms and pieces of software exist for studying hyperplane arrangements (see, for instance, [10, 15, 29, 33, 44]), either their chamber-enumeration computations appear as byproducts of more difficult calculations, the code does not use symmetry, or it only pertains to very specific types of arrangements. For example, reference [29] computes the associated zonotope, whose vertices are in bijection with the chambers of the arrangement, containing much more information than the characteristic polynomial. A similar approach is suggested in [15] involving a search algorithm relying upon linear programming. An example of an approach which computes the number of chambers via a much more difficult computation is via the computation of the so-called broken circuit complex of the arrangement (see [7]): a simplicial complex which has the same number of faces as the number of chambers of the arrangement.

As demonstrated in Table 1, the fastest algorithms we know of for counting chambers in hyperplane arrangements (including the one discussed in this article) also compute the characteristic polynomial of the arrangement as a byproduct. In this sense, we do not know of an algorithm which illustrates that counting the chambers of an arrangement is an easier computation than computing its characteristic polynomial. To the best of our knowledge, our implementation is the first publicly available software for counting chambers which uses symmetry.

We showcase our algorithm and its implementation on a number of well-known examples, such as the resonance and discriminantal arrangements. Additionally, we study sequences of hyperplane arrangements which come from the problem of linearly separating vertices of regular polytopes. In particular, we investigate one corresponding to the hypercube $[0, 1]^d$ whose chambers are in bijection with linearly separable Boolean functions.

[1] Available at https://mathrepo.mis.mpg.de/CountingChambers.

In the presence of symmetry, our implementation outperforms the existing software by several orders of magnitude (cf. Table 1). Moreover, its output is guaranteed to be correct since we compute symbolically over the integers or exact number fields and avoid overflow errors thanks to the package `SaferIntegers.jl` [41].

The ninth resonance arrangement (511 hyperplanes in $\mathbb{R}^9$) approaches the limit of what is possible with our implementation: the computation of its characteristic polynomial took ten days on 42 processors. Our computation confirms that its chamber-count is 1955230985997140 as independently and concurrently computed by Chroman and Singhar with different methods [10].

We first give background on hyperplane arrangements in Sect. 2. The ideas outlined in Sect. 3, regarding deletion and restriction algorithms, form the basic structure of our algorithm. We explain the relevant results regarding symmetries of arrangements in Sect. 4. The algorithm and its implementation details reside in Sect. 5. In Sect. 6 we construct and discuss examples of arrangements exhibiting large symmetry groups. We conclude in Sect. 7 with timings and comparisons to other software.

2 Hyperplane Arrangements

We begin by discussing background on the theory of hyperplane arrangements related to the problem of enumerating chambers: the main goal of this article and the associated software. Our notation will mostly follow the textbook by Orlik and Terao [38].

For any field $\mathbb{K}$, a *hyperplane* in $\mathbb{K}^d$ is an affine linear space of codimension one. Throughout this article, we denote by $\mathcal{A} = \{H_1, \ldots, H_n\}$ a (*hyperplane*) *arrangement* where H_i is a hyperplane in $\mathbb{K}^d$.

Definition 2.1 Suppose $\mathcal{A}$ is an arrangement in $\mathbb{R}^d$. The connected components of the complement $\mathbb{R}^d \setminus \bigcup_{H\in\mathcal{A}} H$ are called *chambers* of $\mathcal{A}$ and the set of chambers of $\mathcal{A}$ is denoted by $\mathrm{ch}(\mathcal{A})$.

Example 2.2 We use the arrangement

$$\big\{\underbrace{\{y - x = 1\}}_{H_1}, \underbrace{\{x = 0\}}_{H_2}, \underbrace{\{x + y = 1\}}_{H_3}, \underbrace{\{y = 0\}}_{H_4}\big\}$$

in $\mathbb{R}^2$ as a running example. This arrangement is depicted in Fig. 1. It has ten chambers: two bounded and eight unbounded.

Given a subset $I \subseteq [n] := \{1, \ldots, n\}$, we write the set $\{H_i\}_{i\in I}$ as H_I and its intersection as $L_I = \bigcap_{i\in I} H_i$. The collection of these intersections form the set $L(\mathcal{A}) = \{L_I \mid I \subseteq [n], L_I \neq \emptyset\}$, a combinatorial shadow of $\mathcal{A}$ known as its *intersection poset*. This poset is ordered by reverse inclusion and graded by the *rank function*, $r\colon L(\mathcal{A}) \to \mathbb{Z}_{\geq 0}$, where $r(L_I) = \mathrm{codim}(L_I)$. As a notational convention, we set $r(I) = r(L_I)$ for $I \subseteq [n]$ whenever $L_I \neq \emptyset$.

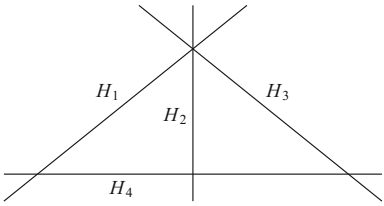


Fig. 1 The arrangement introduced in Example 2.2

2.1 The Characteristic Polynomial

Our algorithm counts chambers of an arrangement by computing a more refined count, namely the characteristic polynomial. The coefficients of this polynomial are known as the unsigned Whitney numbers of the first kind of the intersection poset $L(\mathcal{A})$, which we simply refer to as the *Whitney numbers* of the arrangement.

Definition 2.3 The *characteristic polynomial* of an arrangement $\mathcal{A}$ in $\mathbb{K}^d$ is the polynomial

$$\chi_{\mathcal{A}}(t) = \sum_{\substack{I \subseteq [n] \\ L_I \neq \emptyset}} (-1)^{|I|} t^{d-r(I)} = \sum_{i=0}^{d} (-1)^i b_i(\mathcal{A}) t^{d-i}. \tag{1}$$

The integers $b_i(\mathcal{A})$, defined via (1), are non-negative and are called the *Whitney numbers* of $\mathcal{A}$. We denote the vector of Whitney numbers by $b(\mathcal{A})$.

The characteristic polynomial and Whitney numbers of an arrangement $\mathcal{A}$ depend only on the intersection poset $L(\mathcal{A})$ and have various interpretations depending on the field $\mathbb{K}$ as detailed below.

Real: For an arrangement $\mathcal{A}$ in $\mathbb{R}^d$, Zaslavsky [47] proved that

$$|\mathrm{ch}(\mathcal{A})| = (-1)^d \chi_{\mathcal{A}}(-1) = \sum_{i=0}^{d} b_i(\mathcal{A}).$$

Thus, the Whitney numbers are a refined count of the chambers of $\mathcal{A}$. They have the following geometric interpretation. Given a generic flag $\mathcal{F}_\bullet\colon F_0 \subset F_1 \subset \ldots \subset F_d = \mathbb{R}^d$ of affine linear subspaces F_i [where $\dim(F_i) = i$] the number of chambers of $\mathcal{A}$ which meet F_i but do not meet F_{i-1} is equal to $b_i(\mathcal{A})$ [45, Proposition 2.3.2].
Complex: If $\mathcal{A}$ is an arrangement in $\mathbb{C}^d$ where all hyperplanes contain the origin, then $b_i(\mathcal{A})$ is the ith *topological* Betti number of the complement $\mathbb{C}^d \setminus \bigcup_{H \in \mathcal{A}} H$ with rational coefficients [37]. Because of this, some papers refer to the Whitney numbers $b_i(\mathcal{A})$ as the Betti numbers of the arrangement $\mathcal{A}$ [46].
Finite: When $\mathcal{A}$ is an arrangement over a finite field $\mathbb{F}_q$, Crapo and Rota proved that $\chi_{\mathcal{A}}(q) = \left|\mathbb{F}_q^d \setminus \bigcup_{H \in \mathcal{A}} H\right|$ [12]. Moreover, if $\mathcal{A}$ is a hyperplane arrangement in

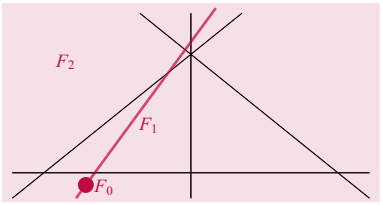


Fig. 2 The intersections of a generic flag (purple) in $\mathbb{R}^2$ with the chambers of $\mathcal{A}$. The point F_0 intersects one chamber, F_1 intersects four others, and F_2 intersects the remaining five, and so $b(\mathcal{A}) = (1, 4, 5)$

$\mathbb{Q}^d$ one may consider its reduction modulo q: $\mathcal{A} \otimes \mathbb{F}_q = \{H_1 \otimes \mathbb{F}_q, \ldots, H_n \otimes \mathbb{F}_q\}$. When q is sufficiently large, we have that $L(\mathcal{A}) = L(\mathcal{A} \otimes \mathbb{F}_q)$ and thus computing $\chi_{\mathcal{A}}(t)$ for rational arrangements also yields the number of points in the complement after reducing modulo large primes.

Example 2.4 Let $\mathcal{A}$ be the arrangement introduced in Example 2.2. Its characteristic polynomial is $\chi_{\mathcal{A}}(t) = t^2 - 4t + 5$. Figure 2 shows a generic flag $\mathcal{F}_\bullet$ intersecting this arrangement verifying that $b(\mathcal{A}) = (1, 4, 5)$.

3 A Deletion–Restriction Algorithm

To compute the Whitney numbers of an arrangement $\mathcal{A}$ in $\mathbb{K}^d$, we take advantage of the behavior of $\chi_{\mathcal{A}}(t)$ under the operations of deletion and restriction. These operations reduce computations about $\mathcal{A}$ to computations about two smaller arrangements. Thus at its core, our main algorithm is a divide-and-conquer algorithm.

Given a hyperplane $H \in \mathcal{A}$, the *deletion* of H in $\mathcal{A}$ is the arrangement $\mathcal{A} \setminus \{H\}$. The *restriction* of H in $\mathcal{A}$ is the arrangement in $H \cong \mathbb{K}^{d-1}$ defined by $\mathcal{A}^H = \{K \cap H \mid K \in \mathcal{A} \setminus \{H\}\}$. The following lemma provides the basic foundation of our algorithm.

Lemma 3.1 [38, Cor. 2.57] *Given a hyperplane $H \in \mathcal{A}$, we have that $\chi_{\mathcal{A}}(t) = \chi_{\mathcal{A}\setminus\{H\}}(t) - \chi_{\mathcal{A}^H}(t)$. In particular, $b(\mathcal{A}) = b(\mathcal{A} \setminus \{H\}) + 0|b(\mathcal{A}^H)$ where $0|b$ means prepending the vector b with a zero.*

3.1 A Simple Deletion–Restriction Algorithm

Lemma 3.1 along with the fact that the empty arrangement in $\mathbb{K}^d$ has the vector of Whitney numbers $(1, 0, \ldots, 0) \in \mathbb{N}^{d+1}$ suggests the following well-known recursive algorithm for computing $b(\mathcal{A})$.

Algorithm 1: Whitney numbers via simple deletion and restriction

Input: A hyperplane arrangement $\mathcal{A}$ in $\mathbb{K}^d$
Output: The vector of Whitney numbers $b(\mathcal{A})$
WhitneyNumbers $(\mathcal{A})$
1 **if** $\emptyset \neq \mathcal{A}$ **then**
2 choose $H \in \mathcal{A}$
3 **return WhitneyNumbers**$(\mathcal{A} \setminus \{H\})$ + 0|**WhitneyNumbers**$(\mathcal{A}^H)$
4 **else**
5 **return** $(1, 0, \ldots, 0)$

Structurally, Algorithm 1 is a depth-first binary tree algorithm on arrangements, rooted at the initial input: one child represents a deletion and the other a restriction, as shown in Fig. 3. The implementation of Algorithm 1 is already nontrivial as it is often the case that some hyperplanes become the same after a restriction. Thus, its proper implementation requires care in representing an arrangement on a computer.

3.2 Computationally Representing Deletions and Restrictions

An arrangement $\mathcal{B}$ coming from $\mathcal{A}$ via deletions and restrictions may be represented by an encoding of the restricted hyperplanes. To be precise, the pair

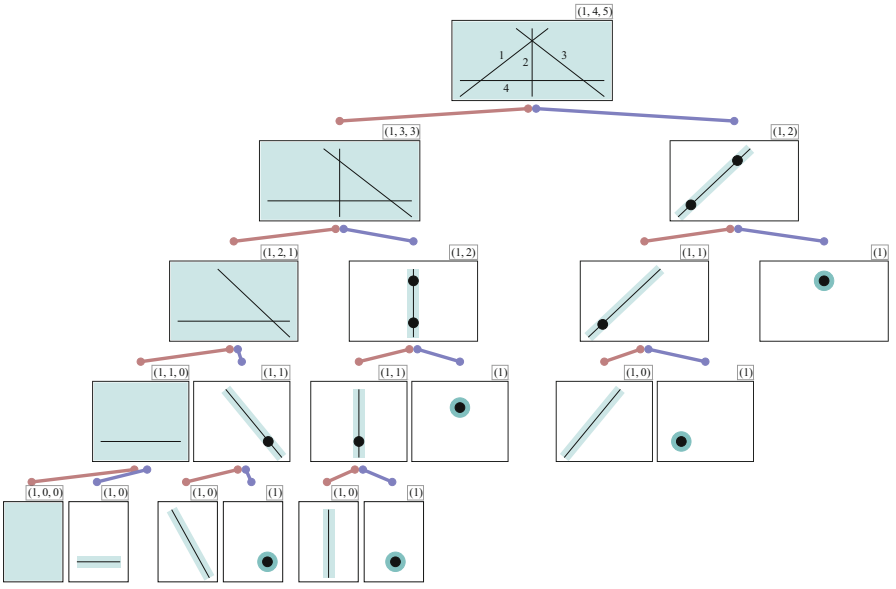


Fig. 3 The tree structure of Algorithm 1 on the hyperplane arrangement from Example 2.2. Hyperplanes are chosen (line 3) according to the ordering $\{1, 2, 3, 4\}$. In each box, the ambient space of the arrangement is shaded green. Deletions are marked with red edges (left children) and restrictions with blue edges (right children). Each arrangement box has the Whitney numbers above its upper right corner

$$B = (\{H_{i_1}, \ldots, H_{i_k}\}, \{H_{j_1}, \ldots, H_{j_\ell}\}) =: (H_I, H_J)$$

represents the hyperplane arrangement $\mathcal{B}$ in $L_I \cong \mathbb{K}^{d-r(L_I)}$ given by the *hyperplanes* in $\{H_j \cap L_I\}_{j\in J}$. Note that $H_j \cap L_I$ may be empty for some $j \in J$, in which case this intersection does not correspond to any hyperplane. We extend notation regarding $\mathcal{B}$ to its representation B (i.e., $\chi_B(t) := \chi_{\mathcal{B}}(t)$ and $b(B) := b(\mathcal{B})$).

If $H_{j_1} \cap L_I$ is a hyperplane which occurs uniquely with respect to the tuple $(H_{j_1} \cap L_I, \ldots, H_{j_\ell} \cap L_I)$, then $\mathcal{B}^{H_{j_1}\cap L_I}$ and $\mathcal{B} \setminus \{H_{j_1} \cap L_I\}$ are represented by

$$B^{H_{j_1}} := (\{H_{i_1}, \ldots, H_{i_k}, H_{j_1}\}, \{H_{j_2}, \ldots, H_{j_\ell}\}),$$
$$B \setminus \{H_{j_1}\} := (\{H_{i_1}, \ldots, H_{i_k}\}, \{H_{j_2}, \ldots, H_{j_\ell}\}),$$

respectively. Whereas if $H_{j_1} \cap L_I$ is either empty or does not occur uniquely, then $B \setminus \{H_{j_1}\}$ trivially represents the same arrangement as B, namely $\mathcal{B}$. The following computational analogue of Lemma 3.1 establishes how such representations behave under deletion and restriction.

Lemma 3.2 *Let $B = (H_I, H_J)$ represent an arrangement $\mathcal{B}$ and fix $H \in H_J$. If $H \cap L_I$ is a hyperplane which occurs uniquely in the tuple $(H_j \cap L_I)_{j\in J}$ then $\chi_B(t) = \chi_{B\setminus\{H\}}(t) - \chi_{B^H}(t)$ and $b(B) = b(B \setminus \{H\}) + 0|b(B^H)$. Otherwise, we have $\chi_B(t) = \chi_{B\setminus\{H\}}(t)$ and $b(B) = b(B \setminus \{H\})$.*

Proof The first case follows from Lemma 3.1. In the second case, B and $B \setminus \{H\}$ represent the same hyperplane arrangement and the result is trivial. □

The following algorithm is equivalent to Algorithm 1.

Algorithm 2: Whitney numbers via extended deletion and restriction

Input: A representation $B = (H_I, H_J)$ of an arrangement in $\mathbb{K}^d$
Output: The vector of Whitney numbers $b(B)$
WhitneyNumbers $B = (H_I, H_J)$
1 **if** $\emptyset \neq H_J$ **then**
2 choose $H \in H_J$
3 **if** $H \cap L_I \neq \emptyset$ *occurs uniquely in* $(H_j \cap L_I)_{j\in J}$ **then**
4 **return WhitneyNumbers**$(B \setminus \{H\}) + 0|$**WhitneyNumbers**(B^H)
5 **else**
6 **return** ***WhitneyNumbers***$(B \setminus \{H\})$
7 **else**
8 **return** $(1, 0, \ldots, 0)$

Given a hyperplane arrangement $\mathcal{A} = \{H_1, \ldots, H_n\}$ in $\mathbb{K}^d$, Algorithm 2 computes the Whitney numbers $b_i(\mathcal{A})$ when given $A = (\emptyset, \{H_1, \ldots, H_n\})$ as input. This algorithm traverses a binary tree which is essentially the same as the one from Algorithm 1. The only difference is that some edges are extended with nodes that have only one child and so we say it computes the Whitney numbers via *extended* deletion and restriction.

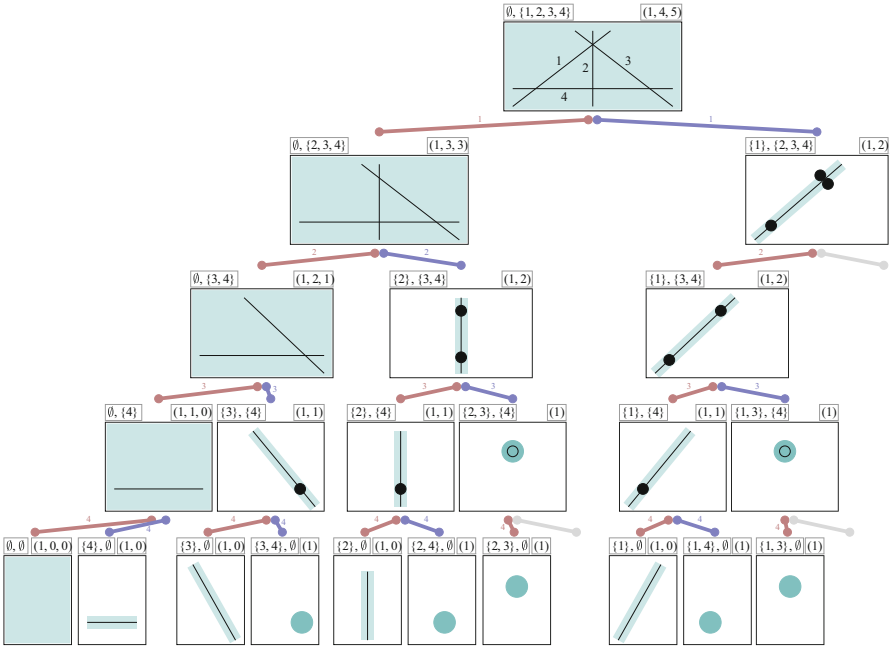


Fig. 4 The tree structure of Algorithm 2 on the hyperplane arrangement from Example 2.2. Its nodes are represented by pairs of subsets $I, J \subset \{1, 2, 3, 4\}$ (top-left) and the Whitney numbers are given (top-right). Grey edges indicate that the condition in line 3 has been violated

Algorithm 2 has the advantage that the representations of the original hyperplanes in $\mathcal{A}$ need not be updated upon restriction, and that representations of hyperplanes in $\mathcal{A}^H$ need not be unique. As a consequence, structural aspects of $\mathcal{A}$ such as its symmetries extend trivially to the representations of the restricted arrangements, as we explain in Sect. 4. Figure 4 displays the tree structure underlying Algorithm 2 on our running example. Note that J is constant amongst nodes in the same depth.

4 Automorphisms of Hyperplane Arrangements

Our main contribution is the inclusion of symmetry-reduction in the deletion-restriction algorithm. Many other algorithms in discrete geometry have also been adapted to take advantage of symmetry [5, 6, 25, 26]. For us, the relevant symmetries for an arrangement are the rank-preserving permutations of its hyperplanes.

Let $\mathfrak{S}_n$ be the permutation group on $[n]$. Elements of a subgroup $G \leq \mathfrak{S}_n$ act on subsets of $[n]$. Given $g \in G$ and $I \subseteq [n]$, we fix the notation:

- $gI = \{g(i)\}_{i \in I}$ for the *image* of I under g,
- $I^G = \{g \in G \mid gI = I\}$ for the *stabilizer* of I in G,
- $G \cdot I = \{gI \mid g \in G\}$ for the *orbit* of I under G.

Definition 4.1 The automorphism group of $\mathcal{A} = \{H_1, \ldots, H_n\}$ is

$$\mathrm{Aut}(\mathcal{A}) = \{g \in \mathfrak{S}_n \mid r(H_I) = r(H_{gI}) \text{ for all } I \subseteq [n]\}.$$

Given a representation $B = (H_I, H_J)$ of an arrangement coming from $\mathcal{A}$, the automorphism group $\mathrm{Aut}(\mathcal{A})$ acts as $gB = (H_{gI}, H_{gJ})$.

Remark 4.2 Our definition of the automorphism group of an arrangement is combinatorial, not geometric. This difference can be quite large. For example, a generic hyperplane arrangement $\mathcal{A} = \{H_1, \ldots, H_n\}$ has no geometric symmetries but $\mathrm{Aut}(\mathcal{A}) = \mathfrak{S}_n$.

Lemma 4.3 *Let $\mathcal{A} = \{H_1, \ldots, H_n\}$ be an arrangement in $\mathbb{K}^d$ and let B_1 and B_2 represent arrangements coming from deletions and restrictions. If B_1 and B_2 are in the same orbit under $\mathrm{Aut}(\mathcal{A})$ then $b(B_1) = b(B_2)$.*

Proof The conclusion of the lemma is equivalent to showing that the characteristic polynomials of B_1 and B_2 are the same. This follows directly from the fact that the characteristic polynomial depends only on the intersection poset (graded by rank) and that B_1 and B_2 are in the same orbit under $\mathrm{Aut}(\mathcal{A})$ if and only if they are related by a rank-preserving permutation. □

Our algorithm relies upon the following corollary of Lemma 4.3.

Corollary 4.4 *Let $B = (H_I, H_J)$ represent a hyperplane arrangement coming from $\mathcal{A} = \{H_1, \ldots, H_n\}$. For $g \in J^{\mathrm{Aut}(\mathcal{A})}$ we have that $gB = (H_{gI}, H_J)$ and B have the same Whitney numbers.*

5 Enumeration Algorithm with Symmetry

Our main algorithm augments Algorithm 2, making particular use of Corollary 4.4. It is essentially a breadth-first tree algorithm except that at each level, nodes may be identified up to symmetry and so the algorithmic structure is no longer that of a tree. The output is the vector of Whitney numbers $b(\mathcal{A})$ of an arrangement $\mathcal{A}$, refining its chamber count. We remark that despite the fact that our algorithm takes advantage of symmetry and counts the number of chambers, it does not reveal any information about the sizes of orbits of chambers under this symmetry group.

Given an arrangement $\mathcal{A} = \{H_1, \ldots, H_n\}$ in $\mathbb{K}^d$, we represent the nodes of the algorithm at depth k by a dictionary T_k. The keys of T_k are orbits $G_k \cdot I$ for $I \subseteq [k]$ where G_k is a subgroup of the stabilizer of $\{k+1, \ldots, n\}$ in $\mathrm{Aut}(\mathcal{A})$. The value of $G_k \cdot I$ in this dictionary is a pair $(B_I, \omega(B_I))$ where B_I represents the hyperplane arrangement $(H_I, H_{\{k+1,\ldots,n\}})$ and $\omega(B_I)$ is some multiplicity, tracking how many arrangements indexed by elements of the orbit $G_k \cdot I$ have appeared. We refer to T_k as a k-th orbit-node dictionary.

Algorithm 3 presents the breadth-first structure of the algorithm.

Algorithm 3: Whitney numbers using symmetry

Input: A hyperplane arrangement $\mathcal{A} = \{H_1, \ldots, H_n\}$ in $\mathbb{K}^d$
A subgroup $G \leq \mathrm{Aut}(\mathcal{A})$
Output: The vector of Whitney numbers $b(\mathcal{A})$
WhitneyNumbers $(\mathcal{A})$
`// compute the stabilizers of` G
compute $\{G_i\}_{i=0}^n$ where $G_i = \{i+1, \ldots, n\}^G$ and $G_n = G$
`// initialize orbit-node dictionaries`
initialize $\{T_i\}_{i=0}^n$ and set $T_0 = \{G_0 \cdot \emptyset \Rightarrow ((\emptyset, \mathcal{A}), 1)\}$
for $k = 1, \ldots, n$ **do**
  set $T_k =$ **NextGeneration**$(\mathcal{A}, G_k, T_{k-1})$
initialize $b = (0, 0, \ldots, 0)$
for $(B_I, \omega(B_I)) \in T_n$ **do**
  increment the entry $b_{|I|}$ by $\omega(B_I)$
return b

Moving from depth $k-1$ to k is performed by Algorithm 4.

Algorithm 4: NextGeneration

Input: A hyperplane arrangement $\mathcal{A} = \{H_1, \ldots, H_n\}$ in $\mathbb{K}^d$
A subgroup $G_k \leq \{k+1, \ldots, n\}^{\mathrm{Aut}(\mathcal{A})}$
An orbit-node dictionary T_{k-1}
Output: An orbit-node dictionary T_k
NextGeneration $(\mathcal{A}, G_k, T_{k-1})$
set $J = \{k+1, \ldots, n\}$
for $(B_I, \omega(B_I)) \in$ `values`(T_{k-1}) **do**
  if $H_k \cap L_I$ *is a unique hyperplane amongst* $(H_j \cap L_I)_{j=k}^n$ **then**
    `// produce the restriction as the right child`
    set $I' = I \cup \{k\}$
    compute the orbit $\mathcal{O} = G_k \cdot I'$
    if $\mathcal{O} \in$ `keys`(T_k) **then**
      increment the multiplicity of $T_k[\mathcal{O}]$ by $\omega(B_I)$
    else
      $T_k[\mathcal{O}] = ((H_{I'}, H_J), \omega(B_I))$
  `// produce the deletion as the left child`
  compute the orbit $\mathcal{O} = G_k \cdot I$
  if $\mathcal{O} \in$ `keys`(T_k) **then**
    increment the multiplicity of $T_k(\mathcal{O})$ by $\omega(B_I)$
  else
    $T_k[\mathcal{O}] = ((H_I, H_J), \omega(B_I))$
return T_k

Example 5.1 The structure underlying Algorithm 3 applied to the arrangement in Example 2.2 is shown in Fig. 5. It is no longer a tree but may be obtained from the tree in Fig. 4 by identifying nodes under the stabilizers of $\mathrm{Aut}(\mathcal{A})$. Each identifi-

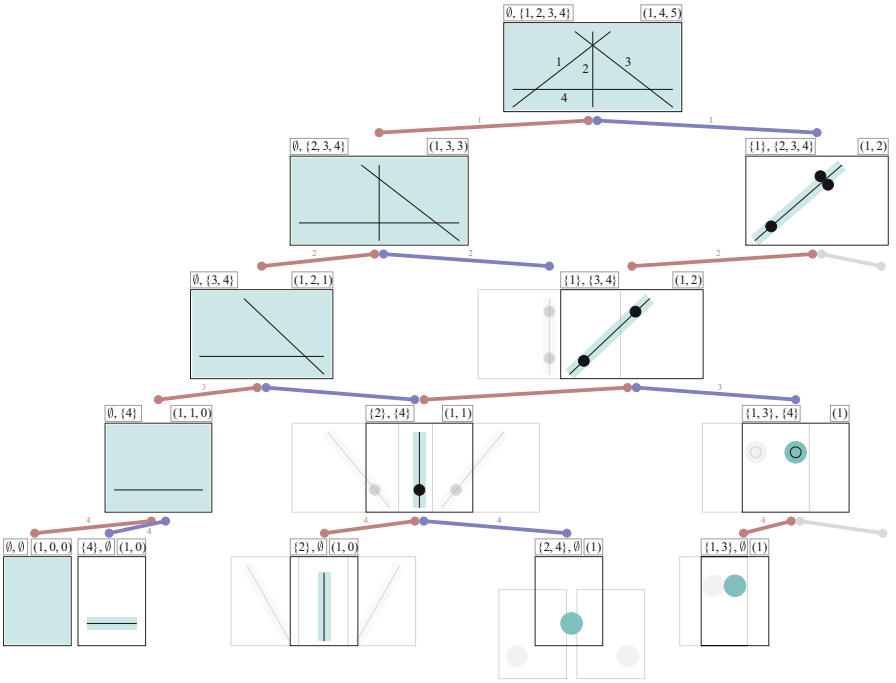


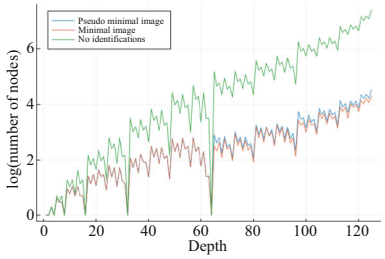
Fig. 5 The algorithmic structure underlying Algorithm 3. Starting at the top node, each call of Algorithm 4 produces the next depth of this graph

cation accumulates multiplicity in the node and that multiplicity is passed down to its children.

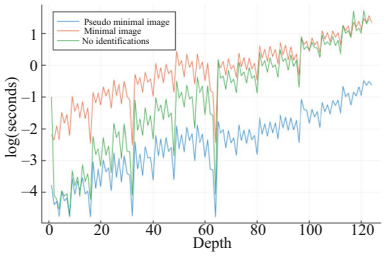
5.1 Representing Orbits

The computations of orbits in lines 5 and 10 require elaboration; specifically in regards to representing an orbit $G \cdot I$ on a computer. One option is to use a canonical element of $G \cdot I$, which can be computed using the `MinimalImage` or `CanonicalImage` functions from `GAP` [23, 24]. An alternative approach is to provide any function $\varphi\colon 2^{[n]} \to S$ taking values in an arbitrary set S such that $\varphi(I) = \varphi(J)$ only if $G \cdot I = G \cdot J$. Equivalently, φ is any factor of the projection $\pi : 2^{[n]} \to 2^{[n]}/G$ as a map of sets where $2^{[n]}/G$ is the set of orbits. In this case, the value of $\varphi(I)$ may be used to represent the orbit $G \cdot I$ as a key in the orbit–node dictionaries. While this approach may fail to identify all nodes in the same orbit, nodes in distinct orbits are never identified and so the algorithm remains correct. The benefit is that it may be significantly more efficient to evaluate φ than it is to compute minimal or canonical images.

Our default option for identifying orbits is called `pseudo_minimal_image`. Given a subset $I \subseteq [n]$ and a collection of elements $g_1, \ldots, g_m \in G \leq \mathfrak{S}_n$, this function sequentially computes $g_i I$ and recursively calls itself on $g_i I$ whenever



(a) Leaves at depth.



(b) Time per depth.

Fig. 6 The leaves per depth and time per depth of Algorithm 3 on the arrangement $\mathcal{R}_7$ using `pseudo_minimal_image`, `MinimalImage`, and no identifications

$g_i I < I$ lexicographically. If no such g_i produces a smaller subset, I itself is returned. Options are implemented for choosing m to be a proportion of $|G|$ subject to maximum and minimum values. For our computations, we take $m = n$ random elements of G. Although this greedy procedure does not make all possible identifications in the algorithm, we have found that it is quicker than `MinimalImage` to evaluate and produces a comparably small algorithmic structure.

Example 5.2 We compare the effect of three choices of identifications in Algorithm 3 (either `pseudo_minimal_image`, the `MinimalImage` function in `GAP`, or no identifications at all) on the resonance arrangement $\mathcal{R}_7$ (see Definition 6.3) consisting of 127 hyperplanes in $\mathbb{R}^7$. We compare the number of leaves of the algorithm at some depth, as well as the time per depth of the algorithm and display the results in Fig. 6.

As depicted, the cost (in number of leaves) of using `pseudo_minimal_image` compared to `MinimalImage` is negligible, while the benefits in terms of speed are significant. Similarly, while the timing of our algorithm with `MinimalImage` is comparable to the timing without any identifications (Algorithm 2), the memory usage is significantly reduced as conveyed by the number of leaves (a reasonable proxy for memory usage). This difference becomes even more dramatic for larger arrangements.

5.2 Accumulating the Whitney Numbers and Skipping Levels

Much of the computational burden occurs in line 3 of Algorithm 4 and involves projecting the normal vectors of the hyperplanes in $\mathcal{A}$ along those hyperplanes which

have been restricted. When implementing Algorithm 4, one may choose whether to save such computations at the cost of memory, or to perform redundant computations throughout the algorithm. We found that, for our benchmark examples, recomputation held the most benefit.

Nonetheless, from the linear algebra involved in the evaluation of line 3, one can read off $j_{\min}$, the smallest $j \in J$ for which this uniqueness condition is true. Hence, one may immediately place the left child of the corresponding node in level $j_{\min}$ rather than k to avoid redundancy in line 3 later on. This comes at the cost of missing some identifications between the layers k and $j_{\min}$.

Another implementation choice we made was to keep a running count of the Whitney numbers of the arrangement throughout the algorithm. Whenever $j_{\min} = n$ while computing the children of $(B_I, \omega(B_I))$, we increment $b_{|I|}$ by $\omega(B_I)$ and delete the node altogether since no other deletions or restrictions are possible. Similarly, if $\mathcal{A}$ is a hyperplane arrangement where each hyperplane contains the origin, $b_{|I|}$ and $b_{|I|+1}$ are incremented by $\omega(B_I)$ whenever $j_{\min} = n - 1$ by a similar reasoning. In this way, we can free memory occupied by nodes throughout the algorithm.

5.3 Relation to `OSCAR`

The new computer algebra system `OSCAR` in `julia` combines the existing systems `GAP` [50], `Singular` [14], `Polymake` [18, 27], and `Antic` (`Hecke`, `Nemo`) [51]. Our software is written in `julia` and builds heavily on these cornerstones. Specifically, we use the number theory components `Nemo` [17] and `Hecke` to work with arrangements defined over algebraic field extensions of $\mathbb{Q}$. For example the separability arrangement of the vertices of the 600-cell is defined over $\mathbb{Q}(\sqrt{5})$. Secondly, we use `GAP` [50] for group theoretic computations in Algorithm 4. Concretely, we compute stabilizers and minimal images using the `GAP` packages `ferret` [22] and `images` [24], respectively.

5.4 Functionality of `CountingChambers.jl`

The `julia` package titled `CountingChambers.jl` contains our implementation and is available at https://mathrepo.mis.mpg.de/CountingChambers. The following code snippet shows some standard functions of our package applied to the arrangement introduced in Example 2.2. A collection of hyperplanes defined by the equations $\ell_i(x_1, \ldots, x_d) = c_i$ for $1 \leq i \leq n$ is encoded by a $d \times n$ matrix A having the coefficients of ℓ_i as columns and a vector c.

```
julia> A = [-1 1 1 0; 1 0 1 1];
julia> c = [1, 0, 1, 0];
julia> whitney_numbers(A; ConstantTerms=c)
3-element Vector{Int64}:
1 4 5
julia> characteristic_polynomial(A; ConstantTerms=c)
t^2 - 4*t + 5
julia> number_of_chambers(A; ConstantTerms=c)
10
```

Note that the automorphism group of this arrangement is $\mathfrak{S}_3 \hookrightarrow \mathfrak{S}_4$ consisting of permutations of the first three hyperplanes. This group can be passed to our algorithm via a list of generators in one-line notation:

```
julia> G = [[2,3,1,4],[2,1,3,4]];
julia> whitney_numbers(A; ConstantTerms=c, SymmetryGroup=G)
3-element Vector{Int64}:
1 4 5
```

As it is easy to run `julia` on multiple threads, we also implemented our algorithm to take advantage of this. By starting `julia` via the command `julia -threads NUM_THREADS` and passing the optional parameter `multi_threaded=true` to our methods, the **for** loop in Algorithm 4 is executed in parallel. Table 2 shows how the multithreading scales.

6 Examples and Integer Sequences

We apply our algorithm to a number of examples. Many of these arise from the following construction of separability arrangements.

6.1 Separability Arrangements

Fix a finite set $\mathbf{V} \subset \mathbb{R}^d$. We associate to every $v \in \mathbf{V}$ the hyperplane $H_v \subset (\mathbb{R}^{d+1})^*$ comprised of linear forms which vanish on $(1, v)$. Equivalently, H_v represents the affine hyperplanes in $\mathbb{R}^d$ which contain v. We call the arrangement $\mathcal{H}_\mathbf{V} := \{H_v \mid v \in \mathbf{V}\}$ the separability arrangement of $\mathbf{V}$. We point out that by increasing the dimension d by one, this construction is distinct from the one which defines *real reflection arrangements* from root systems. In particular, translating $\mathbf{V}$ does not change the combinatorics of $\mathcal{H}_\mathbf{V}$.

A hyperplane H_v partitions the points in $(\mathbb{R}^{d+1})^* \setminus H_v$ into the sets H_v^+ of linear forms which are positive on v and H_v^- which are negative on v. Consequently, all affine hyperplanes corresponding to points in a chamber of $\mathcal{H}_\mathbf{V}$ are positive on some subset $V_1 \subset \mathbf{V}$ and negative on its complement $V_2 = \mathbf{V} \setminus V_1$. Such a partition $V_1 \sqcup V_2 = \mathbf{V}$ is called linearly separable. Hence, chambers of $\mathcal{H}_\mathbf{V}$ are in bijection with linearly separable partitions of $\mathbf{V}$, motivating the terminology for $\mathcal{H}_\mathbf{V}$. This point of view, which connects linear separability and hyperplane arrangements, appears in [2, Sect. 2].

One purpose for introducing separability arrangements is that it immediately provides us with a zoo of arrangements admitting considerable symmetry; for example, those $\mathbf{V}$ which are the vertices of regular polytopes.

6.2 The Threshold Arrangement

The following arrangement appears in the study of neural networks [35, 36, 49] and algebraic statistics [13].

Definition 6.1 The threshold arrangement[2], $\mathcal{T}_d$ is the separability arrangement associated to the vertices of the hypercube $[0, 1]^d$. That is,

$$\mathcal{T}_d := \{\{x_0 + c_1x_1 + \cdots + c_dx_d = 0\} \text{ with } c_i \in \{0, 1\} \text{ for all } c_i\}.$$

As a consequence of the definition of $\mathcal{T}_d$, the linear automorphisms of the hypercube $[0, 1]^d$, namely the hyperoctahedral group of order $d!2^d$, is a subgroup of $\mathrm{Aut}(\mathcal{T}_d)$. The true size of $\mathrm{Aut}(\mathcal{T}_d)$ is $(d + 1)!2^d$.

We computed the Whitney numbers of $\mathcal{T}_d$ for $1 \leq d \leq 8$, and thus their number of chambers. The results are collected in Table 3 and the timings appear in Table 1. The values of $|\mathrm{ch}(\mathcal{T}_d)|$ for $1 \leq d \leq 9$ are listed in entry https://oeis.org/A000609 of the Online-Encyclopedia of Integer Sequences (OEIS), whereas the Whitney numbers of $\mathcal{T}_d$, to the best of our knowledge, have not been published before. Zuev showed that asymptotically $|\mathrm{ch}(\mathcal{T}_d)| \sim 2^{d^2}$ [48].

Remark 6.2 Using similar proof techniques as in [32] one can show that the values of $b_i(\mathcal{T}_d)$ for $1 \leq d \leq 2^i$ determine a formula for $b_i(\mathcal{T}_d)$ for all d. Applying this to the case of $b_2(\mathcal{T}_d)$ and $b_3(\mathcal{T}_d)$ and using the results in Table 3 we obtain $b_2(\mathcal{T}_d) = (4^d - 2^d)/2$ and $b_3(\mathcal{T}_d) = (4 \cdot 8^d - 3 \cdot 6^d - 6 \cdot 4^d + 5 \cdot 2^d)/24$. For $i \geq 4$ this technique requires knowledge of $b_i(\mathcal{T}_d)$ for at least $1 \leq d \leq 16$.

6.3 The Resonance Arrangement

The next arrangement we consider appears as a restriction of the threshold arrangement.

Definition 6.3 The resonance arrangement is the restriction of $\mathcal{T}_d$ to the hyperplane $H_{(0,\ldots,0)}$. Equivalently, for $d \geq 1$ the resonance arrangement is

$$\mathcal{R}_d := \{\{c_1x_1 + c_2x_2 + \cdots + c_dx_d = 0\} \text{ with } c_i \in \{0, 1\} \text{ and not all } c_i \text{ are zero}\}.$$

The chambers of the resonance arrangements are in bijection with generalized retarded functions in quantum field theory [16]. An overview of the applications of the resonance arrangement is given in [32, Sect. 1]. A formula for their number of chambers remains elusive, let alone one for their Whitney numbers. Nonetheless, partial formulas and bounds exist [4, 19, 32, 48].

The numbers of chambers of the resonance arrangements are listed in the sequence https://oeis.org/A034997 in the OEIS up to $d = 9$. The Whitney numbers are published in [28] up to $d = 7$. Our software was able to determine the Whitney numbers of $\mathcal{R}_8$ and $\mathcal{R}_9$ confirming the concurrent computations in [10]. The computation for $\mathcal{R}_9$ took ten days, running multithreaded on 42 `Intel Xeon E7-8867 v3` CPUs. All Whitney numbers of $\mathcal{R}_d$ up to $d = 9$ are given in Table 4 and the timings are listed in Table 1.

[2] The arrangement $\{x_i + x_j\}_{1 \leq i < j \leq d}$ in $\mathbb{R}^d$ is also referred to as a threshold arrangement in the literature. We discuss the arrangement $\mathcal{T}_d$ only as in Definition 6.1.

6.4 Separability Arrangements of the Cross-Polytopes

The cross-polytope of dimension d is the polytope with the $2d$ vertices $\{\pm e_i\}_{i=1}^{d}$. Its symmetry group is the hyperoctahedral group of order $d!2^d$. We define the arrangement $\mathcal{C}_d$ in $\mathbb{R}^{d+1}$ to be the separability arrangement of its vertices. Our computations show that $|\mathrm{ch}(\mathcal{C}_d)| = 2 \cdot 3^d - 2^d$ for $d \leq 20$, suggesting that $|\mathrm{ch}(\mathcal{C}_d)|$ agrees with this sequence (https://oeis.org/A027649 in the OEIS). This can indeed be proven by applying Athanasiadis' finite field method [1] and seems to be a new result obtained through experiments with our algorithm.

6.5 Separability Arrangements of Permutohedra

The permutohedron of dimension $d-1$ is the convex hull of the $d!$ points $\sigma(1, \ldots, d)$ for all $\sigma \in \mathfrak{S}_d$. The separability arrangements $\mathcal{P}_d$ of these points in $\mathbb{R}^{d+1}$ consist of $d!$ hyperplanes. We record their Whitney numbers in Table 6 for $1 \leq d \leq 6$.

6.6 Separability Arrangements of Demicubes

The d-demicube is the convex hull of those vertices of the hypercube $[0, 1]^d$ which have an odd number of 1's. For instance, the 3-demicube is a regular tetrahedron. We denote by $\mathcal{D}_d$ the corresponding separability arrangement consisting of 2^{d-1} hyperplanes in $\mathbb{R}^{d+1}$. Table 5 contains the Whitney numbers of $\mathcal{D}_d$ up to $d = 9$.

6.7 Separability Arrangements of Some Regular Polytopes

In Table 7, we provide the Whitney numbers for the separability arrangements corresponding to the remaining two Platonic solids: the icosahedron and the dodecahedron. This table also contains the Whitney numbers of the separability arrangements of the vertices of the regular 24-cell, 600-cell, and 120-cell. Except for the 24-cell, each of these computations uses irrational realizations.

6.8 Discriminantal Arrangements

Given n points in $\mathbb{R}^d$ in general position, the discriminantal arrangement $\mathrm{Disc}_{d,n}$ is the hyperplane arrangement in $\mathbb{R}^d$ consisting of the $\binom{n}{d}$ hyperplanes spanned by d-subsets of such points. This arrangement, originally called the "geometry of circuits" was introduced by Crapo [11]. We verify the Whitney numbers of $\mathrm{Disc}_{4,n}$ for $5 \leq n \leq 16$ given in [31, Sect. 4.4]. From this data, we recover their formula for the characteristic polynomial of $\mathrm{Disc}_{4,n}$ for all n. A deformation of this arrangement appears in physics [8, 9] and we were able to confirm the chamber counts given in these papers.

7 Timings

While other pieces of software for counting chambers of arrangements exist, they do not take advantage of symmetry and some compute significantly more data than our algorithm does. Consequently, our software outperforms them with respect to the calculation of Whitney numbers as shown below.

The implementation [29] in `polymake` computes much more information than the Whitney numbers, namely a chamber decomposition of the arrangement. The `sage` implementation, on the other hand, uses basic deletion and restriction as in Algorithm 2. Similarly, the `GAP` package `alcove` [33] computes the Tutte polynomial by simple deletion and restriction and then specializes this to the characteristic polynomial.

To illustrate the performance of our software on the arrangements from Sect. 6, we collect our timings in Table 1. This table also shows the growth in complexity for computing the number of chambers of these arrangements. Based on our profiling, the main bottleneck in our implementation is the identifications of orbits. Thus, improving `pseudo_minimal_image` would be the most direct method for making our code faster.

Acknowledgements This research was completed while the first and third author were at the Max Planck Institute for Mathematics in the Sciences. The third author was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation)—SFB-TRR 358/1 2023—491392403. We are very grateful to Tommy Hofmann, Christopher Jefferson, and Marek Kaluba for their support regarding the implementation, and to Michael Cuntz for initial verifications of our computations. We would also like to thank Michael Joswig for his helpful comments throughout the project and Bernd Sturmfels for suggesting the discriminantal arrangement. Lastly, we thank the referees for their careful reading and helpful comments.

Funding Open Access funding enabled and organized by Projekt DEAL.

Appendix: Tables of Whitney Numbers

Table 2 Comparison of the effect of number of threads on run times (`Intel Core i7-8700`)

$\mathcal{A}$	# Threads = 1	2	4	8	12
$\mathcal{R}_8$ (min)	19.8	10.5	6.3	5.9	5.1
$\mathcal{T}_8$ (h)	8.16	3.9	2.4	1.8	1.6

Table 3 The values of $b_i(\mathcal{T}_d)$ and $|\mathrm{ch}(\mathcal{T}_d)|$ of the threshold arrangement for $1 \le d \le 9$ and $0 \le i \le d$

d	1	2	3	4	5	6	7	8
$b_0(\mathcal{T}_d)$	1	1	1	1	1	1	1	1
$b_1(\mathcal{T}_d) = \lvert\mathcal{T}_d\rvert$	2	4	8	16	32	64	128	256
$b_2(\mathcal{T}_d)$	1	6	28	120	496	2016	8128	32640
$b_3(\mathcal{T}_d)$		3	44	460	4240	36848	310464	2569920
$b_4(\mathcal{T}_d)$			23	820	19660	400400	7493808	133492800
$b_5(\mathcal{T}_d)$				465	43014	2453248	112965776	4626016752
$b_6(\mathcal{T}_d)$					27129	7111650	987779688	103818315888
$b_7(\mathcal{T}_d)$						5023907	4075759064	1382897843304
$b_8(\mathcal{T}_d)$							3193753807	8676817935144
$b_9(\mathcal{T}_d)$								7393243346241
$\lvert\mathrm{ch}(\mathcal{T}_d)\rvert$	2	14	104	1882	94572	15028134	8378070864	17561539552946

Table 4 The values of $b_i(\mathcal{R}_d)$ and $|\mathrm{ch}(\mathcal{R}_d)|$ of the resonance arrangement for $1 \le d \le 9$ and $0 \le i \le d$

d	1	2	3	4	5	6	7	8	9
$b_0(\mathcal{R}_d)$	1	1	1	1	1	1	1	1	1
$b_1(\mathcal{R}_d) = \lvert\mathcal{R}_d\rvert$	1	3	7	15	31	63	127	255	511
$b_2(\mathcal{R}_d)$		2	15	80	375	1652	7035	29360	120975
$b_3(\mathcal{R}_d)$			9	170	2130	22435	215439	1957200	17153460
$b_4(\mathcal{R}_d)$				104	5270	159460	3831835	81029004	1582492380
$b_5(\mathcal{R}_d)$					3485	510524	37769977	2076831708	96834110730
$b_6(\mathcal{R}_d)$						371909	169824305	30623870732	3829831100340
$b_7(\mathcal{R}_d)$							135677633	207507589302	89702833260450
$b_8(\mathcal{R}_d)$								178881449368	973784079284874
$b_9(\mathcal{R}_d)$									887815808473419
$\lvert\mathrm{ch}(\mathcal{R}_d)\rvert$	2	6	32	370	11292	1066044	347326352	419172756930	1955230985997140

We submitted these Whitney numbers to the OEIS as the sequence https://oeis.org/A344494

Table 5 The values of $b_i(\mathcal{D}_d)$ and $|\mathrm{ch}(\mathcal{D}_d)|$ of the demicube arrangement for $2 \le d \le 9$ and $0 \le i \le d+1$

d	2	3	4	5	6	7	8	9
$b_0(\mathcal{D}_d)$	1	1	1	1	1	1	1	1
$b_1(\mathcal{D}_d) = \lvert\mathcal{D}_d\rvert$	2	4	8	16	32	64	128	256
$b_2(\mathcal{D}_d)$	1	6	28	120	496	2016	8128	32640
$b_3(\mathcal{D}_d)$	0	4	50	500	4480	38304	319200	2622400
$b_4(\mathcal{D}_d)$		1	44	1160	24340	461496	8283744	143504320
$b_5(\mathcal{D}_d)$			15	1362	76364	3486448	143595816	5483536464
$b_6(\mathcal{D}_d)$				597	120942	15440376	1615624080	145378334304
$b_7(\mathcal{D}_d)$					64903	33803416	10878083096	2574289938400
$b_8(\mathcal{D}_d)$						21424343	35828091880	27816202212040
$b_9(\mathcal{D}_d)$							26430009593	146101801794362
$b_{10}(\mathcal{D}_d)$								120719853808577
$\lvert\mathrm{ch}(\mathcal{D}_d)\rvert$	4	16	146	3756	291558	74656464	74904015666	297363155783764

Table 6 The values of $b_i(\mathcal{P}_d)$ and $|\mathrm{ch}(\mathcal{P}_d)|$ of the permutohedron arrangement for $1 \leq d \leq 6$ and $0 \leq i \leq d$

d	1	2	3	4	5	6
$b_0(\mathcal{P}_d)$	1	1	1	1	1	1
$b_1(\mathcal{P}_d) = \lvert\mathcal{P}_d\rvert$	1	2	6	24	120	720
$b_2(\mathcal{P}_d)$		1	15	276	7140	258840
$b_3(\mathcal{P}_d)$			10	1423	246605	59577390
$b_4(\mathcal{P}_d)$				1170	4290610	9271534305
$b_5(\mathcal{P}_d)$					4051026	834595018036
$b_6(\mathcal{P}_d)$						825382803000
$\lvert\mathrm{ch}(\mathcal{P}_d)\rvert$	2	4	32	2894	8595502	1669309192292

Table 7 The values of $b_i(\mathcal{A})$ and $|\mathrm{ch}(\mathcal{A})|$ of the icosahedral and dodecahedral arrangements as well as arrangements stemming from regular 4-polytopes for $0 \leq i \leq 5$

Polytope	Icosahedron	Dodecahedron	24-Cell	600-Cell	120-Cell
$b_0(\mathcal{A})$	1	1	1	1	1
$b_1(\mathcal{A}) = \lvert\mathcal{A}\rvert$	12	20	24	120	600
$b_2(\mathcal{A})$	66	166	276	7140	179700
$b_3(\mathcal{A})$	157	577	1630	225782	31972550
$b_4(\mathcal{A})$	102	430	4308	3118740	2979870540
$b_5(\mathcal{A})$	–	–	2931	2899979	2948077091
$\lvert\mathrm{ch}(\mathcal{A})\rvert$	338	1194	9170	6251762	5960100482

References

1. Athanasiadis, Ch.A.: Characteristic polynomials of subspace arrangements and finite fields. Adv. Math. **122**(2), 193–233 (1996)
2. Baldi, P., Vershynin, R.: Polynomial threshold functions, hyperplane arrangements, and random tensors. SIAM J. Math. Data Sci. **1**(4), 699–729 (2019)
3. Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B.: Julia: a fresh approach to numerical computing. SIAM Rev. **59**(1), 65–98 (2017)
4. Billera, L.J., Moore, J.T., Moraites, C.D., Wang, Y., Williams, K.: Maximal unbalanced families (2012). arXiv:1209.2309
5. Bremner, D., Dutour Sikirić, M., Pasechnik, D.V., Rehn, Th., Schürmann, A.: Computing symmetry groups of polyhedra. LMS J. Comput. Math. **17**(1), 565–581 (2014)
6. Bremner, D., Dutour Sikirić, M., Schürmann, A.: Polyhedral representation conversion up to symmetries. In: Polyhedral Computation (Montréal 2006). CRM Proc. Lecture Notes, vol. 48, pp. 45–71. American Mathematical Society, Providence (2009)
7. Brylawski, T.: The broken-circuit complex. Trans. Am. Math. Soc. **234**(2), 417–433 (1977)
8. Cachazo, F., Early, N., Guevara, A., Mizera, S.: Scattering equations: from projective spaces to tropical Grassmannians. J. High Energy Phys. **2019**(6), # 39 (2019)
9. Cachazo, F., Umbert, B., Zhang, Y.: Singular solutions in soft limits. J. High Energy Phys. **2020**(5), # 148 (2020)
10. Chroman, Z., Singhal, M.: Computations associated with the resonance arrangement (2021). arXiv:2106.09940

11. Crapo, H.: The combinatorial theory of structures. In: Matroid Theory (Szeged 1982). Colloquia Mathematica Societatis János Bolyai, vol. 40, pp. 107–213. North-Holland, Amsterdam (1985)
12. Crapo, H.H., Rota, G.-C.: On the Foundations of Combinatorial Theory: Combinatorial Geometries. MIT Press, Cambridge (1970)
13. Cueto, M.A., Morton, J., Sturmfels, B.: Geometry of the restricted Boltzmann machine. In: Algebraic Methods in Statistics and Probability II (Urbana-Champaign 2009). Contemporary Mathematics, vol. 516, pp. 135–153. American Mathematical Society, Providence (2010)
14. Decker, W., Greuel, G.-M., Pfister, G., Schönemann, H.: SINGULAR 4-2-0—A computer algebra system for polynomial computations (2020). http://www.singular.uni-kl.de
15. Deza, A., Pournin, L.: A linear optimization oracle for zonotope computation. Comput. Geom. **100**, # 101809 (2022)
16. Evans, T.: What is being calculated with thermal field theory? In: Particle Physics and Cosmology (Lake Louise 1994), pp. 343–352. World Scientific, Singapore (1995)
17. Fieker, C., Hart, W., Hofmann, T., Johansson, F.: Nemo/Hecke: computer algebra and number theory packages for the Julia programming language. In: 42nd International Symposium on Symbolic and Algebraic Computation (Kaiserslautern 2017), pp. 157–164. ACM, New York (2017)
18. Gawrilow, E., Joswig, M.: polymake: a framework for analyzing convex polytopes. In: Polytopes—Combinatorics and Computation (Oberwolfach 1997). DMV Seminar, vol. 29, pp. 43–73. Birkhäuser, Basel (2000)
19. Gutekunst, S.C., Mészáros, K., Petersen, T.K.: Root cones and the resonance arrangement. Electron. J. Comb. **28**(1), # P1.12 (2021)
20. Halperin, D., Sharir, M.: Arrangements. In: Handbook of Discrete and Computational Geometry, 3rd edn, pp. 723–762 (chapter 28). Chapman & Hall/CRC, Boca Raton (2018)
21. Huh, J., Katz, E.: Log-concavity of characteristic polynomials and the Bergman fan of matroids. Math. Ann. **354**(3), 1103–1116 (2012)
22. Jefferson, Ch.: ferret—GAP package, v. 1.0.2 (2019). https://gap-packages.github.io/ferret/
23. Jefferson, Ch., Jonauskyte, E., Pfeiffer, M., Waldecker, R.: Minimal and canonical images. J. Algebra **521**, 481–506 (2019)
24. Jefferson, Ch., Pfeiffer, M., Waldecker, R., Jonauskyte, E.: images—GAP package, v. 1.3.0 (2019). https://gap-packages.github.io/images/
25. Jensen, A.N.: Traversing symmetric polyhedral fans. In: 3rd International Congress on Mathematical Software (Kobe 2010). Lecture Notes in Computer Science, vol. 6327, pp. 282–294. Springer, Berlin (2010)
26. Jordan, Ch., Joswig, M., Kastner, L.: Parallel enumeration of triangulations. Electron. J. Comb. **25**(3), # P3.6 (2018)
27. Kaluba, M., Lorenz, B., Timme, S.: Polymake.jl: a new interface to polymake. In: 7th International Conference on Mathematical Software (Braunschweig 2020). Lecture Notes in Computer Science, vol. 12097, pp. 377–385. Springer, Cham (2020)
28. Kamiya, H., Takemura, A., Terao, H.: Ranking patterns of unfolding models of codimension one. Adv. Appl. Math. **47**(2), 379–400 (2011)
29. Kastner, L., Panizzut, M.: Hyperplane arrangements in polymake. In: 7th International Conference on Mathematical Software (Braunschweig 2020). Lecture Notes in Computer Science, vol. 12097, pp. 232–240. Springer, Cham (2020)
30. Klivans, C.J., Swartz, E.: Projection volumes of hyperplane arrangements. Discrete Comput. Geom. **46**(3), 417–426 (2011)
31. Koizumi, H., Numata, Y., Takemura, A.: On intersection lattices of hyperplane arrangements generated by generic points. Ann. Comb. **16**(4), 789–813 (2012)
32. Kühne, L.: The universality of the resonance arrangement and its Betti numbers. Sém. Lothar. Comb. **85B**, # 75 (2021)
33. Leuner, M.: alcove—GAP package (2019). https://github.com/martin-leuner/alcove
34. Möller, T., Röhrle, G.: Counting chambers in restricted Coxeter arrangements. Arch. Math. (Basel) **112**(4), 347–359 (2019)
35. Montúfar, G., Ay, N., Ghazi-Zahedi, K.: Geometry and expressive power of conditional restricted Boltzmann machines. J. Mach. Learn. Res. **16**, 2405–2436 (2015)
36. Montúfar, G.F., Morton, J.: When does a mixture of products contain a product of mixtures? SIAM J. Discrete Math. **29**(1), 321–347 (2015)

37. Orlik, P., Solomon, L.: Combinatorics and topology of complements of hyperplanes. Invent. Math. **56**(2), 167–189 (1980)
38. Orlik, P., Terao, H.: Arrangements of Hyperplanes. Grundlehren der Mathematischen Wissenschaften, vol. 300. Springer, Berlin (1992)
39. Pendavingh, R., van der Pol, J.: Asymptotics of symmetry in matroids. J. Comb. Theory B **135**, 349–365 (2019)
40. Postnikov, A., Stanley, R.P.: Deformations of Coxeter hyperplane arrangements. J. Comb. Theory A **91**(1–2), 544–597 (2000)
41. Sarnoff, J.: SaferIntegers—julia package, v. 2.5.3 (2021). https://github.com/JeffreySarnoff/SaferIntegers.jl
42. Sleumer, N.H.: Output-sensitive cell enumeration in hyperplane arrangements. Nord. J. Comput. **6**(2), 137–147 (1999)
43. Solomon, L., Terao, H.: A formula for the characteristic polynomial of an arrangement. Adv. Math. **64**(3), 305–325 (1987)
44. Stein, W.A.: Sage Mathematics Software, version x.y.z. The Sage Development Team (2021). http://www.sagemath.org
45. Yoshinaga, M.: Hyperplane arrangements and Lefschetz's hyperplane section theorem. Kodai Math. J. **30**(2), 157–194 (2007)
46. Yoshinaga, M.: Freeness of hyperplane arrangements and related topics. Ann. Fac. Sci. Toulouse Math. **23**(2), 483–512 (2014)
47. Zaslavsky, Th.: Facing up to Arrangements: Face-Count Formulas for Partitions of Space by Hyperplanes. Memoirs of the American Mathematical Society, vol. 154. American Mathematical Society, Providence (1975)
48. Zuev, Yu.A.: Methods of geometry and probabilistic combinatorics in threshold logic. Discrete Math. Appl. **2**(4), 427–438 (1992)
49. Zunic, J.: On encoding and enumerating threshold functions. IEEE Trans. Neural Netw. **15**(2), 261–267 (2004)
50. GAP—Groups, Algorithms, and Programming, v. 4.10.2 (2019). http://www.gap-system.org
51. OSCAR—Computer Algebra System, v. 0.5.2 (2021). https://oscar.computeralgebra.de