Max-Planck-Institut für Mathematik in den Naturwissenschaften Leipzig

Enumerating chambers of hyperplane arrangements with symmetry

by

Taylor Brysiewicz, Holger Eble, and Lukas Kühne

Preprint no.: 14 2021

ENUMERATING CHAMBERS OF HYPERPLANE ARRANGEMENTS WITH SYMMETRY

TAYLOR BRYSIEWICZ, HOLGER EBLE, AND LUKAS KÜHNE

Abstract. We introduce a new algorithm for enumerating chambers 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.

1. Introduction

The problem of enumerating chambers of hyperplane arrangements is a well-known challenge in computational discrete geometry [18, 27, 32, 41]. We develop a novel chamber-counting algorithm which takes advantage of the combinatorial symmetries of an arrangement. This count is derived from the computation of other important combinatorial invariants: Betti numbers and characteristic polynomials [1, 19, 28, 36, 42, 44]. While most arrangements admit few combinatorial symmetries [38], most arrangements of interest do [17, 39, 45].

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` [37] for group theory computations (`GAP` [15]) and the ability to work over number fields (`Hecke` and `Nemo` [14]). To the best of our knowledge, it 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.

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 accurate since we compute symbolically over the integers or exact number fields and avoid overflow errors thanks to the package `SaferIntegers.jl` [40].

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 10 days on 42 processors. Our computation confirms that its number of chambers is 1955230985997140 as recently published in the OEIS sequence A034997 by Zachary Chroman.

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

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

Acknowledgements

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.

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 [36].

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 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 are denoted $\mathrm{ch}(\mathcal{A})$.

Example 2. We use the arrangement

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

in $\mathbb{R}^2$ as a running example. This arrangement is depicted in Figure 1. It has 10 chambers: 2 bounded and 8 unbounded.

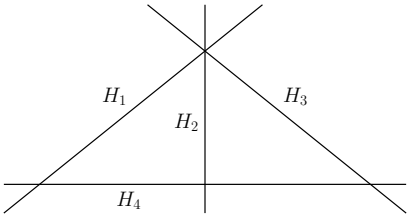


Figure 1. The arrangement introduced in Example 2.

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, $r : L(\mathcal{A}) \to \mathbb{Z}_{\geq 0}$, where $r(L_I) = \operatorname{codim}(L_I)$. As a notational convention, we set $r(I) = r(L_I)$ for $I \subseteq [n]$ whenever $L_I \neq \emptyset$.

2.1. **The characteristic polynomial.** Our algorithm counts chambers of an arrangement by computing a more refined count, namely the Betti numbers. Knowing the Betti numbers of an arrangement is equivalent to knowing its characteristic polynomial.

Definition 3. The characteristic polynomial of an arrangement $\mathcal{A}$ in $\mathbb{K}^d$ is the polynomial

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

The integers $\{b_i(\mathcal{A})\}_{i=0}^{d}$, defined via (1), are non-negative and are called the Betti numbers of $\mathcal{A}$. We denote the vector of Betti numbers by $b(\mathcal{A})$.

The characteristic polynomial and Betti 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 [44] proved that

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

Thus, the Betti numbers are a refined count of the chambers of $\mathcal{A}$. They have the following geometric interpretation. Given a generic flag $\mathcal{F}_\bullet : F_0 \subset F_1 \subset \cdots \subset F_d = \mathbb{R}^d$ 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 $b_i(\mathcal{A})$ [43, 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 i-th *topological* Betti number of the complement $\mathbb{C}^d \setminus \bigcup_{H \in \mathcal{A}} H$ with rational coefficients [35].

Finite: When $\mathcal{A}$ is an arrangement over a finite field $\mathbb{F}_q$, Crapo and Rota proved that $\chi_{\mathcal{A}}(q) = |\mathbb{F}_q^d \setminus \bigcup_{H \in \mathcal{A}} H|$ [10]. Moreover, if $\mathcal{A}$ is a hyperplane arrangement in $\mathbb{Q}^d$ 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 4. Let $\mathcal{A}$ be the arrangement introduced in Example 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 Betti 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.

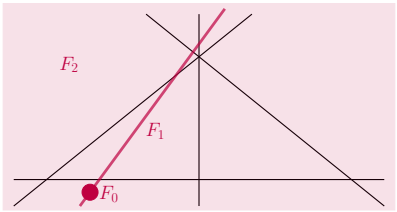


FIGURE 2. The intersections of a generic flag (purple) in $\mathbb{R}^2$ with the chambers of $\mathcal{A}$. The point F_0 intersects one chamber, F_1 intersects four others, and F_2 intersects the remaining 5, and so $b(\mathcal{A}) = (1, 4, 5)$.

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

Lemma 5 [36, Corollary 2.57]**.** *Given a hyperplane $H \in \mathcal{A}$, we have that $\chi_{\mathcal{A}}(t) = \chi_{\mathcal{A}\backslash\{H\}}(t) - \chi_{\mathcal{A}^H}(t)$. In particular, $b(\mathcal{A}) = b(\mathcal{A}\backslash\{H\}) + 0|b(\mathcal{A}^H)$ where $0|b$ means prepending the vector b with a zero.*

3.1. **A simple deletion-restriction algorithm.** Lemma 5 along with the fact that the empty arrangement in $\mathbb{K}^d$ has Betti numbers $(1, 0, \ldots, 0) \in \mathbb{N}^{d+1}$ suggests the following recursive algorithm for computing $b(\mathcal{A})$.

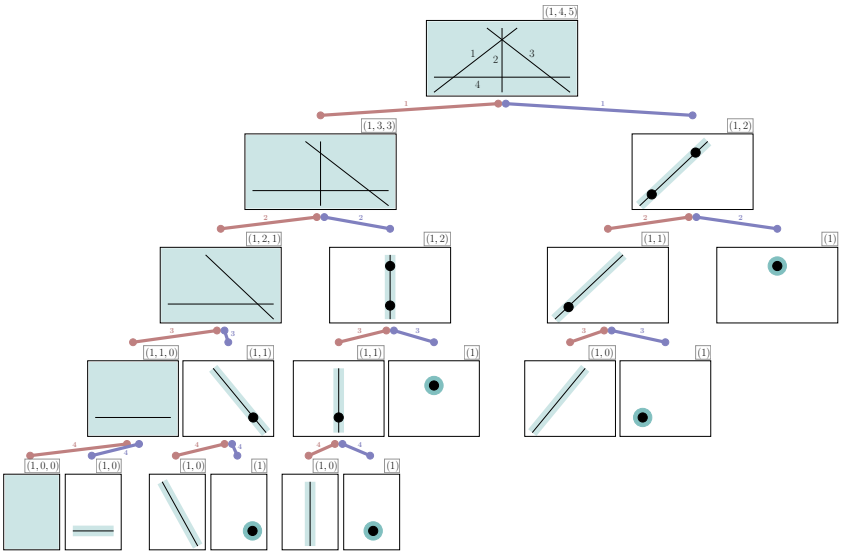


FIGURE 3. The tree structure of Algorithm 1 on the hyperplane arrangement from Example 2. Hyperplanes are chosen (Line 2) according to the ordering $\{1, 2, 3, 4\}$. In each box, the ambient space of the arrangement is shaded green. Deletions are marked with red edges (left children) and restrictions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

Algorithm 1: Betti numbers via simple deletion and restriction

Input: A hyperplane arrangement $\mathcal{A}$ in $\mathbb{K}^d$
Output: The Betti numbers $b(\mathcal{A})$
BettiNumbers $(\mathcal{A})$
1 **if** $\emptyset \neq \mathcal{A}$ **then**
2 choose $H \in \mathcal{A}$
3 **return** ***BettiNumbers***$(\mathcal{A}\backslash\{H\})$ + $0|$***BettiNumbers***$(\mathcal{A}^H)$
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 Figure 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

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

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 $\mathcal{B}\backslash\{H_{j_1} \cap L_I\}$ are represented by

$$\begin{aligned} B^{H_{j_1}} &:= (\{H_{i_1}, \ldots, H_{i_k}, H_{j_1}\}, \{H_{j_2}, \ldots, H_{j_\ell}\}) \\ B\backslash\{H_{j_1}\} &:= (\{H_{i_1}, \ldots, H_{i_k}\}, \{H_{j_2}, \ldots, H_{j_\ell}\}), \end{aligned}$$

respectively. Whereas if $H_{j_1} \cap L_I$ is either empty or does not occur uniquely, then $B\backslash\{H_{j_1}\}$ trivially represents the same arrangement as B, namely $\mathcal{B}$.

The following computational analogue of Lemma 5 establishes how such representations behave under deletion and restriction.

Lemma 6. *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 $\chi_B(t) = \chi_{B\backslash\{H\}}(t) - \chi_{B^H}(t)$ and $b(B) = b(B\backslash\{H\}) + 0|b(B^H)$. Otherwise, we have $\chi_B(t) = \chi_{B\backslash\{H\}}(t)$ and $b(B) = b(B\backslash\{H\})$.*

Proof. The first case follows from Lemma 5. In the second case, B and $B\backslash\{H\}$ represent the same hyperplane arrangement and the result is trivial. □

The following algorithm is equivalent to Algorithm 1.

Algorithm 2: Betti numbers via extended deletion and restriction

Input: A representation $B = (H_I, H_J)$ of an arrangement in $\mathbb{K}^d$
Output: The Betti numbers $b(B)$
BettiNumbers $B = (H_I, H_J)$

```
if ∅ ≠ H_J then
    choose H ∈ H_J
    if H ∩ L_I ≠ ∅ occurs uniquely in (H_j ∩ L_I)_{j∈J} then
        return BettiNumbers(B\{H}) + 0|BettiNumbers(B^H)
    else
        return BettiNumbers(B\{H})
else
    return (1, 0, ..., 0)
```

Given a hyperplane arrangement $\mathcal{A} = \{H_1, \dots, H_n\}$ in $\mathbb{K}^d$, Algorithm 2 computes the Betti numbers $b(\mathcal{A})$ when given $A = (\emptyset, \{H_1, \dots, H_n\})$ 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 Betti numbers via *extended* deletion and restriction.

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 Section 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, 23, 24]. 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 7. The automorphism group of $\mathcal{A} = \{H_1, \dots, H_n\}$ 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 $gB = (H_{gI}, H_{gJ})$.

Lemma 8. *Let $\mathcal{A} = \{H_1, \dots, H_n\}$ 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 $b(B_1) = b(B_2)$.*

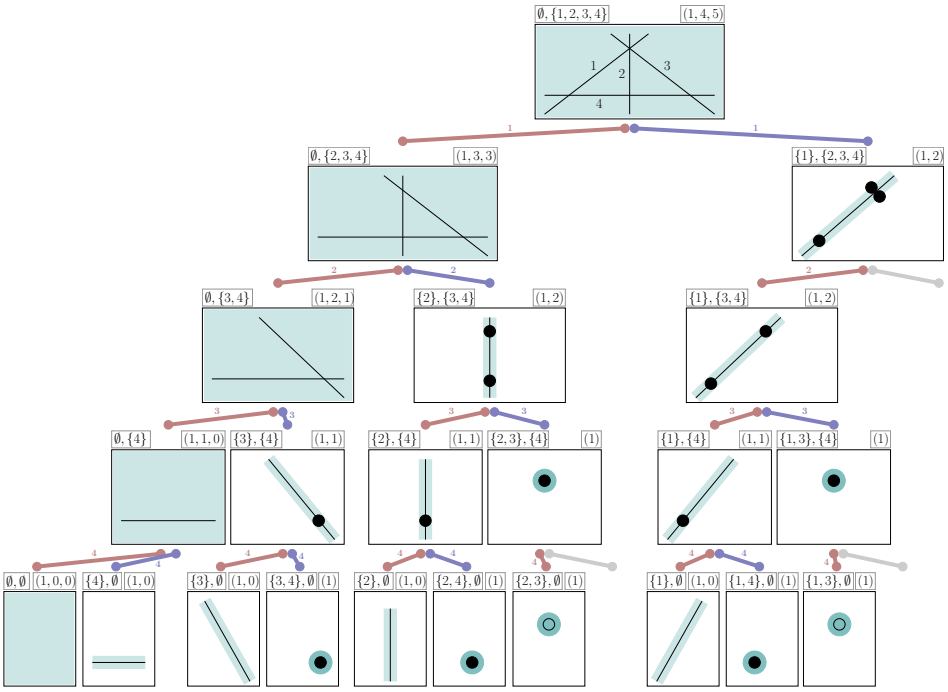


FIGURE 4. The tree structure of Algorithm 2 on the hyperplane arrangement from Example 2. Its nodes are represented by pairs of subsets $I, J \subset \{1, 2, 3, 4\}$ (top-left) and the Betti numbers are given (top-right). Grey edges indicate that the condition in Line 3 has been violated.

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 8.

Corollary 9. *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 Betti numbers.*

5. Enumeration algorithm with symmetry

Our main algorithm augments Algorithm 2, making particular use of Corollary 9. 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.

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: Betti numbers using symmetry

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

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

Algorithm 4: NextGeneration

Input: A hyperplane arrangement $\mathcal{A} = \{H_1, \dots, H_n\}$ in $\mathbb{K}^d$
A subgroup $G_k \leq \{k+1, \dots, n\}^{\mathrm{Aut}(\mathcal{A})}$
An orbit-node dictionary T_{k-1}
Output: An orbit-node dictionary T_k
NextGeneration $(\mathcal{A}, G_k, T_{k-1})$
1 set $J = \{k+1, \dots, n\}$
2 **for** $(B_I, \omega(B_I)) \in \mathit{values}(T_{k-1})$ **do**
3 **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`
4 set $I' = I \cup \{k\}$
5 compute the orbit $\mathcal{O} = G_k \cdot I'$
6 **if** $\mathcal{O} \in \mathit{keys}(T_k)$ **then**
7 increment the multiplicity of $T_k[\mathcal{O}]$ by $\omega(B_I)$
8 **else**
9 $T_k[\mathcal{O}] = ((H_{I'}, H_J), \omega(B_I))$
`// produce the deletion as the left child`
10 compute the orbit $\mathcal{O} = G_k \cdot I$
11 **if** $\mathcal{O} \in \mathit{keys}(T_k)$ **then**
12 increment the multiplicity of $T_k(\mathcal{O})$ by $\omega(B_I)$
13 **else**
14 $T_k[\mathcal{O}] = ((H_I, H_J), \omega(B_I))$
return T_k

Example 10. The structure underlying Algorithm 3 applied to the arrangement in Example 2 is shown in Figure 5. It is no longer a tree but may be obtained from the tree in Figure 4 by identifying nodes under the stabilizers of $\mathrm{Aut}(\mathcal{A})$. Each identification accumulates multiplicity in the node and that multiplicity is passed down to its children.

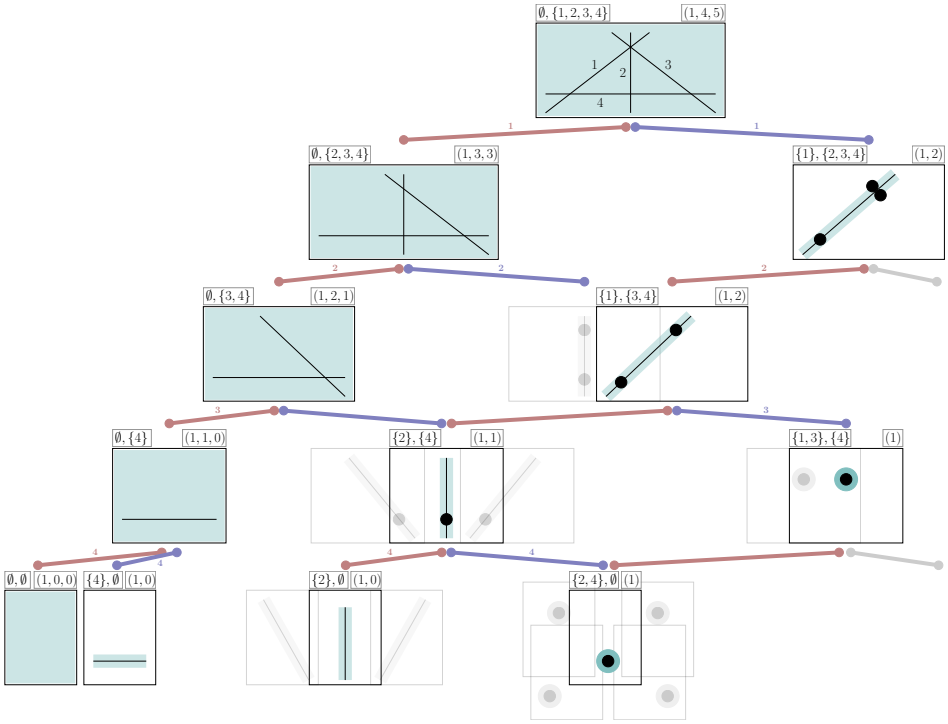


FIGURE 5. The algorithmic structure underlying Algorithm 3. Starting at the top node, each call of Algorithm 4 produces the next depth of this graph.

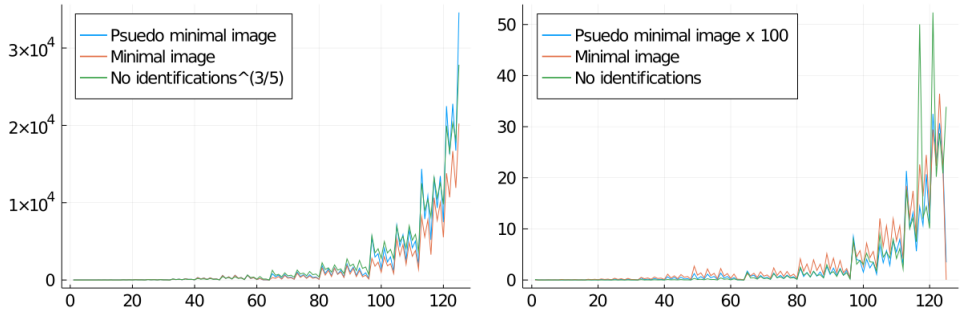
5.1. **Representing orbits.** The computations of orbits in Line 5 and Line 10 require elaboration; specifically in regards to representing an orbit $G \cdot I$ on a computer. One option is to use a canonical element of $G \cdot I$, which can be computed using the `MinimalImage` or `CanonicalImage` functions from `GAP` [21, 22]. An alternative approach is to provide any function $\varphi : 2^{[n]} \to S$ 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 $g_i I < I$ 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 11. 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 14) 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 Figure 6. Note that in the first figure, the number of leaves for "no identifications" is raised to the power 3/5 as to make the plots visually comparable. Similarly, the timing for `pseudo_minimal_image` is scaled by a factor of 100 in the second figure.

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.



(A) Leaves at depth. (B) Time per depth (in seconds).

FIGURE 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. The data is scaled to be comparable as indicated in the legend.

5.2. **Accumulating the Betti 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 Betti 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` [15], `Singular` [12], `Polymake` [16, 25], and `Antic (Hecke, Nemo)` [37]. Our software is written in `julia` and builds heavily on these cornerstones. Specifically, we use the number theory components `Nemo` [14] 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` [15] for group theoretic computations in Algorithm 4. Concretely, we compute stabilizers and minimal images using the `GAP` packages `ferret` [20] and `images` [22], 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. 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> betti_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> betti_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 3 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 $\mathbf{V}$.

A hyperplane H_v partitions the points in $(\mathbb{R}^{d+1})^* \backslash H_v$ 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 $V_2 = \mathbf{V} \backslash V_1$. 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, Section 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 [33, 34, 46] and algebraic statistics [11].

Definition 12. The threshold arrangement[2], $\mathcal{T}_d$ is the separability arrangement associated to the vertices of the hypercube $[0, 1]^d$. That is,

$$\mathcal{T}_d := \{\{x_0 + c_1 x_1 + \cdots + c_d x_d = 0\} \text{ with } c_i \in \{0, 1\} \text{ for all } c_i\}.$$

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 Betti numbers of $\mathcal{T}_d$ for $1 \leq d \leq 8$, and thus their number of chambers. The results are collected in Table 4 and the timings appear in Table 2. The values of $|\operatorname{ch}(\mathcal{T}_d)|$ for $1 \leq d \leq 9$ are listed in entry A000609 of the Online-Encyclopedia of Integer Sequences (OEIS), whereas the Betti numbers of $\mathcal{T}_d$, to the best of our knowledge, have not been published before. Zuev showed that asymptotically $|\operatorname{ch}(\mathcal{T}_d)| \sim 2^{d^2}$ [45].

Remark 13. Using similar proof techniques as in [30] 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 4 we obtain $b_2(\mathcal{T}_d) = \frac{1}{2}(4^d - 2^d)$ and $b_3(\mathcal{T}_d) = \frac{1}{24}\left(4 \cdot 8^d - 3 \cdot 6^d - 6 \cdot 4^d + 5 \cdot 2^d\right)$. 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 14. 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 $\mathcal{R}_d := \{\{c_1 x_1 + c_2 x_2 + \cdots + c_d x_d = 0\}$ with $c_i \in \{0, 1\}$ and not all c_i are zero$\}$.

[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 12.

The chambers of the resonance arrangements are in bijection with generalized retarded functions in quantum field theory [13]. An overview of the applications of the resonance arrangement is given in [30, Section 1]. A formula for their number of chambers remains elusive, let alone one for their Betti numbers. Nonetheless, partial formulas and bounds exist [4, 17, 30, 45].

The numbers of chambers of the resonance arrangements are listed in the sequence A034997 in the OEIS up to $d = 9$. The Betti numbers are published in [26] up to $d = 7$. Our software was able to determine the Betti numbers of $\mathcal{R}_8$ and $\mathcal{R}_9$. The computation for $\mathcal{R}_9$ took ten days, running multithreaded on 42 `Intel Xeon E7-8867 v3` CPUs. All Betti numbers of $\mathcal{R}_d$ up to $d = 9$ are given in Table 5 and the timings are listed in Table 2.

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 $|\operatorname{ch}(\mathcal{C}_d)| = 2 \cdot 3^d - 2^d$ for $d \leq 20$, suggesting that $|\operatorname{ch}(\mathcal{C}_d)|$ agrees with this sequence (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 Betti numbers in Table 7 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 6 contains the Betti numbers of $\mathcal{D}_d$ up to $d = 9$.

6.7. **Other separability arrangements.** We give the Betti numbers of the separability arrangements of the vertices of the regular 24-cell, 600-cell, and 120-cell in Table 8. These are regular 4-polytopes, of which the latter two do not admit rational realizations.

6.8. **Discriminantal arrangements.** Given n points in $\mathbb{R}^d$ in general position, the discriminantal arrangement $\operatorname{Disc}_{d,n}$ 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 [9]. We verify the Betti numbers of $\operatorname{Disc}_{4,n}$ for $5 \leq n \leq 16$ given in [29, Section 4.4]. From this data, we recover their formula for the characteristic polynomial of $\operatorname{Disc}_{4,n}$ for all n. A deformation of this arrangement appears in physics [7, 8] 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 mere calculation of Betti numbers as shown below.

Software	$d=3$	4	5	6	7	8
`CountingChambers.jl`	$0.0038s$	$0.011s$	$0.035s$	$0.12s$	$2.89s$	$19.8m$
`polymake`	$0.3s$	$4.31s$	$3.9m$	$*$		
`sage`	$0.05s$	$0.21s$	$5.45s$	$9.2m$	$*$	
`GAP`	$0,006s$	$0.035s$	$1.09s$	$1.9m$	$12.87h$	$*$

TABLE 1. Timings for computing the number of chambers $|\operatorname{ch}(\mathcal{R}_d)|$ of the resonance arrangement $\mathcal{R}_d$ for $3 \leq d \leq 8$ on a single thread (`Intel Core i7-8700`). An asterisk * indicates that the computation was terminated after a day.

The implementation [27] in `polymake` computes much more information than the Betti 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` [31] 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 Section 6, we collect our timings in Table 2. 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.

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

TABLE 2. Table of our timings on examples from Section 6. All computations ran on a single thread (`Intel Core i7-8700`) except for $\mathcal{R}_9$ which ran on 42 threads (`Intel Xeon E7-8867`).

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

TABLE 3. Comparison of the effect of number of threads on run times (`Intel Core i7-8700`).

Appendix A. Tables of Betti numbers

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
$\lvert\operatorname{ch}(\mathcal{T}_d)\rvert$	2	14	104	1882	94572	15028134	8378070864	17561539552946

Table 4. The values of $b_i(\mathcal{T}_d)$ and $\lvert\operatorname{ch}(\mathcal{T}_d)\rvert$ of the threshold arrangement for $1 \leq d \leq 9$ and $0 \leq i \leq d$.

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
$\lvert\operatorname{ch}(\mathcal{R}_d)\rvert$	2	6	32	370	11292	1066044	347326352	419172756930	1955230985997140

Table 5. The values of $b_i(\mathcal{R}_d)$ and $\lvert\operatorname{ch}(\mathcal{R}_d)\rvert$ of the resonance arrangement for $1 \leq d \leq 9$ and $0 \leq i \leq d$.

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
$\lvert\operatorname{ch}(\mathcal{D}_d)\rvert$	4	16	146	3756	291558	74656464	74904015666	297363155783764

TABLE 6. The values of $b_i(\mathcal{D}_d)$ and $\lvert\operatorname{ch}(\mathcal{D}_d)\rvert$ of the demicube arrangement for $2 \leq d \leq 9$ and $0 \leq i \leq d+1$.

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
$\lvert\operatorname{ch}(\mathcal{P}_d)\rvert$	2	4	32	2894	8595502	1669309192292

TABLE 7. The values of $b_i(\mathcal{P}_d)$ and $\lvert\operatorname{ch}(\mathcal{P}_d)\rvert$ of the permutohedron arrangement for $1 \leq d \leq 6$ and $0 \leq i \leq d$.

polytope	24-cell	600-cell	120-cell
$b_0(\mathcal{A})$	1	1	1
$b_1(\mathcal{A}) = \lvert\mathcal{A}\rvert$	24	120	600
$b_2(\mathcal{A})$	276	7140	179700
$b_3(\mathcal{A}$	1630	225782	31972550
$b_4(\mathcal{A})$	4308	3118740	2979870540
$b_5(\mathcal{A})$	2931	2899979	2948077091
$\lvert\operatorname{ch}(\mathcal{A})\rvert$	9170	6251762	5960100482

TABLE 8. The values of $b_i(\mathcal{A})$ and $\lvert\operatorname{ch}(\mathcal{A})\rvert$ of the arrangements stemming from regular 4-polytopes for and $0 \leq i \leq 5$.

References

[1] C. A. Athanasiadis. Characteristic polynomials of subspace arrangements and finite fields. *Adv. Math.*, 122(2):193–233, 1996.

[2] P. Baldi and R. Vershynin. Polynomial threshold functions, hyperplane arrangements, and random tensors. *SIAM Journal on Mathematics of Data Science*, 1(4):699–729, 2019.

[3] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: a fresh approach to numerical computing. *SIAM Rev.*, 59(1):65–98, 2017.

[4] L. J. Billera, J. T. Moore, C. D. Moraites, Y. Wang, and K. Williams. Maximal unbalanced families. *arXiv preprint arXiv:1209.2309*, 2012.

[5] D. Bremner, M. Dutour Sikirić, D. V. Pasechnik, T. Rehn, and A. Schürmann. Computing symmetry groups of polyhedra. *LMS J. Comput. Math.*, 17(1):565–581, 2014.

[6] D. Bremner, M. Dutour Sikirić, and A. Schürmann. Polyhedral representation conversion up to symmetries. In *Polyhedral computation*, volume 48 of *CRM Proc. Lecture Notes*, pages 45–71. Amer. Math. Soc., Providence, RI, 2009.

[7] F. Cachazo, N. Early, A. Guevara, and S. Mizera. Scattering equations: from projective spaces to tropical grassmannians. *Journal of High Energy Physics*, 2019(6):39, 2019.

[8] F. Cachazo, B. Umbert, and Y. Zhang. Singular solutions in soft limits. *Journal of High Energy Physics*, 2020(5):148, 2020.

[9] H. Crapo. The combinatorial theory of structures. In *Matroid theory (Szeged, 1982)*, volume 40 of *Colloq. Math. Soc. János Bolyai*, pages 107–213. North-Holland, Amsterdam, 1985.

[10] H. H. Crapo and G.-C. Rota. *On the foundations of combinatorial theory: Combinatorial geometries*. The M.I.T. Press, Cambridge, Mass.-London, preliminary edition, 1970.

[11] M. A. Cueto, J. Morton, and B. Sturmfels. Geometry of the restricted Boltzmann machine. In *Algebraic methods in statistics and probability II*, volume 516 of *Contemp. Math.*, pages 135–153. Amer. Math. Soc., Providence, RI, 2010.

[12] W. Decker, G.-M. Greuel, G. Pfister, and H. Schönemann. Singular 4-2-0 — A computer algebra system for polynomial computations. `http://www.singular.uni-kl.de`, 2020.

[13] T. Evans. *What is being calculated with Thermal Field Theory?*, pages 343–352. World Scientific, 1995.

[14] C. Fieker, W. Hart, T. Hofmann, and F. Johansson. Nemo/Hecke: Computer algebra and number theory packages for the julia programming language. In *Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation*, ISSAC '17, pages 157–164, New York, NY, USA, 2017. ACM.

[15] GAP – Groups, Algorithms, and Programming, Version 4.10.2. `https://www.gap-system.org`, Jun 2019.

[16] E. Gawrilow and M. Joswig. *polymake: a Framework for Analyzing Convex Polytopes*, pages 43–73. Birkhäuser Basel, Basel, 2000.

[17] S. C. Gutekunst, K. Mészáros, and T. K. Petersen. Root cones and the resonance arrangement. *Electron. J. Combin.*, 28(1):Paper No. 1.12, 39, 2021.

[18] D. Halperin and M. Sharir. Arrangements. In *Handbook of discrete and computational geometry, 3rd edition*, pages 49–119. CRC Press, 2017.

[19] J. Huh and E. Katz. Log-concavity of characteristic polynomials and the Bergman fan of matroids. *Math. Ann.*, 354(3):1103–1116, 2012.

[20] C. Jefferson. ferret, backtrack search in permutation groups, Version 1.0.2. `https://gap-packages.github.io/ferret/`, Jan 2019. GAP package.

[21] C. Jefferson, E. Jonauskyte, M. Pfeiffer, and R. Waldecker. Minimal and canonical images. *J. Algebra*, 521:481–506, 2019.

[22] C. Jefferson, M. Pfeiffer, R. Waldecker, and E. Jonauskyte. images, minimal and canonical images, Version 1.3.0. `https://gap-packages.github.io/images/`, Mar 2019. GAP package.

[23] A. N. Jensen. Traversing symmetric polyhedral fans. In K. Fukuda, J. v. d. Hoeven, M. Joswig, and N. Takayama, editors, *Mathematical Software – ICMS 2010*, pages 282–294, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[24] C. Jordan, M. Joswig, and L. Kastner. Parallel enumeration of triangulations. *Electron. J. Combin.*, 25(3):Paper No. 3.6, 27, 2018.

[25] M. Kaluba, B. Lorenz, and S. Timme. Polymake.jl: A new interface to polymake. In A. M. Bigatti, J. Carette, J. H. Davenport, M. Joswig, and T. de Wolff, editors, *Mathematical Software – ICMS 2020*, pages 377–385, Cham, 2020. Springer International Publishing.

[26] H. Kamiya, A. Takemura, and H. Terao. Ranking patterns of unfolding models of codimension one. *Adv. in Appl. Math.*, 47(2):379–400, 2011.

[27] L. Kastner and M. Panizzut. Hyperplane arrangements in `polymake`. In A. M. Bigatti, J. Carette, J. H. Davenport, M. Joswig, and T. de Wolff, editors, *Mathematical software – ICMS 2020*, volume 12097 of *Lecture Notes in Computer Science*, pages 232–240. Springer, 2020.

[28] C. J. Klivans and E. Swartz. Projection volumes of hyperplane arrangements. *Discrete Comput. Geom.*, 46(3):417–426, 2011.

[29] H. Koizumi, Y. Numata, and A. Takemura. On intersection lattices of hyperplane arrangements generated by generic points. *Ann. Comb.*, 16(4):789–813, 2012.

[30] L. Kühne. The universality of the resonance arrangement and its Betti numbers. *arXiv preprint arXiv:2008.10553*, 2020.

[31] M. Leuner. alcove. https://github.com/martin-leuner/alcovel, 2019.

[32] T. Möller and G. Röhrle. Counting chambers in restricted Coxeter arrangements. *Arch. Math. (Basel)*, 112(4):347–359, 2019.

[33] G. Montúfar, N. Ay, and K. Ghazi-Zahedi. Geometry and expressive power of conditional restricted boltzmann machines. *Journal of Machine Learning Research*, 16(73):2405–2436, 2015.

[34] G. Montúfar and J. Morton. When does a mixture of products contain a product of mixtures? 2013. 1st International Conference on Learning Representations, ICLR 2013 ; Conference date: 02-05-2013 Through 04-05-2013.

[35] P. Orlik and L. Solomon. Combinatorics and topology of complements of hyperplanes. *Invent. Math.*, 56(2):167–189, 1980.

[36] P. Orlik and H. Terao. *Arrangements of hyperplanes*, volume 300 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 1992.

[37] OSCAR Computer Algebra System, Version 0.5.2. `https://oscar.computeralgebra.de`, April 2021.

[38] R. Pendavingh and J. van der Pol. Asymptotics of symmetry in matroids. *J. Combin. Theory Ser. B*, 135:349–365, 2019.

[39] A. Postnikov and R. P. Stanley. Deformations of Coxeter hyperplane arrangements. volume 91, pages 544–597. 2000. In memory of Gian-Carlo Rota.

[40] J. Sarnoff. SaferInteger, julia package, version 2.5.3. https://github.com/JeffreySarnoff/SaferIntegers.jl, 2021.

[41] N. H. Sleumer. Output-sensitive cell enumeration in hyperplane arrangements. volume 6, pages 137–147. 1999. 6th Scandinavian Workshop on Algorithm Theory (SWAT '98) (Stockholm, 1998).

[42] L. Solomon and H. Terao. A formula for the characteristic polynomial of an arrangement. *Adv. in Math.*, 64(3):305–325, 1987.

[43] M. Yoshinaga. Hyperplane arrangements and Lefschetz's hyperplane section theorem. *Kodai Math. J.*, 30(2):157–194, 2007.

[44] T. Zaslavsky. Facing up to arrangements: face-count formulas for partitions of space by hyperplanes. *Mem. Amer. Math. Soc.*, 1(issue 1, 154):vii+102, 1975.

[45] Y. A. Zuev. Methods of geometry and probabilistic combinatorics in threshold logic. *Discrete Mathematics and Applications*, 2(4):427 – 438, 1992.

[46] J. Zunic. On encoding and enumerating threshold functions. *IEEE Transactions on Neural Networks*, 15(2):261–267, 2004.

[1] Department of Applied and Computational Mathematics and Statistics, University of Notre Dame
Email address: tbrysiew@nd.edu

[2] Technische Universität Berlin, Chair of Discrete Mathematics/Geometry
Email address: eble@math.tu-berlin.de

[3] Max Planck Institute for Mathematics in the Sciences, Leipzig, Germany
Email address: lukas.kuhne@mis.mpg.de