Computing Hereditary Convex Structures

Bernard Chazelle · Wolfgang Mulzer

Abstract Color red and blue the n vertices of a convex polytope $\mathcal{P}$ in $\mathbb{R}^3$. Can we compute the convex hull of each color class in $o(n \log n)$ time? What if we have more than two colors? What if the colors are random? Consider an arbitrary query halfspace and call the vertices of $\mathcal{P}$ inside it blue: can the convex hull of the blue points be computed in time linear in their number? More generally, can we quickly compute the blue hull without looking at the whole polytope? This paper considers several instances of *hereditary* computation and provides new results for them. In particular, we resolve an eight-year old open problem by showing how to split a convex polytope in linear expected time.

Keywords Convex polytope · Halfspace range searching · Hereditary convex hulls

1 Introduction

Given a set of n points in the Euclidean plane and its Voronoi diagram, it was shown in [13] how to compute the Voronoi diagram of any given subset in linear time.[1] The authors asked whether it is also possible to compute the convex hull of an arbitrary

---

[1] All our algorithms are randomized, so the complexity is to be understood in the expected sense. Note that the expectation is only over the randomness used by the algorithm and that the complexity bounds hold for every input, except when explicitly stated otherwise.

B. Chazelle
Department of Computer Science, Princeton University, 35 Olden Street, Princeton, NJ 08540, USA
e-mail: chazelle@cs.princeton.edu

W. Mulzer (✉)
Institut für Informatik, Freie Universität Berlin, Takustraße 9, 14195 Berlin, Germany
e-mail: mulzer@inf.fuberlin.de

subset of the vertices of a convex 3-polytope in linear time. An affirmative answer would, of course, imply the previous result. This paper proves that it is indeed the case. We formulate the question in a *hereditary* setting by assuming that the vertices of a convex polytope $\mathcal{P}$ in $\mathbb{R}^3$ are colored red and blue. The problem is then to "split" $\mathcal{P}$ and compute both monochromatic convex hulls; see Fig. 1. We show how to do this in linear time, which answers the main open question in [13]. We extend our techniques to an arbitrary number of colors by showing how to compute the convex hulls of all the color classes in $O(n\sqrt{\log n})$ time. In subsequent work, Kevin Buchin and the second author improved this bound to $O(n(\log \log n)^2)$ time [6]. Interestingly, we can do this in linear time for any set of χ colors, as long as the coloring is random; the result holds for any $1 \leq \chi \leq n$. We also consider the coloring induced by halfspace range queries: given a query plane, compute the convex hull of the points lying on one side. We show how to do so in time $O(\log n + k)$, where k is the output size; the data structure requires $O(n \log n)$ storage.

Our offline splitting algorithm cannot be output-sensitive, since the output size is linear. But what if we output only one color class? If the chosen vertices form k connected components in the skeleton graph of $\mathcal{P}$, we can compute their convex hull in time $O(n \log^* n + k \log k)$, where n now is the size of the subset. Our result has this intriguing corollary: given a Delaunay triangulation (DT) denoted by $\mathcal{T}$, the DT of any set S of n vertices and edges in $\mathcal{T}$ can be computed in time $O(n \log^* n + k \log k)$, where k is the number of connected components formed by S within $\mathcal{T}$. We actually prove a slightly more general result. It is well known that the convex hull of two convex polytopes can be stitched together in linear time [11]. We consider the case of k disjoint convex polytopes with a total of n vertices. If the vertices of each polytope form a connected component in the convex hull of their union, we can compute their common convex hull in $O(n \log^* n + k \log k)$ time. This assumption is motivated by a lower bound of $\Omega(n \log k)$ for the general case.

To study the complexity of hereditary computing is part of a broader attempt to understand *what makes what hard*. To compute the DT of n points in the plane requires $\Omega(n \log n)$ time, but knowing that the points are the vertices of a convex polygon cuts down the complexity to linear [1, 14]. Given a spanning subgraph of degree at most d, the DT can be completed in time $O(nd \log^* n)$ [21]. In fact, at the cost of a more complicated algorithm, it can be done in linear time [17, 27]. Furthermore, Djidjev and Lingas have proven linearity for any set of points forming a monotone chain in both x and y directions [22]. This might suggest that the hardness of DT is really confined to sorting, but the situation is more complicated. Sorting helps to find Voronoi diagrams in ℓ_∞ [15], but ranking the points in any one direction still leaves us with a $\Theta(n \log n)$ complexity [22]. If we know the order in x and y direction, the DT can be found with $O(n)$ *comparisons*, but no $o(n \log n)$ algorithm is known [6]. More generally, the simplicity of a polygon is known to "linearize" many problems that otherwise exhibit $\Omega(n \log n)$ lower bounds, e.g., polygon triangulation [2, 10, 33], medial axis [16], or constrained Delaunay triangulation [17, 27]. Our work fits into that mold.

Hereditary algorithms are nothing new. Given a subset of a simple polygon, Chan [8] showed how to compute its convex hull in linear time[2] and how to trian-

---

[2]Here, *linear time* means linear in the size of the whole structure, not just the subset.
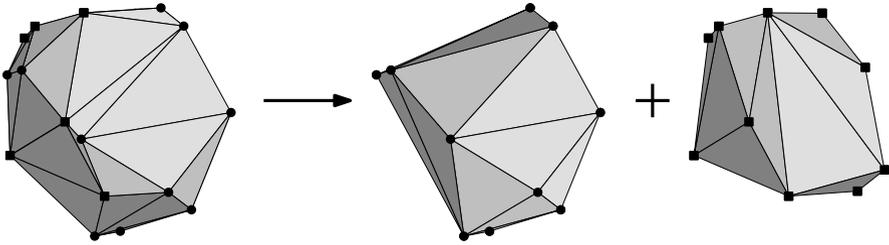
Fig. 1 Given their joint convex hull, we can find the red and blue hulls in linear time

Fig. 2 General hereditary trapezoidal decompositions are hard
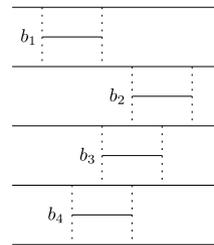
gulate it in $O(n \log^* n)$ time. Van Kreveld, Löffler, and Mitchell [28] improved the latter result by proving that any subset of a given triangulation can in fact be triangulated in linear time. To appreciate the difficulty of obtaining general hereditary algorithms, let us mention the example of hereditary trapezoidal decompositions [8, 26]. Kirkpatrick, Klawe, and Tarjan [26] gave an algorithm for removing a *hole-free* subset of line segments in a trapezoidal decomposition in linear time, where hole-freeness is a property that is necessary to ensure that the subset does not obscure too much information. They also give an example that for general hereditary trapezoidal decompositions, no improvement is possible (see also [8]). Consider the line segments in Fig. 2 and their trapezoidal decomposition. Suppose that we would like to find the trapezoidal decomposition of b_1, b_2, b_3, b_4. To achieve this, we essentially have no choice but to sort their endpoints from scratch, since the long line segments obscure all information. This means that trapezoidal decompositions, unlike convex hulls, do not always give away something about their subsets. There are many other situations in which additional "hereditary" information brings no benefits: if P is a point set in $\mathbb{R}^3$, sorting P in a bounded number of directions does not help in computing its convex hull [32]; nor does knowing the convex hull of P help in finding its diameter [25].

Another way to look at our first result, the linear complexity of bicolored convex hulls, is that the convex hull problem in 3D loses its $\Omega(n \log n)$-hardness if it is embedded in a larger polytope: in other words, computationally speaking, a convex polytope "gives away" the convex hull of any of its subsets.

2 Definitions and Notation

Given a finite point set $P \subseteq \mathbb{R}^3$, let conv P denote the convex hull of P. We denote the edges and facets of conv P by $E[P]$ and $F[P]$. For a point $p \in P$, let $\deg_P p$ be the number of edges in $E[P]$ incident to p, the *degree* of p (with respect to P). Throughout, we will assume that convex hulls are given in a standard planar graph representation, e.g., a DCEL [4, Chap. 2.2]. Our point sets will usually be in *general convex position* (gcp), i.e., every three points in P are linearly independent, and $p \notin$ conv $(P \setminus p)$ for every $p \in P$. In particular, conv P is simplicial,[3] and all the points in P are vertices of conv P.

We use classical geometric random sampling [19, 30]; see Appendix A. For this, we quickly review the notion of *conflict sets*. Given a point set $P \subseteq \mathbb{R}^3$, an edge $e \in E[P]$, and a point $p \notin$ conv P, we say that p *can see e* in conv P or that e *is visible* from p if the triangle spanned by e and p intersects conv P only in e. All the planes we consider are *oriented*, that is, one of the two halfspaces defined by a plane h is designated the *left halfspace* of h, h^+, and the other one is designated the *right halfspace* of h, h^-. We use the convention that every supporting plane of conv P is oriented such that P lies in the right halfspace h^-. Let $Q \subseteq P$, $f \in F[Q]$, and h_f be the supporting plane for f. A point $p \in P$ is *in conflict* with f if p lies in h_f^+. Let $B_f \subseteq P$ denote the points in conflict with f, and b_f the size of B_f. Conversely, for a point $p \in P$, we let $D_p \subseteq F[Q]$ denote the set of facets in conflict with p, and let d_p be its size. The sets B_f and D_p are the *conflict sets* of f and p, and b_f and d_p are the *conflict sizes*. By double counting,

$$\sum_{f \in F[Q]} b_f = \sum_{p \in P} d_p. \tag{1}$$

3 Splitting Polytopes

We are given an n-point set $P \subseteq \mathbb{R}^3$ in general convex position (gcp).[4] Let $B \subseteq P$, and let $R = P \setminus B$. The points in B are called *blue*, the points in R are called *red*. Given conv P, we show how to obtain conv B in linear time.

Theorem 3.1 *Let $P \subseteq \mathbb{R}^3$ be a set of n points in general convex position, colored red and blue. Given* conv P, *the convex hull of the blue points can be computed in $O(n)$ expected time.*

An edge of conv P is called *blue* if both of its endpoints are blue, and *red* if both of its endpoints are red, otherwise it is *bichromatic*. Blue, red, and bichromatic facets are defined similarly. The splitting is performed by a recursive algorithm SplitHull that receives the convex hull and a two-coloring of P. Please refer to Algorithm 1.

---

[3]That is, all facets of conv P are triangles.

[4]See Sect. 2 for basic definitions and notation.

Algorithm 1 Splitting a bichromatic convex hull

SplitHull(conv P)

1. If P contains no red points, return conv P.
2. If there exists a red point r in P with $\deg_P r \leq d_0$ (with a suitable constant d_0), then return SplitHull(conv $(P \setminus r)$).
3. Take random blue points $b \in B$ until (i) $\deg_P b \leq 6$, and (ii) there exists a blue edge e in conv $(P \setminus b)$ that is visible from b.
4. Call SplitHull(conv $(P \setminus b)$) to compute conv $(B \setminus b)$.
5. Using e as a starting edge, insert b into conv $(B \setminus b)$ and return conv B.

SplitHull can be seen as a generalization of Chew's algorithm for Voronoi diagrams of convex polygons [14], and it is also reminiscent of Dobkin and Kirkpatrick's hierarchy [23, 24]. It first tries to delete a red point of small degree. If this is not possible, it removes blue points until there is a red point of small degree again. Later, these blue points must be reinserted into the recursively computed blue hull. In order to do this efficiently, we must be careful about which blue points we delete, so that we have a landmark from where to start the conflict location. SplitHull is easily shown to be correct.

Lemma 3.2 SplitHull(conv P) *computes* conv B.

Proof The proof is by straightforward induction on $|P|$. We only comment on Step 5. Let $B^- = B \setminus b$ and $P^- = P \setminus b$. If e is a blue edge visible from b in conv P^-, then the same holds in conv B^-: since e has both endpoints in B^-, a supporting plane for e in conv P^- supports e also in conv B^-, and since conv $B^- \subseteq$ conv P^-, the triangle spanned by b and e intersects conv B^- only in e. Thus, we can walk from e to determine b's conflict set D_b and replace D_b by new facets incident to b. This takes time $O(|D_b|)$ [4, Chap. 11.2]. When implementing the algorithm, care must be taken that the pointer to e obtained in Step 3 is not invalidated by the recursive call in Step 4. We can easily do it as follows: when deleting a blue edge in Step 4, retain the corresponding record in memory and reuse it when the edge is recreated in Step 5. □

The bulk of the analysis lies in bounding the running time.

Lemma 3.3 *The expected time needed for one invocation of* SplitHull *is constant*, *not counting the time for the recursive calls*.

Proof We argue that each step takes constant expected time. This clearly holds for Step 1: just use a counter for the number of red points. Step 2 is also easy: keep a linked list L for the red points with degree at most d_0. During preprocessing, determine the degrees and initialize L accordingly. When the hull is altered in Steps 2 and 4, update the degrees and L. Since all relevant vertices have bounded degree, this takes constant time. The most interesting part lies in the analysis of Step 3. We show that there is a good chance of sampling a point with the required properties.

Lemma 3.4 *Let $\widetilde{B}$ be the subset of the blue points b with the following properties*: *(i)* $\deg_P b \le 6$; *and (ii)* b *is a vertex of a blue facet of* conv P, *or* $E[P \setminus b] \setminus E[P]$ *contains at least one blue edge.[5] There exists a constant d_0 such that if all red points have degree at least d_0, then $|\widetilde{B}| \ge |P|/5$.*

Proof Call a blue point *pleasant* if it satisfies the properties in the lemma, and *ghastly* otherwise. By Euler's formula, a large fraction of blue points has degree at most 6. If a blue point b is ghastly and has degree at most 6, then either (a) b is incident to a facet with a red edge, or (b) b's neighborhood has only bichromatic edges and to delete b from conv P creates no blue edge. We bound the number of points satisfying (a) and (b) separately and then finish the analysis with a union bound.

In the following, we will assume that d_0 is a large enough constant. By general convex position, we have $|E[P]| = 3n - 6$.[6] Let B' be the set of blue points b with $\deg_P b \le 6$. Since conv P is three-connected [29, Theorem 5.3.3], and since all red nodes have degree at least $d_0 \ge 7$, we get

$$6n - 12 = \sum_{p \in B'} \deg p + \sum_{p \in P \setminus B'} \deg p \ge 3|B'| + 7(n - |B'|).$$

Thus,

$$|B'| > n/4. \tag{2}$$

Similarly,

$$6n - 12 = \sum_{p \in R} \deg p + \sum_{p \in P \setminus R} \deg p \ge d_0|R| + 3(n - |R|) = (d_0 - 3)|R| + 3n,$$

so $|R| < 4n/d_0$ (for $d_0 \ge 12$). Let E_R denote the set of red edges in conv P. Since every red edge of conv P is an edge of conv R,

$$|E_R| \le |E[R]| = 3|R| - 6 < 12n/d_0. \tag{3}$$

For $b \in B'$, let Γ_b be the simple polygon formed by b's neighbors in conv P, and let C be the set of points $b \in B'$ such that Γ_b contains a red edge (this corresponds to property (a) mentioned at the beginning of the proof). Since an edge is incident to two facets, for each $e \in E_R$, there are at most two points $p, q \in C$ such that e is in Γ_p and Γ_q. Hence, by (3),

$$|C| \le 2|E_R| < 24n/d_0. \tag{4}$$

Now, let $D \subseteq B'$ be the set of points b such that Γ_b has no monochromatic edge. For any such b, $\deg_P b$ is even, and red and blue points alternate along Γ_b. Let $E_b = E[P \setminus b] \setminus E[P]$. We say that b *creates* E_b. Note that E_b contains only diagonals of Γ_b. Any edge e is created by at most two points in D: if e is occluded in conv P by

---

[5] Recall that $E[P]$, $F[P]$ denote the edges and facets of conv P (see Sect. 2).

[6] Since all the points are on the hull, Euler's formula [5, Theorem 7.2.1] yields $n - |E[P]| + |F[P]| = 2$, and since all facets are triangles, we have $2|[E[P]| = 3|F[P]|$.

Fig. 3 (a) The edge e_1 is occluded by exactly one edge and is created by u and v; (b) the edge e_2 is occluded by two edges and is created only by w

(a)   (b)
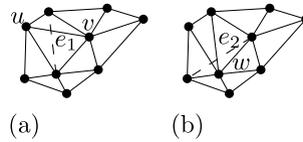
Fig. 4 Every triangulation of a two-colored simple polygon contains at least one monochromatic diagonal (shown in *dashed*)
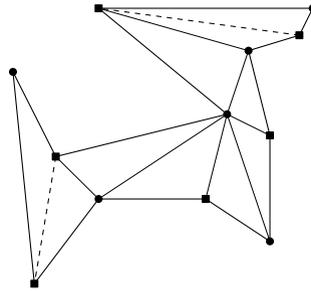
exactly one edge, it is created by the endpoints of this edge; if e is occluded by two or more edges, it can only be created by a point incident to all of them; see Fig. 3.

Furthermore, every $b \in D$ creates at least one monochromatic edge, since every triangulation of a two-colored simple polygon contains at least one monochromatic diagonal;[7] see Fig. 4. Let D' be the set of points in D that do not create a blue edge (these are the points with property (b)). By the previous discussion and (3),

$$|D'| \le 2|E[R]| < 24n/d_0. \tag{5}$$

To conclude, we observe that all the points in the set $B' \setminus (C \cup D')$ are pleasant and that by (2, 4, 5) it contains at least $(1/4 - 48/d_0)n > n/5$ points, for d_0 large enough.   □

By Lemma 3.4 we expect at most five iterations in Step 3, each taking constant time, since all points under consideration have bounded degree. The same holds for Step 4 without the recursive call, as $\deg_P b \le 6$. Finally, we use backwards analysis to handle Step 5. Take $\widetilde{B}$ as in Lemma 3.4. Because $|\widetilde{B}| > |B|/5$, the average degree of a point in $\widetilde{B}$ is less than 30, by Euler's formula. Hence, to delete a random point $b \in \widetilde{B}$ from conv B takes constant expected time, and this is exactly the cost of inserting b into conv $(B \setminus b)$ [4, Chap. 11.2].   □

Theorem 3.1 follows from Lemmas 3.2 and 3.3, since the number of recursive calls is $O(n)$.

---

[7] Since the dual graph of this triangulation is a tree [4, Sect. 3.1], and every tree contains at least one leaf, corresponding to a triangle between two adjacent edges.
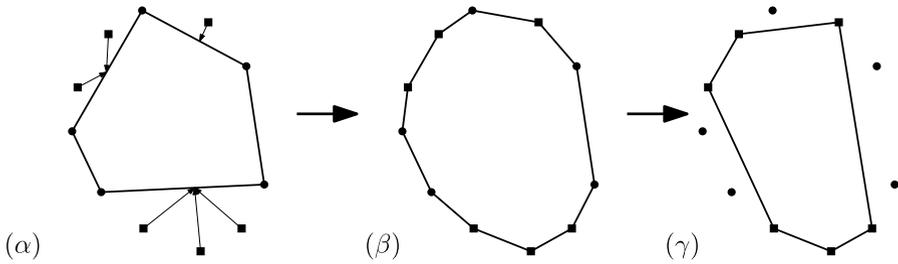
Fig. 5 Splitting random colorings: the algorithm (α) computes conv S and conflict facets for C_i, (β) inserts C_i into conv S, and (γ) extracts conv C_i. The points in C_i are shown as boxes, S as circles

---

Algorithm 2 Splitting random colorings

RandMultiSplit(conv P) (* see Fig. 5 *)

1. Pick a random sample $S \subseteq P$ of size n/χ and compute conv S.
2. For each $p \in P$, determine a facet $f_p \in F[S]$ in conflict with p.
3. For each color i:
    (a) Insert all points of C_i into conv S.
    (b) Extract conv C_i from conv $(C_i \cup S)$.

---

4 Splitting Random Colorings

Now, we extend SplitHull to handle more than two colors: for a point set $P \subseteq \mathbb{R}^3$, let $c : P \to \{1, \ldots, \chi\}$ be a *coloring* of P. For $i \in \{1, \ldots, \chi\}$, we let $C_i = c^{-1}(i)$ denote the points that are colored i, the ith *color class*. The coloring c is called *random* if each point p is colored uniformly and independently with a color in $\{1, \ldots, \chi\}$. This section deals with random colorings; and the next one is about arbitrary colorings.

Theorem 4.1 *Let* $P \subseteq \mathbb{R}^3$ *be a set of n points in general convex position, and let* $c : P \to \{1, \ldots, \chi\}$ *be a random coloring of P. Given* conv P, *we can compute the convex hulls* conv $C_1, \ldots,$ conv C_χ *in* $O(n)$ *expected time (the expectation is over the coloring and the random choices of the algorithm).*

The algorithm for Theorem 4.1 is called RandMultiSplit. See Algorithm 2. It receives the convex hull and a coloring of P as input, and it computes the convex hull of a random sample $S \subseteq P$ into which the points of each color class are then inserted separately. As we will see below, this can be done quickly because c is random. Finally, it uses SplitHull to remove the points from S.

Clearly, the algorithm correctly computes the conv C_i. We bound the running time of each step. Using SplitHull, Step 1 requires $O(n)$ time. The analysis of Step 2 needs more work.

Lemma 4.2 *Step 2 takes* $O(n)$ *expected time.*

Fig. 6 Claim 4.3: the facets
$F[Q]$ are shown *dashed*, $F[P]$
solid. Merge $\Gamma_P(p)$ with $\Gamma_Q(p)$
to determine its conflict facets
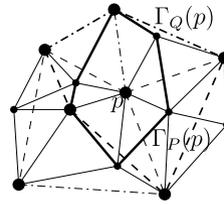
Algorithm 3 Determining the conflict facets in a subset

`SubsetConflictWalk(conv S, conv P)`

1. Let `queue` be a queue with the elements in S.
2. While `queue` $\neq \emptyset$.
   (a) Let p be the next point in `queue`.
   (b) If $p \notin S$, insert p into conv S, using a previously computed conflict facet f_p for p as a starting point.
   (c) For each neighbor $q \in \Gamma_P(p)$, find a conflict facet $\tilde{f}_q$ in conv $(S \cup p)$, using Claim 4.3.
   (d) Using the $\tilde{f}_q$'s, find conflict facets $f_q \in F[S]$ for all $q \in \Gamma_P(p)$. If $q \in \Gamma_P(p)$ has not been encountered yet, insert it into `queue`.

Proof For $Q \subseteq P$ and $p \in Q$, let $\Gamma_Q(p)$ denote the neighbors of p in conv Q. First, we show how to compute the conflict facets for points that are neighbors in conv P of a point in Q.

Claim 4.3 *Let $Q \subseteq P$ and $p \in Q$. Assume that both* conv Q *and* conv P *are available. In time $O(\deg_Q p + \deg_P p)$, we can compute a conflict facet $f_q \in F[Q]$ for every neighbor $q \in \Gamma_P(p)$ of p.*

Proof Consider an *overlay* of conv Q and conv P, i.e., a central projection of their vertices and edges onto the unit sphere centered at a point $O \in$ conv Q. Let $q \in \Gamma_P(p)$, and let $f \in F[Q]$ be the facet incident to p that is intersected by the line segment pq in the overlay. Then q is in conflict with f. To see this, let h_f be the plane supporting f. If q did not conflict with f, then q would lie in h_f^-, and at least part of the line segment pq would be strictly inside conv Q. But then pq could not be an edge of conv P, as conv $Q \subseteq$ conv P. Thus, conflict facets for $\Gamma_P(p)$ can be computed by merging the cyclically ordered lists $\Gamma_P(p)$ and $\Gamma_Q(p)$ with respect to some overlay of the hulls; see Fig. 6. This takes time $O(\deg_Q p + \deg_P p)$.   □

The conflict facets for P can now be found by breadth-first search, using an algorithm called `SubsetConflictWalk`. Please refer to Algorithm 3. Step 2 of the algorithm maintains the invariant that a conflict facet $f_p \in F[S]$ is known for each $p \in$ `queue` $\setminus S$. Using standard techniques, Step 2(b) takes $O(d_p)$ time, where d_p is

the conflict size of p in conv S [4, Chap. 11.2].[8] Furthermore, by Claim 4.3, the conflict facets of $\Gamma_P(p)$ can be found in $O(\deg_{S \cup p} p + \deg_P p)$ time. Finally, Step 2(d) takes time $O(\deg_P p)$: every facet $\tilde{f} \in F[S \cup p]$ shares at least one edge e with an $f \in F[S]$, and if q can see e in conv S, it conflicts with at least one facet adjacent to e. Thus, f_q can be computed from $\tilde{f}_q$ in constant time. It follows that the total running time of `SubsetConflictWalk` is proportional to

$$\mathbf{E}\left[\sum_{p \in P}\left(d_p + \deg_{S \cup p} p + \deg_P p\right)\right].$$

Now, since[9] $\deg_{S \cup p} p \ll d_p$ for $p \notin S$, this is proportional to

$$\mathbf{E}\left[\sum_{p \in S}\deg_S p + \sum_{p \in P \setminus S} d_p + \sum_{p \in P}\deg_P p\right] \ll \mathbf{E}\left[\frac{n}{\chi} + \sum_{f \in F[S]} b_f + n\right],$$

by (1) in Sect. 2. The lemma follows, since $\mathbf{E}[\sum_{f \in F[S]} b_f] \ll n$ by Lemma A.4(8) (b_f is the conflict size of f). $\qquad\square$

Now we consider Step 3 of `RandMultiSplit`. Fix a color i, and for each $f \in F[S]$, let $a_f = |C_i \cap B_f|$. Since the coloring is random, conditioned on b_f, the size a_f is distributed like a sum of independent Bernoulli random variables with mean b_f / χ. By standard moment bounds [12, Lemma A.1], $\mathbf{E}_c[a_f^2] \ll (b_f / \chi)^2$. By Lemma A.5, Step 3(a) takes time $\mathbf{E}_{S,c}[\sum_{f \in F[S]} a_f \log a_f]$, and by Lemma A.4(8), we get

$$\mathbf{E}_{S,c}\left[\sum_{f \in F[S]} a_f \log a_f\right] \ll \mathbf{E}_S\left[\sum_{f \in F[S]} \mathbf{E}_c[a_f^2]\right] \ll \mathbf{E}_S\left[\frac{1}{\chi^2}\sum_{f \in F[S]} b_f^2\right]$$

$$\ll \frac{\chi n}{\chi^2} = \frac{n}{\chi}.$$

Using `SplitHull` in Step 3(b), conv C_i can now be computed in time $O(|C_i| + n/\chi)$. There are χ colors, so Step 3 takes total time proportional to $\sum_i |C_i| + \chi \cdot (n/\chi) \ll n$, and Theorem 4.1 follows.

5 Splitting Arbitrary Colorings

We now consider arbitrary colorings. With `SplitHull` as a black box, we can easily split a χ-colored polytope in time $O(n \log \chi)$. For sufficiently large χ, this can be improved.

Theorem 5.1 *Let $P \subseteq \mathbb{R}^3$ be a set of n points in general convex position, and let $c : P \to \{1, \dots, \chi\}$ be an arbitrary coloring of P. Given conv P, we can compute conv $C_1, \dots,$ conv C_χ in $O(n\sqrt{\log n})$ expected time.*

---

[8]$d_p = 0$ if $p \in S$.

[9]We use the Vinogradov notation $f \ll g$ for $f = O(g)$ and $f \gg g$ for $f = \Omega(g)$.

For the random colorings in the previous section, we could exploit the fact that each color is spread uniformly over the polytope in order to design a simple divide and conquer algorithm that decomposes each color class into subsets of expected constant size. This is no longer possible for arbitrary colorings, because now the distribution of color classes can be highly irregular. Therefore, we need a more sophisticated scheme to partition the color classes. We begin with a useful sampling lemma.

Lemma 5.2 *Let $Q \subseteq \mathbb{R}^3$ be an m-point set in general convex position, and let $\mu \in (0, 1)$ be a constant. There exists an integer α_0 such that the following holds: let $\alpha \in \{\alpha_0, \ldots, \mu m\}$. Given $\operatorname{conv} Q$, in $O(m)$ time we can compute subsets $S, R \subseteq Q$ and a partition $R_1, \ldots, R_\beta$ of R such that*

1. *$|S| = \alpha$, $|R| \gg m$, and $\max_i |R_i| \ll m(\log \alpha)/\alpha$.*
2. *For each R_i, there exists a facet $f_i \in F[S]$ such that all points in R_i are in conflict with f_i.*
3. *Every point in R conflicts with constantly many facets of $\operatorname{conv} S$.*
4. *The conflict sets for two points $p \in R_i$, $q \in R_j$, $i \neq j$, are disjoint and no conflict facet of p shares an edge with a conflict facet of q.*

Furthermore, the convex hulls $\operatorname{conv} S, \operatorname{conv} R_1, \ldots, \operatorname{conv} R_\beta, \operatorname{conv}(Q \setminus (R \cup S))$ can be computed in expected $O(m)$ time.

Proof We call a subset $S \subseteq Q$ *decent* if it has two properties: (i) $\sum_{f \in F[S]} b_f \ll m$; and (ii) $\max_{f \in F[S]} b_f \ll m(\log \alpha)/\alpha$, where b_f denotes the conflict size of f.

Claim 5.3 *A decent subset $S \subseteq Q$ of size α together with $\operatorname{conv} S$ and the conflict sets B_f, $f \in F[S]$, can be found in expected time $O(m)$.*

Proof Let S be a random α-subset of Q. We claim that S is decent with probability at least $1/2$. To see this, we first use Lemma A.4(8) with $\gamma = 1$ to obtain $\mathbf{E}[\sum_{f \in F[S]} b_f] \ll m$. By Markov's inequality, it follows that $\sum_{f \in F[S]} b_f \ll m$ with probability at least $3/4$. Furthermore, using Lemma A.2 with $pn = \alpha$ and $t = 2\log \alpha$, we get[10] $\mathbf{E}[|F_{\geq 2\log \alpha}|] \ll (\log^2 \alpha)/\alpha$, and if α_0 is large enough, this expected value is less than $1/4$. Hence, by Markov's inequality, the probability that there exists a facet with conflict size at least $2m(\log \alpha)/\alpha$ is at most $1/4$. So, we have $\max_{f \in F[S]} b_f \ll m(\log \alpha)/\alpha$ with probability at least $3/4$, and the claimed probability follows from a union bound.

Furthermore, a decent sample can be verified in $O(m)$ time: by the proof of Lemma 4.2, we can find the conflict sets B_f and D_p in time $O(m + \sum_{f \in F[S]} b_f)$. Hence, we can run the algorithm of Lemma 4.2 on the sample S. If the number of steps exceeds cm, for a certain constant c that comes from the proof of Lemma 4.2, we abort the computation and reject the sample, since it cannot be decent. Otherwise, we can check in $O(m)$ time that $\max_{f \in F[S]} b_f \ll m(\log \alpha)/\alpha$, as required. Consequently, since a sample is decent with constant probability, repeated sampling yields the desired result. □

Fig. 7 The pruning step: remove all facets at distance at most 2λ from a facet with maximum conflict size. The points in B'_f, B'_g conflict only with the darker facets at distance at most $\lambda = 1$
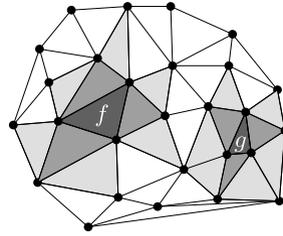
Algorithm 4 Pruning the conflict facets

PruneFS

1. Let $\mathcal{F} = \emptyset$, and let queue be a priority queue containing the facets in $F[S]$.
2. While queue $\neq \emptyset$:
   (a) Let f be a facet in queue with maximum b'_f, and let $N_f = \{f' \in F[S] \mid \delta(f, f') \leq 2\lambda\} \cap$ queue.
   (b) Let queue $=$ queue $\setminus N_f$ and $\mathcal{F} = \mathcal{F} \cup \{f\}$.

Now let S be a decent sample, and let B_f, $f \in F[S]$, denote its conflict sets. By (1) and Property (i) of a decent sample, we have $\sum_{p \in Q} d_p \ll m$, and hence there exists a constant λ such that the set $X = \{p \in Q \mid d_p > \lambda\}$ has cardinality at most $(1 - \mu)m/2$. Let $R' = Q \setminus (S \cup X)$, $B'_f = B_f \cap R'$ and $b'_f = |B'_f|$ for $f \in F[S]$. By definition, all points in R' conflict with at most λ facets. We now prune $F[S]$ to obtain a subset $\mathcal{F}$ of facets whose conflict sets constitute the desired partition. For $f, g \in F[S]$, let $\delta(f, g)$ denote the BFS-distance between f and g in the dual graph of conv S; see Fig. 7.[11] The pruning is done by a greedy algorithm PruneFS, which iteratively takes the facet with the largest conflict size and discards all of its neighbors. For details, see Algorithm 4. Clearly, PruneFS takes $O(m)$ time. Let $f_1, \ldots, f_\beta$ be the facets in $\mathcal{F}$ as computed by PruneFS, and let $R_1, \ldots, R_\beta$ be the corresponding conflict sets with respect to R'. Set $R = \bigcup_{i=1}^{\beta} R_i$.

Claim 5.4 *We have $|R| \gg m$, the R_i constitute a partition of R, and for $p \in R_i, q \in R_j$, $i \neq j$, we have $D_p \cap D_q = \emptyset$, and no facet in D_p shares an edge with a facet in D_q.*

Proof To see that $|R| \gg m$, note that $|N_f| \ll 1$ and $b'_{f'} \leq b'_f$ for every $f' \in N_f$. Thus, we have $b'_f \gg \sum_{f' \in N_f} b'_{f'}$, and therefore

---

[10]Note that by choosing α_0 large enough, we can ensure that $t = 2 \log \alpha \leq \alpha/4 = pn/4$.

[11]More precisely, the *BFS-distance* (BFS = **B**readth **F**irst **S**earch) between f and g is the length of a shortest path between f and g in the graph with vertex set $F[S]$ in which two vertices are adjacent precisely if the corresponding facets share an edge in conv S.

$$|R| = \sum_{f \in \mathcal{F}} b'_f \gg \sum_{f \in \mathcal{F}} \sum_{f' \in N_f} b'_{f'} \geq |R'| \geq m - \alpha - (1 - \mu)m/2$$

$$\geq m - \mu m - (1 - \mu)m/2 = (1 - \mu)m/2 \gg m.$$

To see that $(R_i)_{1 \leq i \leq \beta}$ is a partition, consider two sets R_i, R_j with $i \neq j$, and let f_i, f_j be the corresponding facets. Any point $p \in R$ has $|D_p| \leq \lambda$, and D_p is connected in the dual graph of conv S. By construction, we have $\delta(f_i, f_j) > \lambda$, so there cannot be a point in conflict with both f_i and f_j. It follows that $R_i \cap R_j = \emptyset$, since R_i and R_j are the conflict sets of f_i and f_j. Similarly, we see that D_p, D_q are disjoint for $p \in R_i$, $q \in R_j$, and no facet in D_p is adjacent to a facet in D_q, because $\delta(f_i, f_j) > 2\lambda$ and D_p, D_q are connected with size at most λ.   □

By now, we have established statements 1–4 of Lemma 5.2. It remains to show how to find all the convex hulls quickly. First, using `SplitHull`, we can compute the hulls conv S, conv $(R \cup S)$, and conv $(Q \setminus (R \cup S))$ in time $O(m)$. It remains to consider the R_i's.

Claim 5.5 *For $i = 1, \ldots, \beta$, the convex hull* conv R_i *can be computed in* $O(|R_i|)$ *time.*

Proof Consider an R_i, and let f_i be the corresponding facet in conv S. First, note that the subgraph of conv $(R \cup S)$ induced by R_i is connected, because $R_i = R \cap h_{f_i}^+$. Let Γ denote the points in $(R \cup S) \setminus R_i$ that are adjacent in conv$(R \cup S)$ to a point in R_i. We have $\Gamma \subseteq S$: if there were two points $p \in R_i$, $q \in R_j$, $i \neq j$, such that pq is an edge of conv $(R \cup S)$, then pq would also be an edge of conv $(S \cup \{p, q\})$. This implies that either $D_p \cap D_q \neq \emptyset$ or that there are facets $f' \in D_p$, $f'' \in D_q$ such that f' and f'' share an edge. Both are impossible by Claim 5.4.

Next, we claim that $|\Gamma| \ll 1$: if $p \in R_i$ is adjacent to a point $q \in S$, then it follows that pq is also an edge of conv $(S \cup \{p\})$, and hence D_p contains a facet incident to q. Since $|\bigcup_{p \in R_i} D_p| \ll 1$ and since each facet is incident to three points, the claim follows.

Now we compute conv $(R_i \cup \Gamma)$ in $O(|R_i|)$ time as follows: let F_1 be the set of facets in $F[R \cup S]$ incident to R_i, and let F_2 be the set of facets in $F[R_i \cup \Gamma]$ incident to R_i. We have $F_1 = F_2$. Clearly, $F_1 \subseteq F_2$ by the definition of Γ and since $R_i \cup \Gamma \subseteq R \cup S$. If there were a facet $f \in F_2 \setminus F_1$, the halfspace spanned by f would contain only points in $(R \cup S) \setminus (R_i \cup \Gamma)$. However, this would mean that in conv$(R \cup S)$ all the vertices of f are adjacent to a point in $(R \cup S) \setminus (R_i \cup \Gamma)$, contradicting the choice of Γ. The facets in F_1 can be extracted from conv $(R \cup S)$ in time $O(|R_i \cup \Gamma|)$, and the convex hull of $R_i \cup \Gamma$ can be completed in the same time, since the remaining facets involve only points in Γ, which has constant size. Now conv R_i can be extracted from conv $(R_i \cup \Gamma)$ in linear time, either by using `SplitHull` or by naively removing the points in Γ one by one.   □

This concludes the proof of Lemma 5.2.   □

Now, the splitting is performed by the algorithm `MultiSplit`. Please refer to Algorithm 5. For the recursion to work, we need to avoid small color classes.

Algorithm 5 Splitting arbitrary colorings

`MultiSplit(conv P)`

1. For all colors i with $|C_i| \leq 2^{\sqrt{\log n}}$, find conv C_i directly. Let K denote the remaining colors, and $Q = \bigcup_{i \in K} C_i$. Use `SplitHull` to determine conv Q.
2. Use Lemma 5.2 with $\alpha = 2^{\sqrt{\log n}}$ to obtain $S, R \subseteq Q$, a partition of $R_1, \ldots, R_\beta$ of R, and their convex hulls.
3. Call `MultiSplit(conv(Q \setminus (S \cup R)))` to find the hulls $\mathrm{conv}(C_i \cap (Q \setminus (S \cup R)))$.
4. For $j = 1, \ldots, \beta$, call `MultiSplit(conv R_j)` to find the hulls $\mathrm{conv}(C_i \cap R_j)$.
5. For $i \in K$, do
   (a) For $j = 1, \ldots, \beta$, merge $\mathrm{conv}(C_i \cap R_j)$ into conv S. This yields $\mathrm{conv}(S \cup (C_i \cap R))$.
   (b) Use `SplitHull` to extract $\mathrm{conv}(C_i \cap (S \cup R))$.
   (c) Compute the union of $\mathrm{conv}(C_i \cap (S \cup R))$ and $\mathrm{conv}(C_i \cap (Q \setminus (S \cup R)))$ to obtain conv C_i.

Thus, the algorithm first computes the convex hull of every C_i with $|C_i| \leq 2^{\sqrt{\log n}}$ in time $O(|C_i| \log |C_i|)$ [4, Chap. 11]. Let K denote the remaining colors, and let $Q = \bigcup_{i \in K} C_i$, $n_1 = |Q|$, and $n_2 = n - n_1$. For Step 5(a), we can use an algorithm to combine 3-polytopes separated by a plane [5, Chap. 9.3] to merge each $\mathrm{conv}(C_i \cap R_j)$ with $\mathrm{conv}(S)$. For $j \in \{1, \ldots, \beta\}$, this takes time $O(1 + |C_i \cap R_j|)$, since all new edges are incident to constantly many points in S by properties 3 and 4 of Lemma 5.2 and since the conflict sets of the R_j do not interact. By Theorem 3.1, Step 5(b) takes expected time $O(|S| + |C_i \cap R|)$, and as Chazelle [11] showed, Step 5(c) needs time $O(|C_i|)$. Hence, the total expected time for Step 5 is $O(|K| \cdot |S| + \sum_{i \in K} |C_i|)$. Recall that $|C_i| > 2^{\sqrt{\log n}}$ for all $i \in K$. Hence, $|K| < n/2^{\sqrt{\log n}}$ and $|K| \cdot |S| < n$. Therefore, the total running time of the algorithm is $O(n_2 \sqrt{\log n} + n)$, not counting the recursive calls. The first term represents the time for the convex hull computation in Step 1, and the second term counts the remaining steps. We get the following recursion for the running time:

$$T(n) \leq T(|Q \setminus (S \cup R)|) + \sum_{j=1}^{\beta} T(|R_j|) + c\left(n_2 \sqrt{\log n} + n\right)$$

for some constant $c > 0$. We know that $|R| \geq \gamma n_1$ and $\max_{1 \leq j \leq \beta} |R_j| \leq cn_1 \sqrt{\log n}/2^{\sqrt{\log n}}$, where $\gamma \in (0, 1]$, and we reuse c (making it larger if necessary). A simple induction shows that $T(n) \ll n\sqrt{\log n}$. Recalling that $|Q| = n_1$ and plugging in the inductive hypothesis $T(m) \leq \delta m \sqrt{\log m}$ for $m < n$ and some constant $\delta > 0$, we get

$$T(n) \leq \delta(n_1 - |R|)\sqrt{\log n} + \delta|R|\sqrt{\log n + \log(c\sqrt{\log n}) - \sqrt{\log n}}$$

$$+ cn_2\sqrt{\log n} + cn$$

$$\leq \delta(n_1 - |R|)\sqrt{\log n} + \delta|R|\sqrt{\log n - 0.5\sqrt{\log n}} + cn_2\sqrt{\log n} + cn$$

for n large enough. Since $\sqrt{\log n - 0.5\sqrt{\log n}} \leq \sqrt{\log n} - 1/4$, it follows that

$$T(n) \leq \delta(n_1 - |R|)\sqrt{\log n} + \delta|R|\sqrt{\log n} - \delta|R|/4 + cn_2\sqrt{\log n} + cn$$

$$\leq \delta n_1\sqrt{\log n} + 2cn_2\sqrt{\log n} + (c - \delta\gamma/4)n_1,$$

which is bounded by $\delta n\sqrt{\log n}$ for δ large enough. This concludes the induction and the proof of Theorem 5.1.

6 Points in Halfspaces

The problem in this section is to preprocess a point set to report quickly all the points contained in a query halfspace. However, not only do we want to find the points, but also their convex hull. We base our approach on a data structure by Chan [7] that uses *filtering search* [9]: first, it obtains a superset of the result with comparable size (the *candidate set*), and then examines each point individually to find the result. By storing not only the candidate sets, but also their convex hulls, we obtain a data structure that reports the convex hull of the points in a query halfspace by using SplitHull. We also show how to improve the preprocessing time over the straightforward $O(n \log^2 n)$.

Theorem 6.1 *Let $P \subseteq \mathbb{R}^3$ be an n-point set in general convex position. In $O(n \log n)$ time we can build a randomized data structure of $O(n \log n)$ size to answer queries of the following kind: given an oriented plane h, compute the convex hull of $P \cap h^+$, where h^+ denotes the left halfspace of h. The expected query time is $O(\log n + k)$, where $k = |P \cap h^+|$ denotes the output size.*

The main obstacle in improving the preprocessing time is this: given a sample $S \subseteq P$, compute the convex hulls of the conflict sets B_f for $f \in F[S]$. In the last section, we modified the conflict sets to obtain a simple algorithm for this problem. This is no longer possible, and we need a more sophisticated approach. Given a plane h, let $G(h)$ denote the induced subgraph of conv P with vertex set $P \cap h^+$ (i.e., $G(h)$ has vertex set $P \cap h^+$ and contains all edges of conv P with both endpoints in h^+). Here are some simple facts about $G(h)$ (e.g., [11]); see Fig. 8.

Lemma 6.2 *Let E be the set of edges in $G(h)$ incident to a facet of conv P that intersects h. There exists a closed walk[12] L along the edges in E such that L separates $G(h)$ from the rest of conv P. Every edge $e \in E$ occurs in L once or twice, depending on whether e is incident to one or two such facets. It follows that $G(h)$ is connected. Given $G(h)$, L can be found in time $O(|V[G(h)]|)$.*

Proof Consider the intersection $\mathcal{A}$ of h and conv P (interpreted as a subset of $\mathbb{R}^3$). The set $\mathcal{A}$ is a two-dimensional convex polygon whose edges correspond to the facets

---

[12]In our terminology, a *walk* is an arbitrary sequence of adjacent vertices, whereas a *path* consists of distinct vertices (except possibly the first and the last).

Fig. 8 A lace: h^+ corresponds to the inside of the circle. The lace L is shown in *dashed*
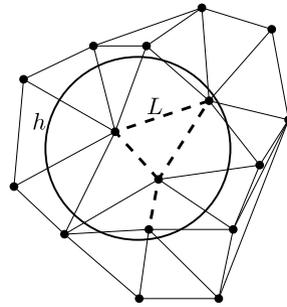
of conv P that intersect h. Let $F = f_1, f_2, \ldots, f_k$ be those facets in counterclockwise order along $\mathcal{A}$, and let $F' = f'_1, f'_2, \ldots, f'_k \subseteq F$ be the subsequence of facets that are incident to an edge in E. Since consecutive facets in F' share an incident vertex in $P \cap h^+$, the sequence F' induces a closed walk L along the edges in E. Every path from a point in $P \cap h^+$ to a point in $P \cap h^-$ has to cross a point incident to a facet in F'. Hence, L separates $G(h)$ from the rest of conv P. Furthermore, every edge in E appears once in L for each incident facet in F'. Finally, since consecutive edges in L are consecutive in the cyclic order of edges around their common endpoint in $G(h)$, L can be computed in time linear in the size of $G(h)$.   □

The walk L is called the *lace* of $G(h)$; see Fig. 8. Knowing $G(h)$ is enough to compute conv $(P \cap h^+)$ quickly.

Corollary 6.3 *Given* conv P *and* $G(h)$, *we can compute* conv $(P \cap h^+)$ *in time* $O(|P \cap h^+|)$.

Proof The idea is to find an intermediate polytope $\mathcal{P}$ of complexity $O(|P \cap h^+|)$ whose vertices contain $P \cap h^+$. This is done by computing (part of) the intersection of conv P with h^+ and adding a few edges to ensure general position; see Fig. 9. Using SplitHull, we extract conv $(P \cap h^+)$ from $\mathcal{P}$ in the desired time.

Let L be the lace of $G(h)$. By Lemma 6.2, L can be found in time $O(|P \cap h^+|)$ from $G(h)$. Let $F = f_1, \ldots, f_k$ be the sequence of facets in $F[P]$ that are incident to L and intersect h, where the ordering is according to L. The sequence F induces in the plane h a sequence E of line segments whose endpoints are in convex position. As the order of E corresponds to the convex hull order, we can compute the convex hull $\mathcal{C}$ of E in linear time. Let $V[\mathcal{C}]$ and $E[\mathcal{C}]$ denote the vertices and edges of $\mathcal{C}$.

We are now ready to construct the convex polytope $\mathcal{P}$. The set of $\mathcal{P}$'s facets consists of three disjoint parts, F_1, F_2, and F_3: (i) F_1 contains the facets of $G(h)$; (ii) for each line segment $e \in E$, F_2 contains a quadrilateral facet spanned by e and its corresponding edge $\tilde{e}$ in L.[13] Furthermore, for each $e \in E[\mathcal{C}] \setminus E$, F_2 contains a triangular facet f_e spanned by e and the point in $P \cap h^+$ incident to the edges whose intersections with h determine e; (iii) let Z be the unbounded prism with base $\mathcal{C}$ that extends

---

[13]That is, $\tilde{e}$ is the edge incident to the facet whose intersections with h create e.
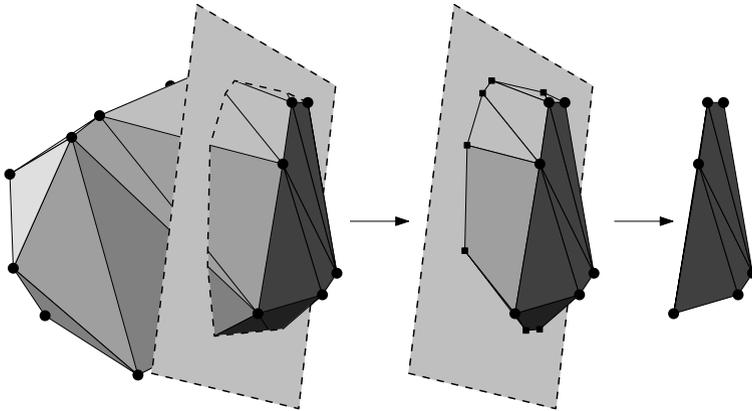
Fig. 9 The halfspace range reporting algorithm: The three stages of Corollary 6.3: Given $G(h)$, find an intermediate polytope that contains the result and split it

Fig. 10 Finding the conflict facets for an edge. $D_{\{p,q\}}$ is darkest, while D_p, D_q are lighter. D_p is bounded by dashed line segments, D_q by dotted line segments
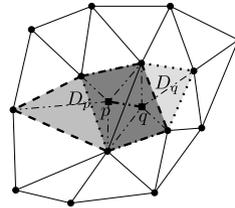
into h^-. Pick a point $q \in Z \cap \mathrm{conv}\, P$ infinitesimally close to h. F_3 contains all facets spanned by q and an edge in $E[\mathcal{C}]$. It is easily seen that the facets in $F_1 \cup F_2 \cup F_3$ are in convex position and bound a convex polytope $\mathcal{P}$ with $O(|P \cap h^+|)$ vertices. Since all the facets of $\mathcal{P}$ have bounded complexity and since all vertices in $V[\mathcal{C}]$ have bounded degree, we can perform a local perturbation of $V[\mathcal{C}]$ to obtain a polytope $\mathcal{P}'$ in general position. Now we compute $\mathrm{conv}\,(P \cap h^+)$ in time $O(|V[\mathcal{P}']|) = O(|P \cap h^+|)$ using `SplitHull`. □

For Corollary 6.3, we need to compute all the graphs $G(h_f)$ for $f \in F[S]$ (recall that h_f denotes the plane supporting f in $\mathrm{conv}\, S$).

Lemma 6.4 *Let $S \subseteq P$ be a random subset. Then the graphs $G(h_f)$ for $f \in F[S]$ can be computed in $O(n)$ expected time.*

Proof By Lemma A.4 the total size of the sets $P \cap h_f^+$ and hence the total complexity of the graphs $G(h_f)$ is $O(n)$. Let $e = (p, q) \in E[P]$, and let $D_e = D_p \cap D_q$ be the facets in conflict with both p and q; see Fig. 10. Note that $e \in G(h_f)$ precisely if $f \in D_e$. We will compute the sets D_e for $e \in E[P]$ and then use them to construct the graphs $G(h_f)$. Let T_e denote the graph on vertex set D_e where two vertices f_1, f_2 are adjacent if f_1, f_2 share an edge in $\mathrm{conv}\, S$ that is destroyed in $\mathrm{conv}\,(S \cup \{p, q\})$.

Algorithm 6 Computing the subgraphs

`ComputeSubgraphs`

1. For every $e \in E[P]$, if $f_e \neq \perp$, use f_e to compute D_e. For each $f \in D_e$, create records for the two half edges corresponding to e in $G(h_f)$.
2. For every point $p \in P$, use f_p to find D_p. For each $f \in D_p$, create a record p_f corresponding to p in $G(h_f)$. Every facet in D_p has a pointer `p` which we set to p_f. For each incident edge e of p in cyclic order, iterate through all facets $f \in D_e$. Use the pointer `p` of f to find the record p_f corresponding to p in $G(h_f)$ and add the appropriate half edge to the edge list of p_f.

Since T_e is connected,[14] it suffices to compute one facet $f_e \in D_e$ (if it exists). The remaining facets can be found by traversing T_e.

We extend `SubsetConflictWalk` to find conflict facets of edges by changing Step 2(d) as follows: when considering a neighbor $q \in \Gamma_P(p)$, we not only compute the conflict facet f_q, but also a conflict facet f_e for the edge $e = \{p, q\}$, if it exists. To do this, let Γ_p denote the simple polygon in conv S that bounds the conflict region of p. The facet $\tilde{f}_q \in F[S \cup p]$ is adjacent to an edge e_q on Γ_p, and q conflicts with at least one facet in conv S incident to e_q. Let $f_1, f_2 \in F[S]$ be the facets incident to e_q, where f_1 conflicts with p while f_2 does not. Now, if q conflicts with f_1, we set $f_q = f_e = f_1$, otherwise, we set $f_q = f_2$ and $f_e = \perp$.[15] This takes constant time, and therefore the running time of the algorithm remains linear, as in the proof of Lemma 4.2.

To prove correctness, we claim that if $D_e \neq \emptyset$, then $f_1 \in D_e$. Indeed, let T be the graph on vertex set $D_p \cup D_q$, where two vertices g_1, g_2 of T are adjacent if g_1, g_2 share an edge in $E[S]$ that is destroyed in conv $(S \cup \{p, q\})$. We have that T_e is a subgraph of T and that T is a tree (by convex position). Observe that e_q corresponds to the edge $e_q^* = \{f_1, f_2\}$ of T. Let T_1 be the connected component of $T \setminus e_q^*$, with $f_1 \in T_1$. Note that $D_p \subseteq V[T_1]$. Furthermore, at least one of f_1, f_2 is in conflict with q, hence $D_q \cap \{f_1, f_2\} \neq \emptyset$. Since the induced subgraph of T on vertex set D_q is connected, it follows that if $V[T_1] \cap D_q \neq \emptyset$, then D_q contains f_1, and hence $f_1 \in D_p \cap D_q = D_e$, as desired.

Using the sets D_e, we can now compute a DCEL representation of the graphs $G(h_f)$ in $O(n)$ time through careful pointer manipulation (Algorithm 6). □

Proof of Theorem 6.1 We rely on a variant of Chan's data structure [7] due to Ramos [31]. The candidate sets are the conflict sets of an appropriate gradation of P. By Corollary 6.3 and Lemma 6.4, we can find their convex hulls in time $O(n \log n)$. To process a query, we extend the original query algorithm to use `SplitHull` on the candidate set after coloring the points in h^+ blue.

The details are as follows: take a *gradation* $\emptyset = P_{-1} \subseteq P_0 \subseteq \cdots \subseteq P_{\log n} = P$, where P_{i-1} is derived from P_i by sampling every point with probability $1/2$. We

---

[14]We define the empty graph to be connected.

[15]As is often done in the study of programming languages, we use $\perp$ as a symbol for an undefined value.

compute the convex hulls conv P_i in time $O(n \log n)$. Using Lemma 6.4 and Corollary 6.3, we then find the convex hulls conv B_f for all the conflict sets B_f, $f \in F[P_i]$, $i = 0, \ldots, \log n$. Since this takes $O(n)$ time for each i, the total time is $O(n \log n)$. Now we switch into dual space. For this, we use duality with respect to the unit paraboloid which turns upper convex hulls into upper envelopes and lower convex hulls into lower envelopes [30, Chap. 2.4.1]. We compute two data structures, one for the upper envelope and one for the lower envelope, focusing the discussion on the lower envelope. For each $i = 0, \ldots, \log n$, we find the set of planes H_i dual to P_i and a canonical triangulation T_i of the lower envelope of H_i (this takes linear time since we know conv P_i). Then we construct a point location structure for the xy-projection of T_i. Every facet Δ of T_i is incident to at most three points of the lower envelope of H_i, corresponding to at most three facets f_1, f_2, f_3 of conv P_i. Let $B_\Delta = B_{f_1} \cup B_{f_2} \cup B_{f_3}$. We compute conv B_Δ in linear time [11] and store it with Δ. By the properties of canonical triangulations and the arguments given by Chan [7], the preprocessing phase takes expected time $O(n \log n)$ and uses expected space $O(n \log n)$. Then we repeat the process to obtain two independent data structures D_1, D_2.

Now suppose that we are given a query plane h. We need to find all the planes in H below h^*, the point dual to h. Let ℓ be the vertical line through h^*. Perform the following procedure simultaneously on D_1 and D_2, until one of them yields the answer: For $i = \log(n/\log n), \log(n/\log n) - 1, \ldots, 0$, locate the facet Δ_i of T_i intersected by ℓ in $O(\log n)$ time with the point location structure. Stop when the dual point h^* lies below the lower envelope of H_i. Now find the planes in H below h^* by inspecting the conflict set B_{Δ_i} and use $\texttt{SplitHull}$ to compute conv$(P \cap h^+)$ in $O(|B_{\Delta_i}|)$ time. As was argued by Ramos [31, Sect. 2.2.1], such a query takes expected time $O(\log n + |P \cap h^+|)$, as claimed. For completeness, we have included the calculation in Appendix B. $\qquad \square$

7 Union of Hulls

Finally, we consider the issue of output-sensitivity: if we are only interested in the hull of the blue points, under which circumstances can it be computed quickly without looking at the whole polytope? For this, we look at the problem DISJUNION, where the task is the following: given point sets $P_1, \ldots, P_k \subseteq \mathbb{R}^3$ and their convex hulls conv $P_1, \ldots,$ conv P_k such that conv $P_i \cap$ conv $P_j = \emptyset$ for $i \neq j$ and such that $P = \bigcup_{i=1}^{k} P_i$ is in convex position, we would like to compute conv P. In general, we cannot do better than to repeatedly merge pairs of the hulls.

Theorem 7.1 *Any algorithm that solves* DISJUNION *requires* $\Omega(|P| \log k)$ *comparisons.*

Proof We use an old lower bound [11, Sect. 4A] and combine it with Seidel's method of including the index as a coordinate [32]. We reduce from the *list merging* problem, in which k sorted lists of numbers need to be merged into one. We lift the lists onto the unit paraboloid $y = x^2$, using the z-coordinate to represent the index of the list.

Fig. 11 An illustration of the reduction from LISTMERGE to DISJUNION for the 3 lists (5, 9, 12, 14), (1, 8), (2, 4, 6, 7, 10, 13), and (3, 11). The path marked by the bold edges represents the merged list
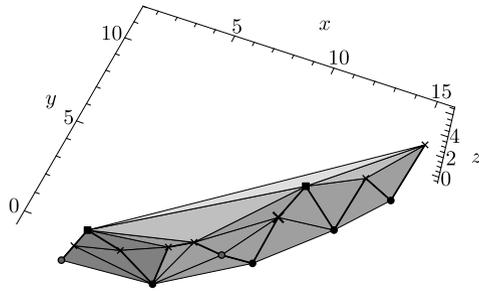
Clearly, the lifting and the individual convex hulls, which are pairwise disjoint, can be found in time $O(n)$. A simple geometric argument now shows that the merged list can be derived from the convex hull of the union in linear time; see Fig. 11.

More precisely, consider the problem LISTMERGE: given k sorted integer sequences $L_1, \ldots, L_k$, compute the sorted list $L = \bigcup_{i=1}^{k} L_i$. A straightforward counting argument shows that any algorithm for LISTMERGE requires $\Omega(|L| \log k)$ comparisons. We describe a linear time reduction from LISTMERGE to DISJUNION: let $L_i = (r_1, \ldots, r_j)$. We map L_i to a point set $P_i \subseteq \mathbb{R}^3$ by mapping each r_z to $p(r_z) = (r_z, r_z^2, i)$. All the points lie on the parabolic surface $y = x^2$, and hence $P = \bigcup_{i=1}^{k} P_i$ is in convex position. Furthermore, each P_i is contained in the plane $z = i$, and hence conv $P_i \cap$ conv $P_j = \emptyset$ for $i \neq j$. The conv P_i can be computed in linear time, since the lists L_i are sorted.

If r, s are consecutive in the sorted list L, then $p(r)p(s)$ is an edge of conv P. To see this, let $\hat{p}(r)$, $\hat{p}(s)$ denote the projections of $p(r)$, $p(s)$ onto the xy-plane, and let h denote the plane orthogonal to the xy-plane that contains the line segment $\hat{p}(r)\hat{p}(s)$. By definition, h contains $p(r)$ and $p(s)$, and hence also the line segment $p(r)p(s)$. Furthermore, all other points of P are on the same side of h. For this, fix $i \in \{1, \ldots, k\}$ and consider the parabola $Z_i : x \mapsto (x, x^2, i)$. Clearly, h intersects Z_i in the points (r, r^2, i) and (s, s^2, i), cutting off the part of Z_i between r and s. Since r and s are consecutive in L, this part contains no points in P_i. It follows that h supports the line segment $p(r)p(s)$, making it an edge of conv P. Consequently, it takes $\Omega(|P| \log k)$ time to compute conv P, since otherwise we could recover the sorted list L by examining the $O(|P|)$ edges of conv P.   □

Intuitively, what makes our lower bound instance hard is the fact that when merging conv P_i, we need to switch often between the individual hulls in an unpredictable way. We can avoid this by imposing additional constraints on the input, and thus obtain a better result.

Theorem 7.2 *Let $Q \subseteq \mathbb{R}^3$ be in general convex position. Let $P = \bigcup_{i=1}^{k} P_i \subseteq Q$ with $|P| = n$ such that the P_i are pairwise disjoint and the subgraphs $\mathrm{conv}\, Q|_{P_i}$ are connected. Then, given spanning trees $T_1, \ldots, T_k$ for $\mathrm{conv}\, Q|_{P_i}$, we can compute $\mathrm{conv}\, P$ in expected time $O(n \log^* n + k \log k)$.*

Proof We use Seidel's tracing technique [33]: pick a subset $K \subseteq P$ that meets each T_i in exactly one point, and an appropriate gradation $S_0 \subseteq \cdots \subseteq S_\beta = P \setminus K$ with

$\beta \ll \log^* n$. Then compute $\mathrm{conv}(S_0 \cup K)$ in time $O(n + k \log k)$ and successively each $\mathrm{conv}(S_i \cup K)$ in $O(n)$. Here, the bottleneck is to locate the conflict facets for S_{i+1} in $\mathrm{conv}(S_i \cup K)$. This is done using the spanning trees T_i and an appropriate variant of `SubsetConflictWalk`.

We may assume that $k < n/2$, since otherwise the theorem is easy. Let $K \subseteq P$ be such that K contains exactly one point of each P_i, and let $m = n - k$. Let $z = \max\{k, m/\log m\}$ and choose $1 \le \alpha \le \log^* m$ such that $m/\log^{(\alpha-1)} m < z \le m/\log^{(\alpha)} m$, where $\log^{(i)} m$ denotes the ith iterated logarithm[16] of m. Let $\beta = \log^* m - \alpha + 1$. Compute a gradation of subsets $S_0 \subseteq \cdots \subseteq S_\beta = P \setminus K$ such that S_i is a random subset of S_{i+1} with $|S_0| = z$ and $|S_{i+1}| = |S_i| \log^{(\alpha+i)} m / \log^{(\alpha+i+1)} m$ for $0 \le i < \beta$. By induction, it follows that $|S_i| \le m/\log^{(\alpha+i)}$. For $i = 0, \ldots, \beta$, let $\widetilde{S}_i = S_i \cup K$. We will show how to compute $\mathrm{conv}\,\widetilde{S}_{i+1}$ from $\mathrm{conv}\,\widetilde{S}_i$ in time $O(n)$ for each i. Furthermore, $\mathrm{conv}\,\widetilde{S}_0$ can be computed in time $O(n + k \log k)$ with a regular convex hull algorithm, as $|S_0 \cup K| = O(n/\log n + k)$. Hence, it takes $O(n \log^* n + k \log k)$ steps to compute $\mathrm{conv}\,Q = \mathrm{conv}\,\widetilde{S}_\beta$.

To derive $\mathrm{conv}\,\widetilde{S}_{i+1}$ from $\mathrm{conv}\,\widetilde{S}_i$, we proceed in two steps: first, we determine the conflict sets B_f for $f \in F[\widetilde{S}_i]$. Below, we will argue that this can be done in linear time. Then, we use the algorithm from Lemma A.5 to compute $\mathrm{conv}\,\widetilde{S}_{i+1}$. This takes time proportional to

$$\left(|S_{i+1}| + k \frac{|S_{i+1}|}{|S_i|}\right) \log \frac{|S_{i+1}|}{|S_i|} \le 2|S_{i+1}| \log\left(\frac{\log^{(\alpha+i)} m}{\log^{(\alpha+i+1)} m}\right)$$

$$\ll \frac{m}{\log^{(\alpha+i+1)} m} \log\left(\frac{\log^{(\alpha+i)} m}{\log^{(\alpha+i+1)} m}\right),$$

since $k \le |S_0| \le |S_i|$. The last term is $O(n)$, as claimed.

It remains to show how to find the conflict sets B_f in time $O(n)$. For each $j = 1, \ldots, k$, we determine conflict facets for P_j as follows: let $r_j = P_j \cap K$. We use a slight modification of `SubsetConflictWalk`: merge the neighbors of r_j in $\mathrm{conv}\,\widetilde{S}_i$ with the neighbors $\Gamma_{T_j}(r_j)$ of r_j in T_j in order to find a conflict facet f_p for each $p \in \Gamma_{T_j}(r_j)$. Then continue in a BFS-manner along T_j, inserting in turn each $p \in \Gamma_{T_j}(r_j)$ into $\mathrm{conv}\,\widetilde{S}_i$, and so on. As in Sect. 4, we see that the total time is proportional to

$$\sum_{p \in \widetilde{S}_i} \deg_{\widetilde{S}_i} p + \sum_{j=1}^{k} \sum_{p \in T_j} \deg_{T_j} p + \sum_{p \in P \setminus S_i} d_p \ll |\widetilde{S}_i| + |P| + \sum_{f \in F[\widetilde{S}_i]} b_f$$

$$\ll |P| + n - k + k \frac{|S_{i+1}|}{|S_i|},$$

by Lemma A.4. Since $k \le |S_i|$ and $|S_{i+i}| \le n$, the last term is linear. This finishes the proof. $\square$

---

[16]Defined by $\log^{(0)} m = m$ and $\log^{(k)} m = \max\{1, \log(\log^{(k-1)} m)\}$ for $k \ge 1$.

For our original question, this means that we can quickly compute the blue hull without considering the whole polytope, as long as the number of induced blue components is small.

Corollary 7.3 *Let $P \subseteq \mathbb{R}^3$ be a finite point set in general convex position, and let B be a subgraph of* conv P *with n vertices. Then* conv $V[B]$ *can be computed in time $O(n \log^* n + k \log k)$, where k denotes the number of connected components of B.*

Proof This follows immediately from Theorem 7.2. Given B, we can find spanning trees for its components in $O(n)$ time, using, say, depth-first search [20]. ☐

In particular, we get the following nice fact about Delaunay triangulations, which provides a Delaunay analogue to an old result by Bar-Yehuda and Chazelle [3].

Corollary 7.4 *Let $T = (V, E)$ be a Delaunay triangulation, and let $S \subseteq T$ be a set of n vertices and edges of T with k connected components. Then the Delaunay triangulation of S can be computed in time $O(n \log^* n + k \log k)$.*

Proof Use Corollary 7.3 and the connection between planar Delaunay triangulations and three-dimensional convex hulls [4, Chap. 11.4]. ☐

Appendix A: Clarkson–Shor Toolbox

We review a few tools from geometric random sampling theory [19, 30]. Our presentation follows Ramos [31]. Let $P \subseteq \mathbb{R}^3$ with $|P| = n$, and $K \subseteq P$ with $|K| = k$. Given a triple $\mathbf{u} = (p_1, p_2, p_3) \in P^3$, let $h_{\mathbf{u}}$ be the plane spanned by $\mathbf{u}$, oriented such that the set of vectors $\{u_2 - u_1, u_3 - u_1, p - u_1\}$ has positive determinant for $p \in h_{\mathbf{u}}^+$. A point $p \in P$ *conflicts* with $\mathbf{u}$ if p lies in $h_{\mathbf{u}}^+$. Let $B_{\mathbf{u}}$ denote the set of all points in P that conflict with $\mathbf{u}$, and $b_{\mathbf{u}} = |B_{\mathbf{u}}|$.

Lemma A.1 *Fix $p \in (0, 1]$ and $t \geq 1$. Let $S \subseteq P \setminus K$ be a random subset of size $p(n - k)$, and let $S' \subseteq P \setminus K$ be a random subset of size $p'(n - k)$ for $p' = p/t$. Suppose that $p'(n - k) \geq 4$. Fix $\mathbf{u} = (p_1, p_2, p_2) \in P^3$, and let $f_{\mathbf{u}}$ be the facet defined by $\mathbf{u}$. Then*

$$\Pr[f_{\mathbf{u}} \in F[S \cup K]] \ll t^3 \exp\left(-\frac{(t - 1)pb_{\mathbf{u}}}{t}\right) \Pr[f_{\mathbf{u}} \in F[S' \cup K]]. \qquad (6)$$

Proof Let $\sigma = \Pr[f_{\mathbf{u}} \in F[S \cup K]]$ and $\sigma' = \Pr[f_{\mathbf{u}} \in F[S' \cup K]]$. Note that $f_{\mathbf{u}}$ appears in $F[S \cup K]$ precisely if $\mathbf{u} \subseteq S \cup K$ and $B_{\mathbf{u}} \cap (S \cup K) = \emptyset$. If $K \cap B_{\mathbf{u}} \neq \emptyset$, then $\sigma = \sigma' = 0$, and the lemma holds. Thus, we may assume that K and $B_{\mathbf{u}}$ are disjoint. Let $m = n - k$, and let $d_{\mathbf{u}}$ denote $3 - |K \cap \mathbf{u}|$, the number of points in $\mathbf{u}$ not in K. Since there are $\binom{m - b_{\mathbf{u}} - d_{\mathbf{u}}}{pm - d_{\mathbf{u}}}$ ways of choosing a pm-subset from $P \setminus K$ that avoids all elements in $B_{\mathbf{u}}$ and contains all the relevant points of $\mathbf{u}$, we have

$$\sigma = \binom{m - b_{\mathbf{u}} - d_{\mathbf{u}}}{pm - d_{\mathbf{u}}} \bigg/ \binom{m}{pm}$$

$$= \frac{\prod_{j=0}^{pm-d_{\mathbf{u}}-1}(m-b_{\mathbf{u}}-d_{\mathbf{u}}-j)}{\prod_{j=0}^{pm-d_{\mathbf{u}}-1}(pm-d_{\mathbf{u}}-j)} \bigg/ \frac{\prod_{j=0}^{pm-1}(m-j)}{\prod_{j=0}^{pm-1}(pm-j)}$$

$$= \prod_{j=0}^{d_{\mathbf{u}}-1} \frac{pm-j}{m-j} \prod_{j=0}^{pm-d_{\mathbf{u}}-1} \frac{m-b_{\mathbf{u}}-d_{\mathbf{u}}-j}{m-d_{\mathbf{u}}-j}$$

$$\leq p^{d_{\mathbf{u}}} \prod_{j=0}^{pm-d_{\mathbf{u}}-1} \left(1 - \frac{b_{\mathbf{u}}}{m-d_{\mathbf{u}}-j}\right).$$

Similarly, we get

$$\sigma' = \prod_{j=0}^{d_{\mathbf{u}}-1} \frac{p'm-j}{m-j} \prod_{j=0}^{p'm-d_{\mathbf{u}}-1} \left(1 - \frac{b_{\mathbf{u}}}{m-d_{\mathbf{u}}-j}\right),$$

and since $p'm \geq 4$ and $j \leq 2$ (in the first product), it follows that

$$\sigma' \geq \left(\frac{p'}{2}\right)^{d_{\mathbf{u}}} \prod_{j=0}^{p'm-d_{\mathbf{u}}-1} \left(1 - \frac{b_{\mathbf{u}}}{m-d_{\mathbf{u}}-j}\right).$$

Therefore, since $p' = p/t$,

$$\frac{\sigma}{\sigma'} \leq 8\left(\frac{p}{p'}\right)^{d_{\mathbf{u}}} \prod_{j=p'm-d_{\mathbf{u}}}^{pm-d_{\mathbf{u}}-1} \left(1 - \frac{b_{\mathbf{u}}}{m-d_{\mathbf{u}}-j}\right) \leq 8t^3 \left(1 - \frac{b_{\mathbf{u}}}{m}\right)^{(t-1)pm/t}$$

$$\leq 8t^3 \exp\left(-\frac{(t-1)pb_{\mathbf{u}}}{t}\right),$$

as desired. $\square$

The lemma implies a Chernoff-type bound for the conflict size of a random sample.

Lemma A.2 Fix $p \in (0,1]$ and let $S \subseteq P$ be a random subset of size pn. Fix $t \geq 1$ such that $t \leq pn/4$ and let $F_{\geq t} = \{f \in F[S] \mid b_f \geq t/p\}$. Then

$$\mathbf{E}\left[|F_{\geq t}|\right] \ll t^2 e^{-t} pn.$$

Proof Let $S' \subseteq P$ be a random subset of size pn/t. Since $pn/t \geq 4$, we have

$$\mathbf{E}\left[|F_{\geq t}|\right] = \sum_{\substack{\mathbf{u} \in P^3 \\ b_{\mathbf{u}} \geq t/p}} \Pr\left[f_{\mathbf{u}} \in F[S]\right]$$

$$\ll \sum_{\substack{\mathbf{u} \in P^3 \\ b_{\mathbf{u}} \geq t/p}} t^3 \exp\left(-\frac{(t-1)pb_{\mathbf{u}}}{t}\right) \Pr\left[f_{\mathbf{u}} \in F[S']\right] \qquad \text{(by (6))}$$

$$\ll t^3 e^{-t} \mathbf{E}\left[\left|F[S']\right|\right] \ll t^2 e^{-t} pn,$$

because $\mathbf{E}\left[|F[S']|\right] \ll pn/t$. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

Next, we want to bound the average conflict size. For this, we first determine the average for a particular function, from which we then deduce bounds for a large class of well-behaved functions.

Lemma A.3 *Fix $p \in (0, 1]$ and let $S \subseteq P \setminus K$ be a random subset of size $p(n-k)$. Then*

$$\mathbf{E}\left[\sum_{f \in F[S \cup K]} \exp\left(\frac{pb_f}{2}\right)\right] \ll p(n-k) + k. \qquad (7)$$

Proof We may assume that $p(n-k)/2 \geq 4$, because otherwise $pb_f = O(1)$ for every $f \in F[S \cup K]$ (as all these f have $b_f \leq n-k$ and $\mathrm{conv}(S \cup K)$ has $O(p(n-k)+k)$ facets), and the lemma would hold trivially. Let $S' \subseteq P \setminus K$ be a random subset of size $p(n-k)/2$. We have

$$\mathbf{E}\left[\sum_{f \in F[S \cup K]} \exp\left(\frac{pb_f}{2}\right)\right] = \sum_{\mathbf{u} \in P^3} \Pr\left[f_{\mathbf{u}} \in F[S \cup K]\right] \exp\left(\frac{pb_{\mathbf{u}}}{2}\right)$$

$$\ll \sum_{\mathbf{u} \in P^3} \Pr\left[f_{\mathbf{u}} \in F[S' \cup K]\right] \qquad \text{(by (6))}$$

$$= \mathbf{E}\left[\left|F[S' \cup K]\right|\right] \ll p(n-k) + k.$$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

Using this bound, we can show that the sum of every well-behaved function over the conflict sizes of a random sample gives the value one would expect. This remains true if a few points from P are always included in the sample.

Lemma A.4 *Fix $p \in (0, 1]$ and let $S \subseteq P \setminus K$ be a random subset of size $p(n-k)$. Let g be a function such that $g(tn) \ll e^t g(n)$ for all $t \geq 0$. Then*

$$\mathbf{E}\left[\sum_{f \in F[S \cup K]} g(b_f)\right] \ll \left(p(n-k) + k\right) g\left(1/p\right).$$

In particular, choosing $k = 0$ and $g : n \mapsto n^{\gamma}$ for $\gamma \geq 0$, we have

$$\mathbf{E}\left[\sum_{f \in F[S]} b_f^{\gamma}\right] \ll np^{1-\gamma}, \qquad (8)$$

and choosing $g : n \mapsto n \log n$, we get

$$\mathbf{E}\left[\sum_{f \in F[S \cup K]} b_f \log b_f\right] \ll (n - k + k/p) \log(1/p). \qquad (9)$$

Proof We have

$$\mathbf{E}\left[\sum_{f\in F[S\cup K]} g(b_f)\right] = \mathbf{E}\left[\sum_{f\in F[S\cup K]} g\left(\frac{pb_f}{2}\cdot\frac{2}{p}\right)\right]$$

$$\ll e^2 g(1/p)\mathbf{E}\left[\sum_{f\in F[S\cup K]} \exp(pb_f/2)\right]$$

$$\ll \big(p(n-k)+k\big)g\,(1/p),$$

by (7). $\square$

The following lemma follows from a standard application of the geometric divide-and-conquer technique [12, 18, 19] and asserts that a convex hull can be computed faster if a random partial hull and the corresponding conflict information are known.

Lemma A.5 *Fix $p\in(0,1]$ and let $S\subseteq P\setminus K$ be a subset of size $p(n-k)$. Suppose that $\mathrm{conv}\,(S\cup K)$ and the conflict sets $B_f\subseteq P$ for $f\in F[S\cup K]$ are available. Then we can find $\mathrm{conv}\,P$ in expected time $\sum_{f\in F[S\cup K]} b_f\log b_f$. In particular, if S is a random subset, the running time is $O((n-k+k/p)\log(1/p))$.*

Proof Let $\widetilde{S}=S\cup K$. Without loss of generality, we assume that $\mathrm{conv}\,\widetilde{S}$ contains the origin. Instead of $\mathrm{conv}\,P$, we compute $(P^*)^\cap$, the intersection of the halfspaces dual to the points in P. For this, we first obtain $(\widetilde{S}^*)^\cap$, which takes linear time, since $\mathrm{conv}\,\widetilde{S}$ is known. The vertices of $(\widetilde{S}^*)^\cap$ correspond to the facets of $\mathrm{conv}\,\widetilde{S}$. In particular, each vertex f of $(\widetilde{S}^*)^\cap$ has a conflict list B_f^* of size b_f. We compute a tetrahedralization $\mathcal{T}$ of $(\widetilde{S}^*)^\cap$ as follows: for each facet g of $(\widetilde{S}^*)^\cap$, determine the vertex f_g incident to g with minimum b_{f_g}.[17] The vertex f_g is called the *apex* of g. Triangulate g by adding line segments from the apex to all other vertices of g. Finally, extend this triangulation to a tetrahedralization by lifting it to the origin. This takes linear time.

The conflict set of a simplex s is precisely $B_s^*=B_{f_1}^*\cup B_{f_2}^*\cup B_{f_3}^*$, where f_1, f_2, f_3 are the vertices of s other than the origin. Let $b_s=|B_s|$. We determine the intersection of the halfspaces in B_s^* and clip it to s. Then we glue the parts together to obtain $(P^*)^\cap$ and hence $\mathrm{conv}\,P$. This takes time $O(\sum_{s\in\mathcal{T}} b_s\log b_s)$. Consider a simplex $s\in\mathcal{T}$ and let f_s, f_1, f_2 be its vertices other than the origin. Here f_s denotes the apex of the facet of $(\widetilde{S}^*)^\cap$ that contains a facet of s, and we call f_s also the *apex* of s. By definition, we have $b_s=f_s+f_1+f_2\le 2(f_1+f_2)$, and hence $b_s\log b_s\ll f_1\log f_1+f_2\log f_2$. By general position, every vertex of $(\widetilde{S}^*)^\cap$ has degree 3 and thus appears in only constantly many simplices of $\mathcal{T}$ as a nonapex. Hence, $\sum_{s\in\mathcal{T}} b_s\log b_s\ll\sum_{f\in F[\widetilde{S}]} b_f\log b_f$, as claimed. Now, if S is a random sample, this sum is proportional to $(n-k+k/p)\log(1/p)$, by Corollary A.4(9). $\square$

---

[17]We take the lexicographically smallest if there is more than one such vertex.

Appendix B: Analyzing the Halfspace Range Reporting Structure

We present Ramos' analysis [31] of a variant of Chan's data structure [7]. Let $\mathcal{E}_i$ denote the event that i is the largest index for which h^* lies below the lower envelope of H_i in either D_1 or D_2. The expected running time is

$$\sum_{i=0}^{\log n - \log\log n} \left((\log n - \log\log n - i)\log n + \mathbf{E}\left[|B_{\Delta_i}| \,|\, \mathcal{E}_i \right] \right) \Pr[\mathcal{E}_i]. \qquad (10)$$

Let $k = |P \cap h^+|$. Note that $\mathbf{E}[|B_{\Delta_i}| \,|\, \mathcal{E}_i] = O(k + n/2^i)$, since by standard random sampling theory the expected size of a conflict list in T_i is $O(n/2^i)$ and the random choices for the points in $P \setminus (P \cap h^+)$ are independent of $\mathcal{E}_i$. Thus,

$$(10) \ll \sum_{i=0}^{\log(n/\log n)} \left(\left(\log\left(\frac{n}{\log n}\right) - i \right)\log n + k + \frac{n}{2^i} \right) \Pr[\mathcal{E}_i]$$

$$\ll k + \sum_{i=\log(n/k)}^{\log(n/\log n)} \left(\left(\log\left(\frac{n}{\log n}\right) - i \right)\log n + \frac{n}{2^i} \right) \Pr[\mathcal{E}_i]$$

$$+ \sum_{i=0}^{\log(n/k)-1} \left(\left(\log\left(\frac{n}{\log n}\right) - i \right)\log n + \frac{n}{2^i} \right) \Pr[\mathcal{E}_i].$$

If $k < \log n$, the first sum is zero. Otherwise, we get

$$\sum_{i=\log(n/k)}^{\log(n/\log n)} \left(\left(\log\left(\frac{n}{\log n}\right) - i \right)\log n + \frac{n}{2^i} \right) \Pr[\mathcal{E}_i]$$

$$\leq (\log n)\log\left(\frac{k}{\log n}\right) + \sum_{i \geq \log(n/k)} \frac{n}{2^i} \ll k.$$

To bound the second sum, we observe that $\Pr[\mathcal{E}_i] \leq (k2^{i+1}/n)^2$, because if $\mathcal{E}_i$ holds, then P_{i+1} in both data structures D_1 and D_2 must necessarily contain one of the at most k points in $P \cap h^+$. We get

$$\sum_{i=0}^{\log(n/k)-1} \left(\left(\log\left(\frac{n}{\log n}\right) - i \right)\log n + \frac{n}{2^i} \right) \Pr[\mathcal{E}_i]$$

$$\leq (\log n)\log\left(\frac{k}{\log n}\right) + \sum_{i=0}^{\log(n/k)-1} \left(\left(\log\left(\frac{n}{k}\right) - i \right)\log n + \frac{n}{2^i} \right) \left(\frac{k2^{i+1}}{n} \right)^2$$

$$\ll k + \sum_{j=1}^{\log(n/k)} \left(j\log n + 2^j k \right) 2^{-2(j-1)} \ll \log n + k,$$

as desired.

References

1. Aggarwal, A., Guibas, L.J., Saxe, J., Shor, P.W.: A linear-time algorithm for computing the Voronoi diagram of a convex polygon. Discrete Comput. Geom. **4**(6), 591–604 (1989)
2. Amato, N.M., Goodrich, M.T., Ramos, E.A.: Linear-time triangulation of a simple polygon made easier via randomization. In: Proc. 16th Annu. ACM Sympos. Comput. Geom. (SoCG), pp. 201–212 (2000)
3. Bar-Yehuda, R., Chazelle, B.: Triangulating disjoint Jordan chains. Int. J. Comput. Geom. Appl. **4**(4), 475–481 (1994)
4. de Berg, M., Cheong, O., van Kreveld, M., Overmars, M.: Computational Geometry: Algorithms and Applications, 3rd edn. Springer, Berlin (2008)
5. Boissonnat, J.D., Yvinec, M.: Algorithmic Geometry. Cambridge University Press, New York (1998)
6. Buchin, K., Mulzer, W.: Delaunay triangulations in $O(\mathrm{sort}(n))$ time and more. In: Proc. 50th Annu. IEEE Sympos. Found. Comput. Sci. (FOCS), pp. 139–148 (2009)
7. Chan, T.M.: Random sampling, halfspace range reporting, and construction of $(\leq k)$-levels in three dimensions. SIAM J. Comput. **30**(2), 561–575 (2000)
8. Chan, T.M.: Three problems about simple polygons. Comput. Geom. Theory Appl. **35**(3), 209–217 (2006)
9. Chazelle, B.: Filtering search: a new approach to query-answering. SIAM J. Comput. **15**(3), 703–724 (1986)
10. Chazelle, B.: Triangulating a simple polygon in linear time. Discrete Comput. Geom. **6**(5), 485–524 (1991)
11. Chazelle, B.: An optimal algorithm for intersecting three-dimensional convex polyhedra. SIAM J. Comput. **21**(4), 671–696 (1992)
12. Chazelle, B.: The Discrepancy Method: Randomness and Complexity. Cambridge University Press, New York (2000)
13. Chazelle, B., Devillers, O., Hurtado, F., Mora, M., Sacristán, V., Teillaud, M.: Splitting a Delaunay triangulation in linear time. Algorithmica **34**(1), 39–46 (2002)
14. Chew, L.P.: Building Voronoi diagrams for convex polygons in linear expected time. Tech. Rep. PCS-TR90-147, Dartmouth College, Computer Science, Hanover, NH (1990)
15. Chew, L.P., Fortune, S.: Sorting helps for Voronoi diagrams. Algorithmica **18**(2), 217–228 (1997)
16. Chin, F., Snoeyink, J., Wang, C.A.: Finding the medial axis of a simple polygon in linear time. Discrete Comput. Geom. **21**(3), 405–420 (1999)
17. Chin, F., Wang, C.A.: Finding the constrained Delaunay triangulation and constrained Voronoi diagram of a simple polygon in linear time. SIAM J. Comput. **28**(2), 471–486 (1998)
18. Clarkson, K.L.: A randomized algorithm for closest-point queries. SIAM J. Comput. **17**(4), 830–847 (1988)
19. Clarkson, K.L., Shor, P.W.: Applications of random sampling in computational geometry. II. Discrete Comput. Geom. **4**(5), 387–421 (1989)
20. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd edn. MIT Press, Cambridge (2009)
21. Devillers, O.: Randomization yields simple $O(n \log^* n)$ algorithms for difficult $\Omega(n)$ problems. Int. J. Comput. Geom. Appl. **2**(1), 97–111 (1992)
22. Djidjev, H.N., Lingas, A.: On computing Voronoi diagrams for sorted point sets. Int. J. Comput. Geom. Appl. **5**(3), 327–337 (1995)
23. Dobkin, D.P., Kirkpatrick, D.G.: Fast detection of polyhedral intersection. Theor. Comput. Sci. **27**(3), 241–253 (1983)
24. Dobkin, D.P., Kirkpatrick, D.G.: A linear algorithm for determining the separation of convex polyhedra. J. Algorithms **6**(3), 381–392 (1985)
25. Fournier, H., Vigneron, A.: A tight lower bound for computing the diameter of a 3D convex polytope. Algorithmica **49**(3), 245–257 (2007)
26. Kirkpatrick, D.G., Klawe, M.M., Tarjan, R.E.: Polygon triangulation in $O(n \log \log n)$ time with simple data structures. Discrete Comput. Geom. **7**(4), 329–346 (1992)
27. Klein, R., Lingas, A.: A linear-time randomized algorithm for the bounded Voronoi diagram of a simple polygon. Int. J. Comput. Geom. Appl. **6**(3), 263–278 (1996)
28. van Kreveld, M.J., Löffler, M., Mitchell, J.S.B.: Preprocessing imprecise points and splitting triangulations. In: Proc. 19th Annu. Internat. Sympos. Algorithms Comput. (ISAAC), pp. 544–555 (2008)
29. Matoušek, J.: Lectures on Discrete Geometry. Graduate Texts in Mathematics, vol. 212. Springer, New York (2002)

30. Mulmuley, K.: Computational Geometry: An Introduction Through Randomized Algorithms. Englewood Cliffs, Prentice-Hall (1994)
31. Ramos, E.A.: On range reporting, ray shooting and k-level construction. In: Proc. 15th Annu. ACM Sympos. Comput. Geom. (SoCG), pp. 390–399 (1999)
32. Seidel, R.: A method for proving lower bounds for certain geometric problems. Tech. Rep. TR84-592, Cornell University, Ithaca, NY, USA (1984)
33. Seidel, R.: A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. Comput. Geom. Theory Appl. **1**(1), 51–64 (1991)