Computing Hereditary Convex Structures*

Bernard Chazelle
chazelle@cs.princeton.edu

Wolfgang Mulzer
wmulzer@cs.princeton.edu

Department of Computer Science
Princeton University
35 Olden Street
Princeton, NJ 08540, USA

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 $o(n \log n)$? What if we have $\chi > 2$ 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.

Categories and Subject Descriptors

F.2.2 [**Nonnumerical Algorithms and Problems**]: Geometrical problems and computations

General Terms

Algorithms, Theory

Keywords

convex polytope, half-space 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 [10] how to compute the Voronoi diagrams of any given subset in linear time.[1] The authors asked whether the convex hull of an arbitrary subset of the vertices of a convex 3-polytope can be computed in linear time, too. 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. We show how to do this in linear time, which answers the main open question in [10]. 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. 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 $O(k + \log n)$, 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 [8]. 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 [1,11]. Given a spanning subgraph of degree at most d, the DT can be completed in time $O(nd \log^* n)$ [18]. In fact, Djidjev and Lingas have proven linearity for any set of points forming a monotone chain in both x and y directions [19]. This might suggest that the hardness of DT is really confined to sorting. Of course, we know this is not true: in the general Euclidean case, sorting does not help (though it does in ℓ_∞ [12]). Ranking the points in any one direction still leaves us with a $\Theta(n \log n)$ complexity [19]. The simplic-

*This work was supported in part by NSF grant CCF-0634958 and NSF CCF 0832797.

[1]All our algorithms are randomized, so the complexity is to be understood in the expected sense.

SCG'09, June 8–10, 2009, Aarhus, Denmark.

ity of a polygon is known to "linearize" many problems that otherwise exhibit $\Omega(n \log n)$ lower bounds, eg, polygon triangulation [2, 7, 26], medial axis [13], constrained Delaunay triangulation [14, 21]. Our work fits into that mold.

Hereditary algorithms are nothing new. Given a subset of a simple polygon, Chan [5] has shown how to compute its convex hull in linear time[2] and how to triangulate it in $O(n \log^* n)$ time. Van Kreveld, Löffler, and Mitchell [27] proved that any subset of a given triangulation can in fact be triangulated in linear time. Chan [5] also gave an $\Omega(n \log n)$ lower bound for hereditary trapezoidal decompositions, and there are many more 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 [25]; nor does knowing the convex hull of P help in finding its diameter [20].

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. SPLITTING POLYTOPES

We are given an n-point set $P \subseteq \mathbb{R}^3$ in general convex position.[3] 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 $\operatorname{conv} P$, we show how to obtain $\operatorname{conv} B$ in linear time.

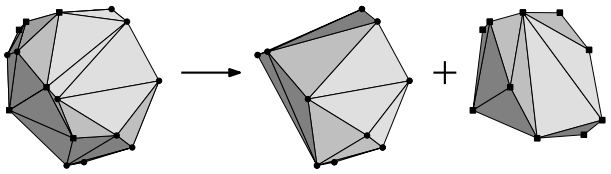

Figure 1: Given their joint convex hull, we can find the red and blue hulls in linear time.

THEOREM 2.1. *Let $P \subseteq \mathbb{R}^3$ be a set of n points in general convex position, colored red and blue. Given* $\operatorname{conv} P$*, the convex hull of the blue points can be computed in $O(n)$ expected time.*

An edge of $\operatorname{conv} P$ 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` (Algorithm 1) that receives the convex hull and a two-coloring of P. `SplitHull` 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. These blue points are later reinserted into the recursively computed blue hull, which can be done efficiently by remembering a landmark from where to start the conflict location. `SplitHull` is easily shown to be correct.

LEMMA 2.2. *`SplitHull`*$(\operatorname{conv} P)$ *computes* $\operatorname{conv} B$.

[2]Here, *linear time* means linear in the size of the whole structure, not just the subset.

[3]See Appendix A for basic definitions and notation.

Algorithm `SplitHull`$(\operatorname{conv} P)$

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

Algorithm 1: Splitting a bichromatic convex hull.

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 $\operatorname{conv} P^-$, then the same holds in $\operatorname{conv} B^-$: since e has both endpoints in B^-, a supporting plane for e in $\operatorname{conv} P^-$ supports e also in $\operatorname{conv} B^-$, and since $\operatorname{conv} B^- \subseteq \operatorname{conv} P^-$, the triangle spanned by b and e intersects $\operatorname{conv} B^-$ 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 $O(|D_b|)$ [17, Chapter 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 2.3. *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, we merely lose a constant factor. 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 2.4. *Let $\widetilde{B}$ be the subset of the blue points b with the following properties: (i)* $\deg_P b \leq 6$*; and (ii) b is a vertex of a blue facet of* $\operatorname{conv} P$ *or $E[P \setminus b] \setminus E[P]$ contains at least one blue edge. There exists a constant d_0 such that if all red points have degree at least d_0, then $|\widetilde{B}| \geq |P|/5$.*

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, then either (i) b is incident to a facet with a red edge; or (ii) b's neighborhood has only bichromatic edges and to delete b from $\operatorname{conv} P$ creates no blue edge. We bound (i) and (ii) 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 $|E[P]| = 3n-6$. Let B' be the set of blue points b with $\deg_P b \le 6$. Since $\operatorname{conv} P$ is three-connected [22, 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

$$|B'| > n/4. \tag{1}$$

A similar calculation using that all red points have degree greater than d_0 shows $|R| < 4n/d_0$ (for d_0 large enough). Let E_R denote the set of red edges in $\operatorname{conv} P$. Since every red edge of $\operatorname{conv} P$ is an edge of $\operatorname{conv} R$,

$$|E_R| \le |E[R]| \le 3|R| - 6 < 12n/d_0. \tag{2}$$

For $b \in B'$, let Γ_b be the simple polygon formed by b's neighbors in $\operatorname{conv} P$, and let C be the set of points $b \in B'$ such that Γ_b contains a red edge. 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 (2),

$$|C| \le 2|E_R| < 24n/d_0. \tag{3}$$

Now, let $D \subseteq B'$ be the set of points b such that Γ_b has no monochromatic edge. For any such b, red and blue points alternate along Γ_b and $\deg_P b \in \{4, 6\}$. 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, namely the endpoints of an edge in $E[P]$ that occludes e in $\operatorname{conv} P$. 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 (there has to be a diagonal between two points at distance 2 along the polygon, see Figure 2). Let D' be the set of points in D that create no blue edge. By the previous discussion and (2),

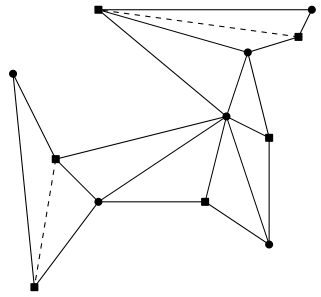

Figure 2: Every triangulation of a two-colored simple polygon contains at least one monochromatic diagonal (dashed lines).

$$|D'| \le 2|E[R]| < 24n/d_0. \tag{4}$$

To finish, we observe that all the points in the set $B' \setminus (C \cup D')$ are pleasant and that by (1, 3, 4) it contains at least $(1/4 - 48/d_0)\, n > n/5$ points, for d_0 large enough. □

By Lemma 2.4 we expect 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 2.4. 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 $\operatorname{conv} B$ takes constant expected time, which is exactly the cost of inserting b into $\operatorname{conv}(B \setminus b)$ [17, Chapter 11.2]. □

Theorem 2.1 follows from Lemmas 2.2 and 2.3, since the number of recursive calls is $O(n)$.

3. SPLITTING RANDOM COLORINGS

Now, we extend `SplitHull` to handle more than two colors: for $P \subseteq \mathbb{R}^3$, let $c : P \to \{1, \dots, \chi\}$ be a *coloring* of P. For $i \in \{1, \dots, \chi\}$, we let $C_i = c^{-1}(i)$ denote the point set 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 $\{1, \dots, \chi\}$. This section deals with random colorings, the next one is about arbitrary colorings.

THEOREM 3.1. *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 a random coloring of P. Given $\operatorname{conv} P$, we can compute the convex hulls $\operatorname{conv} C_1, \dots, \operatorname{conv} C_\chi$ in $O(n)$ expected time (the expectation is over the coloring and the random choices of the algorithm).*

The algorithm for Theorem 3.1 is called `RandMultiSplit` (Algorithm 2). It receives the convex hull and a coloring of P, and it computes the convex hull of a random sample S 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.

Algorithm `RandMultiSplit`($\operatorname{conv} P$) (* see Figure 3a *)

1. Pick a random sample $S \subseteq P$ of size n/χ and compute $\operatorname{conv} S$.
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 $\operatorname{conv} S$.
   (b) Extract $\operatorname{conv} C_i$ from $\operatorname{conv}(C_i \cup S)$.

Algorithm 2: Splitting random colorings.

Clearly, the algorithm correctly computes the $\operatorname{conv} C_i$. Using `SplitHull`, Step 1 requires $O(n)$ time. The analysis of Step 2 needs more work.

LEMMA 3.2. *Step 2 takes $O(n)$ expected time.*

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

CLAIM 3.3. *Let $Q \subseteq P$ and $p \in Q$. Assume that both $\operatorname{conv} Q$ and $\operatorname{conv} P$ are available. In $O(\deg_Q p + \deg_P p)$ time, we can compute a conflict facet $f_q \in F[Q]$ for every neighbor $q \in \Gamma_P(p)$ of p.*

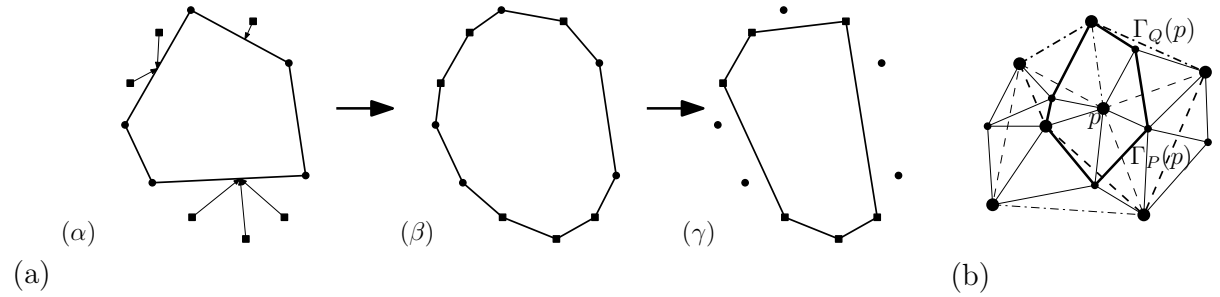

Figure 3: Splitting random colorings: (a) the algorithm (α) computes $\operatorname{conv} S$ and conflict facets for C_i, (β) inserts C_i into $\operatorname{conv} S$, and (γ) extracts $\operatorname{conv} C_i$. The points in C_i are shown as boxes, S as circles. (b) Claim 3.3: the facets $F[Q]$ are shown dashed, $F[P]$ solid. Merge $\Gamma_P(p)$ with $\Gamma_Q(p)$ to determine its conflict facets.

PROOF. Consider an *overlay* of $\operatorname{conv} Q$ and $\operatorname{conv} P$, ie, a central projection of their vertices and edges onto the unit sphere centered at a point $O \in \operatorname{conv} Q$. 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: 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 $\operatorname{conv} Q$. But then pq could not be an edge of $\operatorname{conv} P$, as $\operatorname{conv} Q \subseteq \operatorname{conv} P$. 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 Figure 3b. This takes time $O(\deg_Q p + \deg_P p)$. □

The conflict facets for P can now be found by BFS,using an algorithm named `SubsetConflictWalk` (Algorithm 3).

Algorithm `SubsetConflictWalk`($\operatorname{conv} Q, \operatorname{conv} P$)

1. Let `queue` be a queue with the elements in Q.
2. While `queue` $\neq \emptyset$.
   (a) Let p be the next point in `queue`.
   (b) If $p \notin Q$, insert p into $\operatorname{conv} Q$, 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 $\operatorname{conv}(Q \cup p)$, using Claim 3.3.
   (d) Using the $\tilde{f}_q$'s, find conflict facets $f_q \in F[Q]$ for $\Gamma_P(p)$. If $q \in \Gamma_P(p)$ has not been encountered yet, insert it into `queue`.

Algorithm 3: Determining the conflict facets in a subset.

Step 2 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 2b takes $O(\deg_{S\cup p} p)$ time [17, Chapter 11.2]. Furthermore, by Claim 3.3, the conflict facets of $\Gamma_P(p)$ can be found in $O(\deg_{S\cup p} p + \deg_P p)$ time. Finally, Step 2d 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 $\operatorname{conv} S$, it conflicts with at least one facet adjacent to e. Thus, f_q can be computed from $\tilde{f}_q$ in constant time. The total running time of `SubsetConflictWalk` is proportional to $\mathbf{E}\left[\sum_{p\in P}\left(\deg_{S\cup p} p + \deg_P p\right)\right]$. Now, since[4] $\deg_{S\cup p} p \ll d_p$ for $p \notin S$, where d_p is the conflict size of p in $\operatorname{conv} S$, 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 (5). The lemma follows, since we have $\mathbf{E}\left[\sum_{f\in F[S]} b_f\right] \ll n$ by Lemma B.2(6) (b_f is the conflict size of f). □

Now we consider Step 3. 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 [9, Lemma A.1], $\mathbf{E}[a_f^2] \ll (b_f/\chi)^2$. By Lemma B.3, Step 3a takes time $\mathbf{E}\left[\sum_{f\in F[S]} a_f \log a_f\right]$, and by Lemma B.2(6), we get

$$\mathbf{E}\left[\sum_{f\in F[S]} a_f \log a_f\right] \ll \mathbf{E}\left[\sum_{f\in F[S]} a_f^2\right] \ll \mathbf{E}\left[\frac{1}{\chi^2}\sum_{f\in F[S]} b_f^2\right] \ll \frac{\chi n}{\chi^2} = \frac{n}{\chi}.$$

Using `SplitHull` in Step 3b, $\operatorname{conv} C_i$ can now be computed in $O(|C_i| + n/\chi)$. There are χ colors, so Step 3 takes $O(n)$ time, and Theorem 3.1 follows.

4. 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 large χ, this can be improved.

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 an arbitrary coloring of P. Given $\operatorname{conv} P$, we can compute $\operatorname{conv} C_1, \ldots, \operatorname{conv} C_\chi$ in $O\left(n\sqrt{\log n}\right)$ expected time.*

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

As in `RandMultiSplit`, we use random sampling for geometric divide-and-conquer. For this, however, we need to avoid small color classes. Thus, the algorithm first computes the convex hull of every C_i with $|C_i| \leq 2^{\sqrt{\log n}}$ in time $O(|C_i| \log |C_i|)$ [17, Chapter 11]. Let K denote the remaining colors, and let $Q = \bigcup_{i \in K} C_i$, $n_1 = |Q|$ and $n_2 = n - n_1$. We begin with a useful sampling lemma.

LEMMA 4.2. *Let $Q \subseteq \mathbb{R}^3$ be an m-point set in general convex position, and let $\alpha \in \{1, \ldots, m\}$. Given* 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. *$|S| = \alpha$, $|R| = \Omega(m)$, $\max_i |R_i| = O\left(\frac{m}{\alpha} \log \alpha\right)$.*
2. *For $i = 1, \ldots, \beta$, 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* 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 conv S, conv $R_1, \ldots,$ conv R_β, 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 4.3. *A decent subset $S \subseteq Q$ of size α together with* 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, first note that by Lemma B.2(6) and Markov's inequality, we have $\sum_{f \in F[S]} b_f \ll m$ with probability at least $3/4$, and by Lemma B.1 and Markov's inequality, we also have $\max_{f \in F[S]} b_f \ll m(\log \alpha)/\alpha$ with probability at least $3/4$. Thus, the claimed probability follows from a union bound.

Furthermore, a decent sample can be verified in $O(m)$ time: by the proof of Lemma 3.2, we can find the conflict sets B_f and D_p in time $O\left(m + \sum_{f \in F[S]} b_f\right)$. Hence, if the number of steps exceeds a certain threshold of $O(m)$, we reject the sample. 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. □

Now let S be a decent sample, and let B_f, $f \in F[S]$, denote its conflict sets. By (5) and Property (i) of a decent sample, we have $\sum_{p \in P} d_p = O(m)$, and hence there exists a constant λ such that the set $X = \{p \in Q \mid d_p \geq \lambda\}$ has cardinality at most, say, $m/100$. Let $R' = Q \setminus X$, $B'_f = B_f \cap R'$ 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 Figure 4. The pruning is done by a greedy algorithm `PruneFS` (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 $R = \bigcup_{i=1}^{\beta} R_i$.

Algorithm `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 $N_f = \{f' \in F[S] \mid \delta(f, f') \leq 2\lambda\}$.
   (b) Let `queue` = `queue` $\setminus N_f$ and $\mathcal{F} = \mathcal{F} \cup \{f\}$.

Algorithm 4: Pruning the conflict facets.

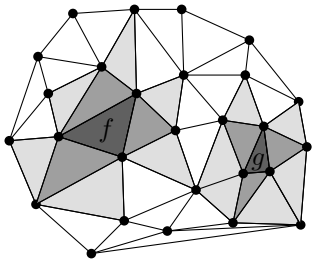

Figure 4: The pruning step: remove all facets at distance at most 2λ from a facet with maximum conflict size. The points in B'_f, B'_g conflict only with the darker facets at distance at most $\lambda = 1$.

CLAIM 4.4. *We have $|R| = \Omega(m)$, 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 is shares an edge with a facet in D_q.*

PROOF. To see that $|R| = \Omega(m)$, note that $|N_f| = O(1)$ 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

$$|R| = \sum_{f \in \mathcal{F}} b'_f \gg \sum_{f \in \mathcal{F}} \sum_{f' \in N_f} b'_{f'} \geq |R'| \gg m.$$

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. Since any point $p \in R$ has $|D_p| \leq c$, and since D_p is connected in the dual graph of conv S, it follows that $R_i \cap R_j = \emptyset$, since R_i, R_j are the conflict sets of f_i, f_j and since by construction $\delta(f_i, f_j) > 2\lambda$. For the same reason, 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 with a facet in D_q. □

By now, we have established Properties 1–4. It remains to show how to find all the convex hulls quickly. First, using `SplitHull`, we can compute conv S, conv $(R \cup S)$ and conv $(Q \setminus (R \cup S))$ in time $O(n_1)$. It remains to consider the R_i.

CLAIM 4.5. *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 $R_i = R \cap h_f^+$. Let Γ_i denote the points in $(R \cup S) \setminus R_i$ that are adjacent 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 4.

Next, we claim that $|\Gamma| = O(1)$: 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 $|\bigcup_{p \in R_i} D_p| = O(1)$ 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 half-space 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 4.2. □

We use Lemma 4.2 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. Recursively, we split both conv $(Q \setminus (S \cup R))$ and conv R_j for $j = 1, \ldots, \beta$ to obtain the hulls conv $(C_i \cap (Q \setminus (S \cup R)))$ as well as conv $(C_i \cap R_j)$. To get conv C_i, we then merge all the hulls conv $(C_i \cap R_j)$ into conv S, using an algorithm to combine 3-polytopes separated by a plane [3, Chapter 9.3]. For each $j \in \{1, \ldots, \beta\}$, this takes time $O(1 + |C_i \cap R_j|)$, since all new edges are incident to constantly many points in S. Then, we extract conv $(C_i \cap (S \cup R))$ in expected time $O(|S| + |C_i \cap (S \cup R)|)$ via `SplitHull` and compute the union of conv $(C_i \cap (S \cup R))$ with conv $(C_i \cap (Q \setminus (S \cup R)))$ in time $O(|C_i|)$ [8]. The total expected time 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 small color classes at the beginning, the second term counts the remaining steps. We get the following recursion for the running time:

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

Using $|R| \gg n_1$ and $\max_{1 \leq j \leq \beta} |R_j| \ll n_1\sqrt{\log n}/2^{\sqrt{\log n}}$, we obtain $T(n) \ll n\sqrt{\log n}$, as desired.

5. POINTS IN HALFSPACES

The problem now is to preprocess a point set to report quickly all the points contained in a query halfspace. However, we want to find not only the points, but also their convex hull. We base our approach on a data structure by Chan [4] that uses *filtering search* [6]: first, it obtains a superset of the result with comparable size (the *candidate set*), and then examines each point 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 via `SplitHull`. We also show how to improve the preprocessing time over the naive $O(n \log^2 n)$.

THEOREM 5.1. *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 $P \cap h^+$ (ie, $G(h)$ has vertex set $P \cap h^+$ and contains all edges of conv P with both endpoints in h^+). Here are some standard facts about $G(h)$ (eg, [8]).

LEMMA 5.2. *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[5] L along the edges of 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 $O(|V[G(h)]|)$.* □

The walk L is called the *lace* of $G(h)$, see Figure 5a. Knowing $G(h)$ is enough to compute conv $(P \cap h^+)$ quickly.

COROLLARY 5.3. *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 Figure 5b. 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 5.2, 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 h a sequence E of line segments 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 $\mathcal{C}$.

The goal now is to construct a convex polytope $\mathcal{P}$ of complexity $O(|P \cap h^+|)$ whose vertex set contains $P \cap h^+$. 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 $G_P(h)$; (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.[6] Furthermore, for each

[5] 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).

[6] That is, $\tilde{e}$ is the edge incident to the facet whose intersections with h create e.

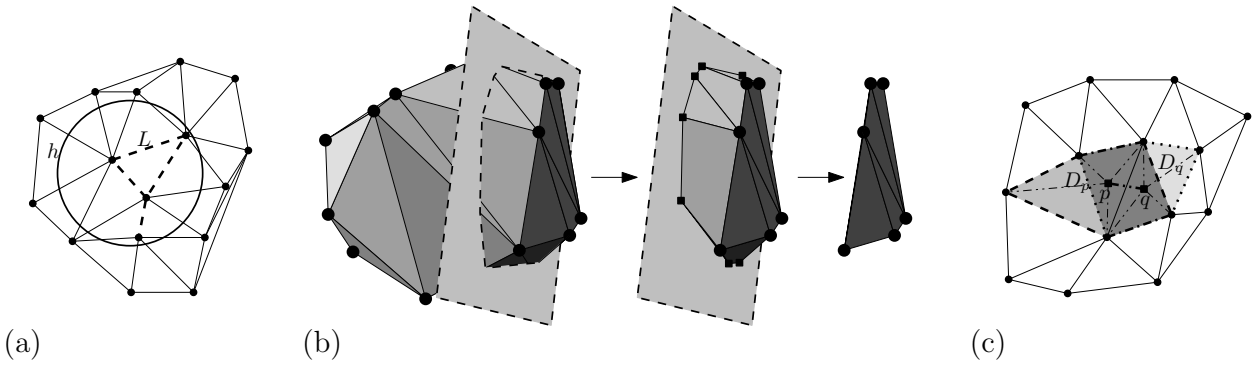

Figure 5: The halfspace range reporting algorithm: (a) A lace: h^+ corresponds to the inside of the circle. The lace L is shown as a dashed line. (b) The three stages of Corollary 5.3: Given $G(h)$, find an intermediate polytope that contains the result, and split it. (c) Finding the conflict facets for an edge. $D_{\{p,q\}}$ is darkest, while D_p, D_q are lighter. D_p is bounded by a dashed line, D_q by a dotted line.

$e \in E[\mathcal{C}] \setminus E$, 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 into h^-. Pick a point $q \in Z \cap \operatorname{conv} P$ 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 $\operatorname{conv}(P \cap h^+)$ in time $O(|V[\mathcal{P}']|) = O(|P \cap h^+|)$ using `SplitHull`. □

For Corollary 5.3, 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 $\operatorname{conv} S$).

LEMMA 5.4. *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 B.2 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 conflict facets of p and q. 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 $\operatorname{conv} S$ that is removed in $\operatorname{conv}(S \cup \{p, q\})$. Since T_e is connected[7], 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 2d 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 $\operatorname{conv} S$ 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 $\operatorname{conv} S$ 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 $f_{\{p,q\}} = \perp$. This takes constant time, and therefore the running time of the algorithm is linear, as we saw in the proof of Lemma 3.2. 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 $\operatorname{conv}(S \cup \{p, q\})$. 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 $T|_{D_q}$ 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 compute a DCEL representation of the graphs $G(h_f)$ in $O(n)$ time through careful pointer manipulation (Algorithm 5). □

Algorithm `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.

Algorithm 5: Computing the subgraphs.

PROOF OF THEOREM 5.1. We rely on a variant of Chan's data structure [4] due to Ramos [24]. The candidate sets are the conflict sets of an appropriate gradation of P. By Corollary 5.3 and Lemma 5.4, 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.

[7] We define the empty graph to be connected

The details are as follows: take a *gradation* $\emptyset = P_{-1} \subseteq P_0 \subseteq \cdots \subseteq P_{\log n} = P$ of P, where P_{i-1} is derived from P_i by sampling every point with probability $1/2$. We compute the convex hulls $\operatorname{conv} P_i$ in time $O(n \log n)$. Using Lemma 5.4 and Corollary 5.3, we then find the convex hulls $\operatorname{conv} B_f$ 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 [23, Chapter 2.4.1]. We compute two data structures, one for the upper envelope and one for the lower envelope, focussing 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 $\operatorname{conv} P_i$). 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 $\mathcal{H}_i$, corresponding to at most three facets f_1, f_2, f_3 of $\operatorname{conv} P_i$. Let $B_\Delta = B_{f_1} \cup B_{f_2} \cup B_{f_3}$. We compute $\operatorname{conv} B_\Delta$ in linear time [8] and store it with Δ. By the properties of canonical triangulations and the arguments given by Chan [4], 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 `SplitHull` to compute $\operatorname{conv}(P \cap h^+)$ in $O(|B_{\Delta_i}|)$ time. As was argued by Ramos [24, Section 2.2.1] such a query takes expected time $O(\log n + |P \cap h^+|)$, as claimed. □

6. UNION OF HULLS

Finally, we consider the problem 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 will look at the following problem DISJUNION: given point sets $P_1, \ldots, P_k \subseteq \mathbb{R}^3$ and their convex hulls $\operatorname{conv} P_1, \ldots, \operatorname{conv} P_k$ such that $\operatorname{conv} P_i \cap \operatorname{conv} P_j = \emptyset$ for $i \neq j$ and such that $P = \bigcup_{i=1}^k P_i$ is in convex position, we would like to compute $\operatorname{conv} P$. In general, we cannot do better than to repeatedly merge pairs of the hulls.

THEOREM 6.1. *Any algorithm that solves* DISJUNION *requires* $\Omega(|P| \log k)$ *comparisons.*

PROOF. We use an old lower bound [8, Section 4A] and combine it with Seidel's method of including the index as a coordinate [25]. 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. 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 Figure 6.

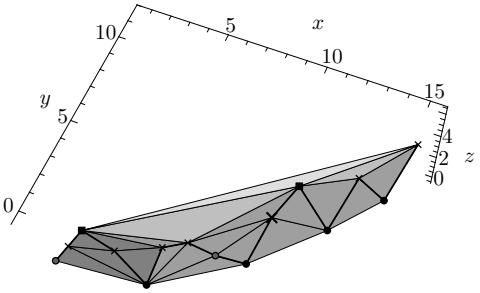

Figure 6: The lower bound for $(5, 9, 12, 14)$, $(1, 8)$, $(2, 4, 6, 7, 10, 13)$, and $(3, 11)$. Edges between consecutive elements are bold.

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 $\operatorname{conv} P_i \cap \operatorname{conv} P_j = \emptyset$ for $i \neq j$. The $\operatorname{conv} P_i$ can be computed in linear time, since the lists L_i are sorted. Now, if r, s are consecutive in the sorted list L, then $p(r)p(s)$ is an edge of $\operatorname{conv} P$: let $\hat{p}(r)$, $\hat{p}(s)$ denote the projection 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 $\operatorname{conv} P$. Consequently, it takes $\Omega(|P| \log k)$ time to compute $\operatorname{conv} P$, since otherwise we could recover the sorted list L by examining the $O(|P|)$ edges of $\operatorname{conv} P$. □

Intuitively, what makes our lower bound instance hard is the fact that we need to switch often between the $\operatorname{conv} P_i$. By imposing additional constraints to avoid this, we can do better.

THEOREM 6.2. *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 $\operatorname{conv} Q|_{P_i}$ are connected. Then, given spanning trees $T_1, \ldots, T_k$ for $\operatorname{conv} Q|_{P_i}$, we can compute $\operatorname{conv} P$ in expected time $O(n \log^* n + k \log k)$.*

PROOF. We follow Seidel's strategy for polygon triangulation [26]: we 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 we compute $\operatorname{conv}(S_0 \cup K)$ in time $O(n + k \log k)$ and successively each $\operatorname{conv}(S_i \cup K)$ in $O(n)$. Here, the bottleneck is to locate the conflict facets for S_{i+1} in $\operatorname{conv}(S_i \cup K)$. This is done using the spanning trees T_i and an appropriate variant of `Subset-ConflictWalk`.

We may assume that $k < n/2$, since otherwise the theorem is easy. Let $K \subseteq P$ 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 i-th iterated logarithm[8] 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 $\operatorname{conv} \widetilde{S}_{i+1}$ from $\operatorname{conv} \widetilde{S}_i$ in time $O(n)$ for each i. Furthermore, $\operatorname{conv} \widetilde{S}_0$ 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 $\operatorname{conv} Q = \operatorname{conv} \widetilde{S}_\beta$.

To derive $\operatorname{conv} \widetilde{S}_{i+1}$ from $\operatorname{conv} \widetilde{S}_i$ 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 B.3 to compute $\operatorname{conv} \widetilde{S}_{i+1}$. This takes time proportional to

$$\begin{aligned}
&\left(|S_{i+1}| + k\frac{|S_{i+1}|}{|S_i|}\right) \log \frac{|S_{i+1}|}{|S_i|} \\
&\le 2|S_{i+1}| \log \left(\frac{\log^{(\alpha+i)} m}{\log^{(\alpha+i+1)} m}\right) \\
&\ll \frac{m}{\log^{(\alpha+i+1)} m} \log \left(\frac{\log^{(\alpha+i)} m}{\log^{(\alpha+i+1)} m}\right),
\end{aligned}$$

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 variant of `SubsetConflictWalk`: merge the neighbors of r_j in $\operatorname{conv} \widetilde{S}_i$ 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 $\operatorname{conv} \widetilde{S}_i$, and so on. As in Section 3, we see that the total time is proportional to

$$\begin{aligned}
&\sum_{p \in \widetilde{S}_i} \deg_{\widetilde{S}_i} p + \sum_{j=1}^{k} \sum_{p \in T_j} \deg_{T_j} p + \sum_{p \in P \setminus S_i} d_p \\
&\ll |\widetilde{S}_i| + |P \setminus \widetilde{S}_i| + \sum_{f \in F[\widetilde{S}_i]} b_f \\
&\ll |P| + n - k + k\frac{|S_{i+1}|}{|S_i|},
\end{aligned}$$

by Lemma B.2. Since $k \le |S_i|$ and $|S_{i+i}| \le n$, the last term is linear. This finishes the proof. □

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

[8] We set $\log^{(0)} m = m$ and $\log^{(\log^* m + 1)} m = 1$.

In particular, we get the following nice fact about Delaunay triangulations.

COROLLARY 6.4. *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 $O(n \log^* n + k \log k)$.* □

7. REFERENCES

[1] A. Aggarwal, L. J. Guibas, J. Saxe, and P. W. Shor. A linear-time algorithm for computing the Voronoi diagram of a convex polygon. *Discrete Comput. Geom.*, 4(6):591–604, 1989.

[2] N. M. Amato, M. T. Goodrich, and E. A. Ramos. Linear-time triangulation of a simple polygon made easier via randomization. In *SCG '00: Proceedings of the sixteenth annual symposium on Computational geometry*, pages 201–212, New York, NY, USA, 2000. ACM.

[3] J.-D. Boissonnat and M. Yvinec. *Algorithmic geometry*. Cambridge University Press, New York, NY, USA, 1998.

[4] T. M. Chan. Random sampling, halfspace range reporting, and construction of $(\le k)$-levels in three dimensions. *SIAM J. Comput.*, 30(2):561–575, 2000.

[5] T. M. Chan. Three problems about simple polygons. *Comput. Geom.*, 35(3):209–217, 2006.

[6] B. Chazelle. Filtering search: a new approach to query-answering. *SIAM J. Comput.*, 15(3):703–724, 1986.

[7] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, 6(5):485–524, 1991.

[8] B. Chazelle. An optimal algorithm for intersecting three-dimensional convex polyhedra. *SIAM J. Comput.*, 21(4):671–696, 1992.

[9] B. Chazelle. *The discrepancy method: randomness and complexity*. Cambridge University Press, New York, NY, USA, 2000.

[10] B. Chazelle, O. Devillers, F. Hurtado, M. Mora, V. Sacristán, and M. Teillaud. Splitting a Delaunay triangulation in linear time. *Algorithmica*, 34(1):39–46, 2002.

[11] L. P. Chew. Building Voronoi Diagrams for Convex Polygons in Linear Expected Time. Technical Report PCS-TR90-147, Dartmouth College, Computer Science, Hanover, NH, 1990.

[12] L. P. Chew and S. Fortune. Sorting helps for Voronoi diagrams. *Algorithmica*, 18(2):217–228, 1997.

[13] F. Chin, J. Snoeyink, and C. A. Wang. Finding the medial axis of a simple polygon in linear time. *Discrete Comput. Geom.*, 21(3):405–420, 1999.

[14] F. Chin and C. A. Wang. Finding the constrained Delaunay triangulation and constrained Voronoi diagram of a simple polygon in linear time. *SIAM Journal on Computing*, 28(2):471–486, 1998.

[15] K. L. Clarkson. A randomized algorithm for closest-point queries. *SIAM Journal on Computing*, 17(4):830–847, 1988.

[16] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry. II. *Discrete Comput. Geom.*, 4(5):387–421, 1989.

[17] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational geometry*. Springer-Verlag, Berlin, revised edition, 2000. Algorithms and applications.

[18] O. Devillers. Randomization yields simple $O(n \log^* n)$ algorithms for difficult $\Omega(n)$ problems. *Internat. J. Comput. Geom. Appl.*, 2(1):97–111, 1992.

[19] H. N. Djidjev and A. Lingas. On computing Voronoi diagrams for sorted point sets. *Internat. J. Comput. Geom. Appl.*, 5(3):327–337, 1995.

[20] H. Fournier and A. Vigneron. A tight lower bound for computing the diameter of a 3D convex polytope. *Algorithmica*, 49(3):245–257, 2007.

[21] R. Klein and A. Lingas. A linear-time randomized algorithm for the bounded Voronoi diagram of a simple polygon. *Int. J. Comput. Geometry Appl.*, 6(3):263–278, 1996.

[22] J. Matoušek. *Lectures on discrete geometry*, volume 212 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2002.

[23] K. Mulmuley. *Computational Geometry: An Introduction through Randomized Algorithms*. Prentice-Hall, Englewood Cliffs, 1994.

[24] E. A. Ramos. On range reporting, ray shooting and k-level construction. In *SCG '99: Proceedings of the fifteenth annual symposium on Computational geometry*, pages 390–399, New York, NY, USA, 1999. ACM.

[25] R. Seidel. A method for proving lower bounds for certain geometric problems. Technical Report TR84-592, Cornell University, Ithaca, NY, USA, 1984.

[26] R. Seidel. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Comput. Geom.*, 1(1):51–64, 1991.

[27] M. J. van Kreveld, M. Löffler, and J. S. B. Mitchell. Preprocessing imprecise points and splitting triangulations. In S.-H. Hong, H. Nagamochi, and T. Fukunaga, editors, *ISAAC*, volume 5369 of *Lecture Notes in Computer Science*, pages 544–555. Springer, 2008.

APPENDIX

A. DEFINITIONS AND NOTATION

Given a finite point set $P \subseteq \mathbb{R}^3$, let $\operatorname{conv} P$ denote the convex hull of P. We denote the edges and facets of $\operatorname{conv} P$ 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. Throughout, we will assume that convex hulls are given in a standard planar graph representation, eg a DCEL [17, Chapter 2.2]. Our point sets will usually be in *general convex position*, ie, every three points in P are linearly independent and $p \notin \operatorname{conv}(P \setminus p)$ for every $p \in P$. In particular, $\operatorname{conv} P$ is simplicial and all the points in P are vertices of $\operatorname{conv} P$.

We use classical geometric random sampling [16, 23]; see Appendix B. We quickly review the concept of *conflict sets*. Given a point set $P \subseteq \mathbb{R}^3$, an edge $e \in E[P]$, and a point $p \notin \operatorname{conv} P$, we say that p *can see* e in $\operatorname{conv} P$ or that e *is visible* from p, if the triangle spanned by e and p intersects $\operatorname{conv} P$ 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 $\operatorname{conv} P$ 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 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,

$$\sum_{f \in F[Q]} b_f = \sum_{p \in P} d_p. \tag{5}$$

B. CLARKSON-SHOR TOOLBOX

We review a few tools from geometric random sampling theory [16,23]. First, we have a Chernoff-type bound for the conflict size of a random sample.

LEMMA B.1. *Fix $0 \le p \le 1$ and let $S \subseteq P$ be a random subset of size pn. Fix $t \ge 2$ and let $F_{\ge t} = \{f \in F[S] \mid b_f \ge t/p\}$. Then*

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

□

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 must be included in the sample.

LEMMA B.2. *Fix $0 \le p \le 1$ and let $S \subseteq P \setminus K$ be a random subset of size $p(n-k)$. Let g be a function such that $g(tn) \ll \exp(t) g(n)$ for all $t \ge 0$. Then*

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

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

$$\mathbf{E}\left[\sum_{f \in F[S]} b_f^\gamma\right] \ll np^{1-\gamma}, \tag{6}$$

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

$$\mathbf{E}\left[\sum_{f \in F[S \cup K]} b_f \log b_f\right] \ll \left((n-k) + \frac{k}{p}\right) \log\left(\frac{1}{p}\right). \tag{7}$$

□

The following lemma is a standard application of geometric divide-and-conquer [9, 15, 16] and asserts that a convex hull can be computed faster if a random partial hull and the corresponding conflict information are known.

LEMMA B.3. *Fix $0 \le p \le 1$ and let $S \subseteq P \setminus K$ be a subset of size $p(n-k)$. Suppose that $\operatorname{conv}(S \cup K)$ and the conflict sets $B_f \subseteq P$ for $f \in F[S \cup K]$ are available. Then we can find $\operatorname{conv} P$ 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 $O((n - k + k/p) \log(1/p))$.* □