Convex Hull of Planar H-Polyhedra

Axel Simon and Andy King
Computing Laboratory
University of Kent, CT2 7NF, UK
{a.simon,a.m.king}@ukc.ac.uk

December 8, 2003

Abstract

Suppose $\langle A_i, \vec{c}_i \rangle$ are planar (convex) H-polyhedra, that is, $A_i \in \mathbb{R}^{n_i \times 2}$ and $\vec{c}_i \in \mathbb{R}^{n_i}$. Let $P_i = \{\vec{x} \in \mathbb{R}^2 \mid A_i\vec{x} \leq \vec{c}_i\}$ and $n = n_1 + n_2$. We present an $O(n \log n)$ algorithm for calculating an H-polyhedron $\langle A, \vec{c} \rangle$ with the smallest $P = \{\vec{x} \in \mathbb{R}^2 \mid A\vec{x} \leq \vec{c}\}$ such that $P_1 \cup P_2 \subseteq P$.

Keywords: convex hull, computational geometry
C. R. Categories: I.3.5 [Computational Geometry and Object Modeling]: Boundary representations; Geometric algorithms, languages, and systems, I.3.6 [Methodology and Techniques]: Graphics data structures and data types, F.3.1 [Specifying and Verifying and Reasoning about Programs]: Invariants, Mechanical verification.

1 Introduction

The convex hull problem is classically stated as the problem of computing the minimal convex region that contains n distinct points $\{\langle x_i, y_i \rangle\}_{i=1}^n$ in the Euclidean plane $\mathbb{R}^2$. The seminal work of Graham [4] showed that the convex hull problem can be solved in $O(n \log n)$ worse-case running time. It inspired many to elaborate on, for example, the three and more dimensional case, specialised algorithms for polygons, on-line variants, *etc.* [8, 10]. The convex hull of polytopes (bounded polyhedra) can be calculated straightforwardly by taking the convex hull of their extreme points. However, calculating the convex hull for polyhedra turns out to be more subtle due to a large number of geometric configurations. Even for planar polyhedra, the introduction of rays makes it necessary to handle polyhedra such as a single half-space, a single ray, a single line, two facing (not coinciding) half-spaces, etc., all of which require special handling in a point-based algorithm. The problem is exacerbated by the number of ways these special polyhedra can be combined. In contrast, we present a direct reduction of the convex hull problem of planar polyhedra to the convex hull problem for a set of points [4]. By confining all input points to a box

and applying the rays to translate these points outside the box, a linear pass around the convex hull of all these points is sufficient to determine the resulting polyhedron. By adopting the classic Graham scan algorithm, our algorithm inherits its $O(n \log n)$ time complexity. The standard tactic for calculating the convex hull of H-polyhedra is to convert the input into an intermediate ray and vertex representation. Two common approaches to this conversion problem are the double description method [7] (also known as the Chernikova algorithm [3]) and the vertex enumeration algorithm of Avis and Fukuda [2]. The Chernikova method leads to a cubic time solution for calculating the convex hull of planar H-polyhedra [6] whereas the Avis and Fukuda approach runs in quadratic time.

The remaining sections are organised as follows: A self-contained overview of the algorithm, together with a worked example, is given in the next Section. A formal proof of correctness is given in Section 3. Section 4 concludes.

2 Planar Convex Hull Algorithm

The planar convex hull algorithm takes as input two H-polyhedra and outputs the smallest H-polyhedron which includes the input. The H-representation of a planar polyhedron corresponds to a set of inequalities each of which takes the form $ax + by \leq c$, where $a, b, c \in \mathbb{R}$ and either $a \neq 0$ or $b \neq 0$. Let Lin denote the set of all such inequalities. The vector $\langle a, b \rangle$ is orthogonal to the boundary of the halfspace induced by $ax + by \leq c$ and points away from the feasible space. This vector induces an ordering on halfspaces via the orientation mapping θ. This map $\theta : Lin \rightarrow [0, 2\pi)$ is defined such that $\theta(ax + by \leq c) = \psi$ where $\cos(\psi) = a/\sqrt{a^2 + b^2}$ and $\sin(\psi) = b/\sqrt{a^2 + b^2}$. The mapping θ corresponds to the counter-clockwise angle which the half-space of $x \leq 0$ has to be turned through to coincide with that of $ax + by \leq c$. Sorting half-spaces by angle is the key to efficiency in our algorithm. However, θ is only used for comparing the orientation of two half-spaces. To aid the explanation of the algorithm, the concept of angular difference $e_1 \measuredangle e_2$ between two inequalities e_1 and e_2 is introduced as the counter-clockwise angle between $\theta(e_1)$ and $\theta(e_2)$. More precisely $e_1 \measuredangle e_2 = (\theta(e_2) - \theta(e_1))\ mod\ 2\pi$. Note that this comparator can be realized without recourse to trigonometric functions [9].

The algorithm makes use of a number of simple auxiliary functions. The function $intersect(a_1x + b_1y \leq c_1,\ a_2x + b_2y \leq c_2)$ calculates the set of intersection points of the two lines $a_1x + b_1y = c_1$ and $a_2x + b_2y = c_2$. In practice an implementation of this function only needs to be partial since it is only applied in the algorithm when the result set contains a single point. The remaining auxiliaries are listed in Figure 1. The *connect* function generates an inequality from two points subject to the following constraints: the halfspace induced by $connect(p_1, p_2)$ has p_1 and p_2 on its boundary and if p_1, p_2, p_3 are ordered counter-clockwise then p_3 is in the feasible space. The notation $\overline{p_1, p_2}$ is used to abbreviate $connect(p_1, p_2)$. Furthermore, the predicate $saturates(p, e)$ holds whenever the point p is on the boundary of the halfspace defined by the inequality e. Finally, the predicate $inBox(s, p)$ determines whether the point p occurs

```
1  function extreme(E) begin
2      sort E to obtain e_0 ..., e_{n-1} such that θ(e_0) < θ(e_1) < ... < θ(e_{n-1});
3      V := R := ∅;
4      if E = {ax + by ≤ c} then R := {⟨−a/√(a²+b²), −b/√(a²+b²)⟩};
5      for i ∈ [0, n − 1] do let e_i ≡ ax + by ≤ c in begin
6          // are the intersection points of this inequality degenerated?
7          d_pre := (θ(e_i) − θ(e_{i−1 mod n})) mod 2π ≥ π ∨ n = 1;
8          d_post := (θ(e_{i+1 mod n}) − θ(e_i)) mod 2π ≥ π ∨ n = 1;
9          if d_pre then R := R ∪ {⟨b/√(a²+b²), −a/√(a²+b²)⟩};
10         if d_post then R := R ∪ {⟨−b/√(a²+b²), a/√(a²+b²)⟩};
11                 else V := V ∪ intersect(e_i, e_{(i+1) mod n});
12         if d_pre ∧ d_post then V := V ∪ {v} where v ∈ {⟨x, y⟩ | ax + by = c}
13     end
14     return ⟨V, R⟩
15 end
16
17 function connect(⟨x_1, y_1⟩, ⟨x_2, y_2⟩)
18     return (y_2 − y_1)x + (x_1 − x_2)y ≤ (y_2 − y_1)x_1 + (x_1 − x_2)y_1
19
20 function saturates(⟨x_1, y_1⟩, ax + by ≤ c)
21     return (ax_1 + by_1 = c)
22
23 function inBox(s, ⟨x, y⟩)
24     return |x| < s ∧ |y| < s
```

Figure 1: Convex hull algorithm for planar polyhedra

within a square of width $2s$ that is centred on the origin.

The algorithm divides into a decomposition and a reconstruction phase. The *hull* function decomposes the input polyhedra into their corresponding ray and vertex representations by calling the function *extreme* in lines 3 and 4. The remainder of the *hull* function reconstructs a set of inequalities whose halfspaces enclose both sets of rays and points. The functions *extreme* and *hull* are presented in Figures 1 and 2, respectively. The algorithm requires the input polyhedra to be non-redundant. This means that no proper subset of the inequalities induces the same space as the original set of inequalities. The algorithm itself produces a non-redundant system.

To illustrate the algorithm consider Figure 3. The polyhedron $E = \{e_0, e_1, e_2\}$ and the polytope $E' = \{e'_0, \ldots, e'_5\}$ constitute the input to the *hull* function. They are passed to the function *extreme* at line 28 and 29. Within *extreme* the inequalities of each polyhedron are sorted at line 2. Note that for ease of presentation the indices coincide with the angular ordering. The loop at lines 5–13 examines the relationship of each inequality with its two angular neighbours. If d_{post} is false, the intersection point $intersect(e_i, e_{(i+1) \bmod n})$ is a vertex which is

```
25 function hull(E_1, E_2) begin
26     // assertion: each E_i is satisfiable and non-redundant
27     if E_1 = ∅ ∨ E_2 = ∅ then return ∅;
28     ⟨P_1, R_1⟩ := extreme(E_1);
29     ⟨P_2, R_2⟩ := extreme(E_2);
30     P := P_1 ∪ P_2;
31     R := R_1 ∪ R_2; // Note: |R| ≤ 8
32     s := max{|x|, |y| | ⟨x, y⟩ ∈ P} + 1;
33     // add a point along the ray, that goes through x, y
34     // and is outside the box
35     Q := P;
36     for ⟨x, y, a, b⟩ ∈ P × R do Q := Q ∪ {⟨x + 2√2sa, y + 2√2sb⟩};
37     // construct four inequalities in the zero dimensional case
38     if Q = {⟨x_1, y_1⟩} then return {x ≤ x_1, y ≤ y_1, −x ≤ −x_1, −y ≤ −y_1};
39     // the centre of gravity q_p is feasible but not a vertex (since |Q| > 1)
40     q_p := ⟨∑_{⟨x,y⟩∈Q} x/|Q|, ∑_{⟨x,y⟩∈Q} y/|Q|⟩;
41     // q_p is pivot point for sorting: ∀i ∈ [0, n−2] . θ(q_p, q_i) ≤ θ(q_p, q_{i+1})
42     ⟨q_0, ..., q_{n−1}⟩ := sort(q_p, Q)
43     // identify the m vertices q_{k_i} where 0 ≤ k_0 < ... < k_{m−1} < n
44     ⟨q_{k_0}, ..., q_{k_{m−1}}⟩ := scan(⟨q_0, ..., q_{n−1}⟩)
45     E_res := ∅;
46     for i ∈ [0, m − 1] do begin
47         let ⟨x_1, y_1⟩ = q_{k_i}, ⟨x_2, y_2⟩ = q_{k_{(i+1) mod m}}
48         let e = connect(⟨x_1, y_1⟩, ⟨x_2, y_2⟩)
49         // add e to E_res if q_{k_i} or q_{k_{(i+1) mod m}} is in the box...
50         add := inBox(s, ⟨x_1, y_1⟩) ∨ inBox(s, ⟨x_2, y_2⟩) ∨ m = 2;
51         j := (k_i + 1) mod n;
52         while ¬add ∧ j ≠ k_{i+1} do begin
53             // ...or any boundary point is in the box
54             add := saturates(q_j, e) ∧ inBox(s, q_j);
55             j := (j + 1) mod n;
56         end;
57         if m = 2 ∧ inBox(s, ⟨x_1, y_1⟩) then
58             if y_1 = y_2 then E_res := E_res ∪ {sgn(x_1 − x_2)x ≤ sgn(x_1 − x_2)x_1}
59                          else E_res := E_res ∪ {sgn(y_1 − y_2)y ≤ sgn(y_1 − y_2)y_1}
60         if add then E_res := E_res ∪ {e};
61         end
62     end
63     return E_res
64 end
```

Figure 2: Convex hull algorithm for planar polyhedra

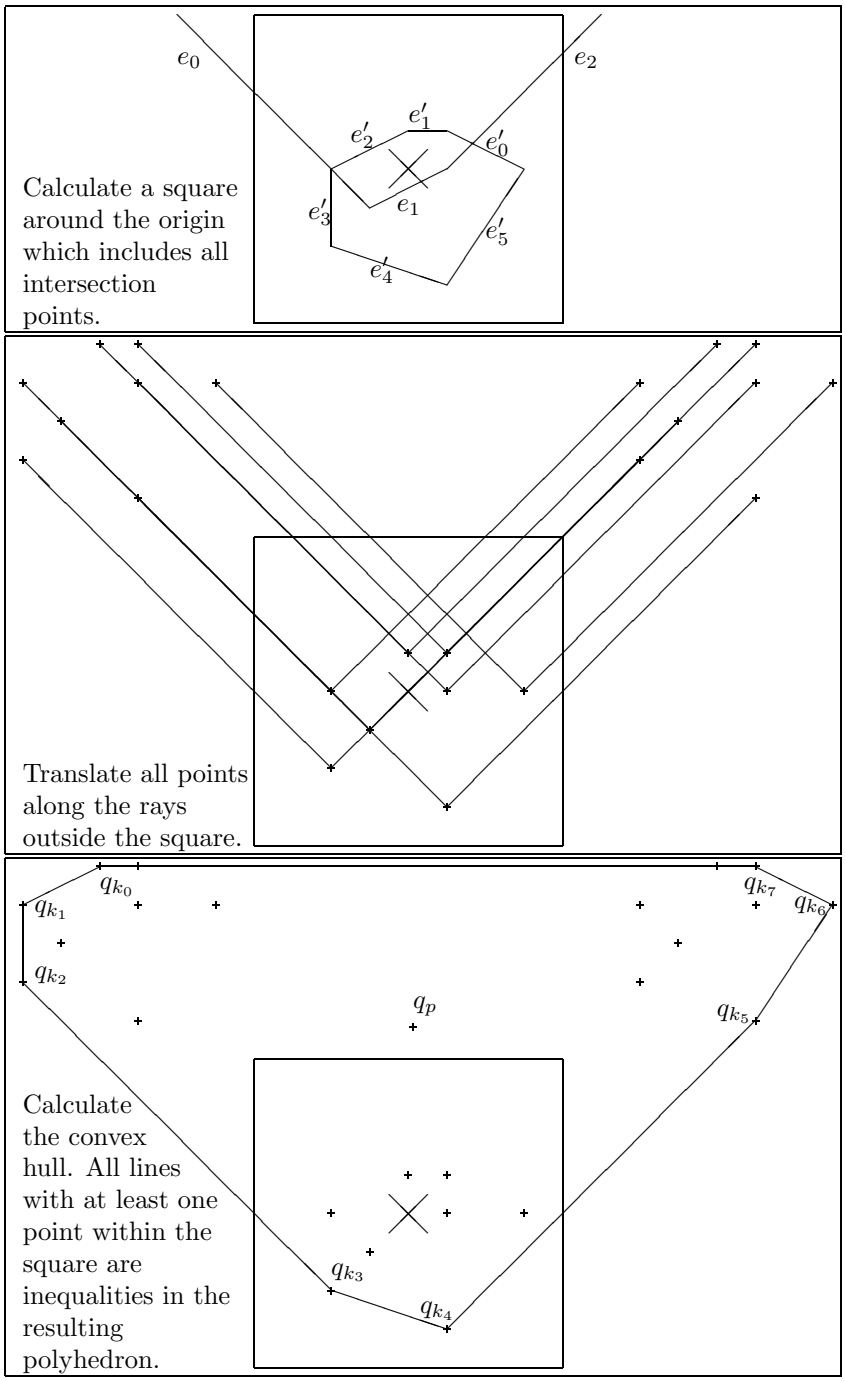

Figure 3: The different stages of the polyhedra convex hull algorithm.

added at line 11. Conversely, if d_{post} is true, the intersection point is degenerate, that is, either E contains a single inequality or the angular difference between the current inequality and its successor is greater or equal to π. In the example two vertices are created for E, namely v_1 and v_2 where $\{v_1\} = intersect(e_0, e_1)$ and $\{v_2\} = intersect(e_1, e_2)$. The intersection point $intersect(e_2, e_0)$ is degenerate, thus it is not added to V; in fact the point lies outside the feasible space. Six vertices are created for E'. Rays are created at line 9 and 10 if the intersection point is degenerate. The two rays along the boundaries of e_i and $e_{(i+1) \bmod n}$ are generated in loop iteration i when d_{post} is true and iteration $(i+1) \bmod n$ when d_{pre} is true. In our example d_{post} is true for e_2, generating a ray along the boundary of e_2 which recedes in the direction of the first quadrant, whereas d_{pre} is only true for e_0 yielding a ray along e_0 which recedes towards the second quadrant. No rays are created for the polytope E'.

In general both flags might be true. In this circumstance the current inequality e_i cannot define a vertex. In this case an arbitrary point on the boundary of the halfspace of e_i is created at line 12 to fix its representing rays in space. Another case not encountered in this example arises when the polyhedron consists of a single halfspace ($|E| = 1$). In this case a third ray is created (line 4) to indicate on which side the feasible space lies. Note that the maximum number of elements in R never exceeds four, which occurs when the input defines two facing halfspaces.

The remainder of the *hull* function is dedicated to the reconstruction phase. The point and ray sets, returned by *extreme*, are merged at line 30 and 31. At line 32 the size of a square is calculated which includes all points in P. The square has $\langle s, s\rangle$, $\langle -s, s\rangle$, $\langle s, -s\rangle$, $\langle -s, -s\rangle$ as its corners. The square in the running example is depicted in all three frames of Figure 3 and the origin is marked with a cross. Each point $p \in P$ is then translated by each ray $r \in R$ yielding the point set Q. Translated points appear outside the square since all normalised rays are translated by the length of the diagonal $2\sqrt{2}s$ of the square. The translation process for the worked example is depicted in the second frame. Line 38 is not relevant to this example as it traps the case when the output polyhedron consists of a single point. Line 40 calculates a feasible point q_p of the convex hull of Q which is not a vertex. This point serves as the pivot point in the classic Graham scan. First, the point set Q is sorted counter-clockwise with respect to q_p. Second, interior points are removed, yielding the indices of all vertices, in the case of the example $k_0, \ldots, k_7$. What follows is a round-trip around the hull which translates pairs of adjacent vertices into inequalities by calling *connect* at line 48. Whether this inequality actually appears in the result depends on the state of the *add* flag. In our particular example the *add* flag is only set at line 50. Whenever it is set, it is because one of the two vertices lies within the square. The resulting polyhedron consists of the inequalities $\overline{q_{k_2}, q_{k_3}}$, $\overline{q_{k_3}, q_{k_4}}$ and $\overline{q_{k_4}, q_{k_5}}$ which is a correct solution for this example.

In general, the reconstruction phase has to consider certain anomalies that mainly arise in outputs of lower dimensionality. One subtlety in the two dimensional case is the handling of polyhedra which contain lines. This is illustrated in Figure 4 where the two inequalities e_0, e_1 are equivalent to one equation which

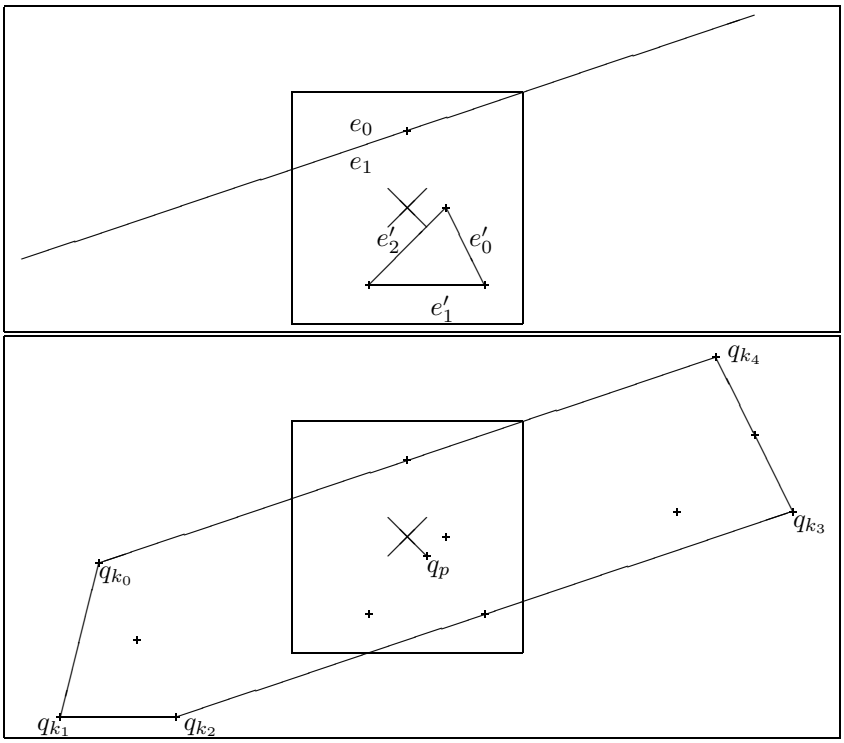

Figure 4: The resulting convex hull contains a line.

defines a space that is a line. Observe from the second frame that no point in the square is a vertex in the hull of Q. Therefore the predicate *inBox* does not hold for the two vertices q_{k_2} and q_{k_3} and the desired inequality $\overline{q_{k_2}, q_{k_3}}$ is not emitted. Similarly for the vertices q_{k_4} and q_{k_0}. However, in such cases there always exists a point in $p \in Q$ with $\overline{q_p, q_{k_i}} \measuredangle \overline{q_p, p} < \overline{q_p, q_{k_i}} \measuredangle \overline{q_p, q_{k_{(i+1) \bmod m}}}$ which lies in the square. Thus it is sufficient to search for an index $j \in [k_i + 1, k_{(i+1) \bmod m} - 1]$ such that q_j is both in the square and on the line connecting the vertices q_{k_i} and $q_{k_{(i+1) \bmod m}}$. The inner loop at lines 52–56 tests if Q contains such a point and sets *add* appropriately.

The one dimensional case is handled by the $m = 2$ tests at line 50 and 57. Figure 5 illustrates why the test in line 50 is necessary. Suppose E_1 and E_2 are given such that $\mathit{extreme}(E_1) = \langle \{q_4, q_5\}, \emptyset \rangle$ and $\mathit{extreme}(E_2) = \langle \{q_3\}, \{r, -r\} \rangle$ where r is any ray parallel to the line. Observe that all points are collinear, thus the pivot point is on the line and a stable sort could return the ordering depicted in the figure. The correct inequalities for this example are $E_{\text{res}} = \{\overline{q_0, q_8}, \overline{q_8, q_0}\}$. The Graham scan will identify $q_{k_0} = q_0$ and $q_{k_1} = q_8$ as vertices. Since there exists $j \in [k_0 + 1, k_1 - 1]$ such that $\mathit{inBox}(s, q_j)$ holds, $\overline{q_0, q_8} \in E_{\text{res}}$. In contrast, although there are boundary points between q_8 and q_0 the loop cannot locate

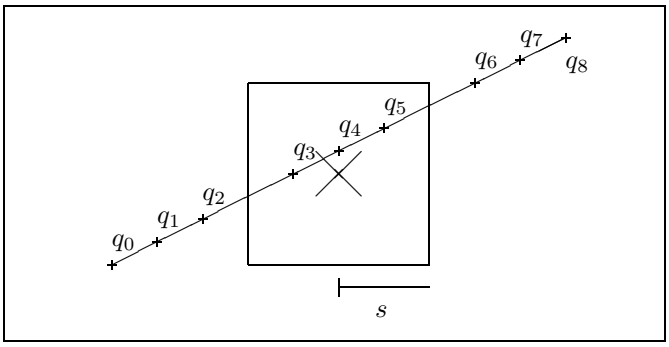

Figure 5: Handling the one-dimensional case.

them due to the ordering. In this case the $m = 2$ test ensures that *add* is set, guaranteeing that $\overline{q_8, q_0} \in E_{\text{res}}$.

The output polyhedron must include q_{k_i} as a vertex whenever $inBox(s, q_{k_i})$ holds. If $inBox(s, q_{k_i})$ holds, the algorithm generates $e_{i-1} = \overline{q_{k_{(i-1) \bmod m}}, q_{k_i}}$ and $e_i = \overline{q_{k_i}, q_{k_{(i+1) \bmod m}}}$. If $e_{i-1} \measuredangle e_i < \pi$, then $\{q_{k_i}\} = intersect(e_{i-1}, e_i)$ and the vertex q_{k_i} is realized. Observe that if $m = 2$, $e_{i-1} \measuredangle e_i = \pi$ which necessitates an additional inequality to define the vertex q_{k_i}. This is the rôle of the inequality generated on lines 58 or 59. Observe that this inequality e guarantees $e_{i-1} \measuredangle e < \pi$ and $e \measuredangle e_i < \pi$ which is sufficient to define q_{k_i}.

The zero dimensional case corresponds to the case of when both input polyhedra consist of the single point v. Line 38 traps this case and returns a set of inequalities describing $\{v\}$. Observe that the zero and one dimensional case only require minute changes to the general two dimensional case.

As a note on implementation, observe that the search for a pivot point at line 40 can be refined. One method for finding a definite vertex is to search for a point with extremal coordinates [1]. However, this process requires all points to be examined. The presented algorithm follows Graham [4] in creating an interior point as the pivot point. This does not necessarily require the whole point set to be examined. By choosing two arbitrary points q_1, q_2, it is sufficient to search the point set for a point q_i which does not saturate the line $\overline{q_1, q_2}$. The centre of the triangle q_1, q_2, q_i is guaranteed to be an interior point of Q.

Note also that the sorting in *extreme* is unnecessary if the input inequalities are consecutive in terms of angle. In particular, the output of one run of the algorithm can serve as an input to another without applying the sort at line 2.

Finally observe that the inner loop at lines 52–56 can often be skipped: if the line between q_{k_i} and $q_{k_{(i+1) \bmod m}}$ does not intersect with the square, $inBox(s, q)$ cannot hold for any $q \in Q$. Hence *add* will not be set at line 54 and the inner loop has no effect.

3 Proof of Correctness

Section 3.1 introduces the mathematical language necessary for expressing the two parts of the proof. The proof itself reflects the structure of the algorithm: Section 3.2 concerns the conversion of planar H-polyhedra into their ray and point representations; Section 3.3 argues that the reconstructed polyhedron encloses the points and rays generated from the two input polyhedra, and yet is also minimal.

3.1 Preliminaries

A polyhedron is a set $P \subseteq \mathbb{R}^d$ such that $P = \{\vec{x} \in \mathbb{R}^d \mid A\vec{x} \leq \vec{c}\}$ for some matrix $A \in \mathbb{R}^{n \times d}$ and vector $\vec{c} \in \mathbb{R}^n$. An H-polyhedron is a pair $\langle A, \vec{c}\rangle$ where $A \in \mathbb{R}^{n \times d}$ and $\vec{c} \in \mathbb{R}^n$ that is interpreted by $[\![.]\!]$ as the polyhedron $[\![\langle A, \vec{c}\rangle]\!] = \{\vec{x} \in \mathbb{R}^d \mid A\vec{x} \leq \vec{c}\}$. For brevity we manipulate H-polyhedra as a finite set of inequalities $E = \{\vec{a}_1 \cdot \vec{x} \leq c_1, \ldots, \vec{a}_n \cdot \vec{x} \leq c_n\}$ which is equivalent to the matrix representation $\langle A, \vec{c}\rangle$ with $A = \langle \vec{a}_1, \ldots, \vec{a}_n\rangle^T$ and $\vec{c} = \langle c_1, \ldots, c_n\rangle^T$. E is said to be satisfiable if $[\![E]\!] \neq \emptyset$ and non-redundant if $\forall e \in E \,.\, [\![E \setminus \{e\}]\!] \neq [\![E]\!]$. Two inequalities e, e' coincide, written $\mathrm{c}(e, e')$, iff there exists $\vec{a} \in \mathbb{R}^d, c \in \mathbb{R}$ with $[\![\{e, e'\}]\!] = \{\vec{x} \in \mathbb{R}^d \mid \vec{a} \cdot \vec{x} = c\}$.

The convex hull of a finite set of points $P = \{\vec{p}_1, \ldots, \vec{p}_n\} \subseteq \mathbb{R}^d$ is defined as $\mathit{conv}(P) = \{\sum_{i=1}^n \lambda_i \vec{p}_i \mid 0 \leq \lambda_i \wedge \sum_{i=1}^n \lambda_i = 1\}$. Moreover, the cone of a finite set of vectors $R = \{\vec{r}_1, \ldots, \vec{r}_m\} \subseteq \mathbb{R}^d$ is defined as $\mathit{cone}(R) = \{\sum_{i=1}^m \lambda_i \vec{r}_i \mid 0 \leq \lambda_i\}$. The Minkowski sum of two sets $X, Y \subseteq \mathbb{R}^d$ is defined as $X + Y = \{\vec{x} + \vec{y} \mid \vec{x} \in X \wedge \vec{y} \in Y\}$. Let $\mathit{vert}(S) = \{p \in S \mid p \notin \mathit{conv}(S \setminus \{p\})\}$, $\mathit{ray}(S) = \{r \in \mathbb{R}^d \setminus \{\vec{0}\} \mid S + \mathit{cone}(\{r\}) \subseteq S\}$ and $\mathit{line}(S) = \{r \in \mathit{ray}(S) \mid -r \in \mathit{ray}(S)\}$ denote the vertices, rays and lines of a convex set S. The following result, accredited to Motzkin [11], relates the two classic representation of polyhedra.

Theorem 3.1 The following statements are equivalent for any $S \subseteq \mathbb{R}^d$:

1. $S = \mathit{conv}(P) + \mathit{cone}(R)$ for some finite $P, R \subseteq \mathbb{R}^d$;
2. $S = \{\vec{x} \in \mathbb{R}^d \mid A\vec{x} \leq \vec{c}\}$ for some matrix $A \in \mathbb{R}^{n \times d}$ and vector $\vec{c} \in \mathbb{R}^n$.

Our algorithm converts the two (planar) input H-polyhedra E_i into their rays R_i and points P_i and calculates an H-polyhedron E with the smallest $[\![E]\!]$ such that $[\![E]\!] \supseteq [\![E_1]\!] \cup [\![E_2]\!]$. In fact $[\![E]\!] = \mathit{conv}(P_1 \cup P_2) + \mathit{cone}(R_1 \cup R_2)$.

3.2 Decomposition

The discussion of the function *extreme* is organised by the dimension of the polyhedron. Reoccurring or self-contained arguments are factored out in the following lemmata. The first lemma states how redundancy can follow from the angular relationship between three inequalities. Since *extreme* requires its input to be non-redundant, useful angular properties of the input inequalities flow from the lemma.

Lemma 3.1 *Suppose* $e_i = a_i x + b_i y \leq c_i$, $i = 1, 2, 3$. *Let* $c_1 = c_3 = 0$, $c_2 \geq 0$. *Then* $[\![\{e_1, e_3\}]\!] \subseteq [\![\{e_2\}]\!]$ *if* $0 = \theta(e_1) < \theta(e_2) < \theta(e_3) < \pi$.

Proof. To show $[\![\{e_1, e_3\}]\!] \subseteq [\![\{e_2\}]\!]$ it is sufficient to show $[\![\{e_1, e_3\}]\!] \subseteq [\![\{a_2 x + b_2 y \leq 0\}]\!]$, thus let $c_2 = 0$. Furthermore, w.l.o.g. let $e_1 \equiv x \leq 0$ (since $\theta(e_1) = 0$), $e_2 \equiv a_2 x + y \leq 0$ (since $\theta(e_2) < \pi$) and $e_3 \equiv a_3 x + y \leq 0$ (since $\theta(e_3) < \pi$). Note that $a_2 = \lambda_1 a_1 + \lambda_3 a_3$ and $b_2 = \lambda_1 b_1 + \lambda_3 b_3$ has the solution $\lambda_1 = a_2 - a_3$ and $\lambda_3 = 1$. Due to $b_2 = 1$, $a_2 = \cot^{-1}(\theta(e_2))$ and similarly $a_3 = \cot^{-1}(\theta(e_3))$. Because $\theta(e_2) < \theta(e_3)$ and cot is an anti-monotone on $(0, \pi)$, it follows that $a_2 > a_3$, hence $\lambda_1 > 0$. Let $\langle x, y \rangle$ satisfy $e_1 \equiv x \leq 0$ and $e_3 \equiv a_3 x + y \leq 0$. Due to $\lambda_1 x \leq 0$, $\lambda_1(x) + \lambda_3(a_3 x + y) = (\lambda_1 + a_3)x + y \leq 0 \equiv e_2$, thus e_2 holds, hence $[\![\{e_1, e_3\}]\!] \subseteq [\![\{e_2\}]\!]$ as required. ■

The following lemma states that there is an injection between vertices and inequalities. This result is used to show that the loop in *extreme* does indeed generate all vertices of the input polyhedron.

Lemma 3.2 *Let* $E = \{e_0, \ldots, e_{n-1}\}$ *be non-redundant and ordered by* θ *and* $v \in vert([\![E]\!])$. *Then there exists* $e_i \in E$ *such that* $\{v\} = intersect(e_i, e_{i'})$ *and* $e_i \measuredangle e_{i'} < \pi$ *where* $i' = (i + 1)$ *mod* n.

Proof. Let $E' \subseteq E$ contain those inequalities that v saturates. Note that $|E'| \geq 2$, in particular there exist $e, e' \in E'$ with $e \measuredangle e' \notin \{0, \pi\}$, otherwise $[\![E']\!] \setminus \{v\}$ is not convex. Choose $e_i, e_k \in E'$ such that $e_i \measuredangle e_k < \pi$. Suppose for the sake of a contradiction that $k \neq (i + 1)$ *mod* n. Then $e_j \in E \setminus E'$ exists with $e_i \measuredangle e_j < e_i \measuredangle e_k$. W.l.o.g. assume that $\theta(e_i) = 0$ and $v = \langle 0, 0 \rangle$. Then $e_i \measuredangle e_j < e_i \measuredangle e_k$ reduces to $0 = \theta(e_i) < \theta(e_j) < \theta(e_k) < \pi$. Lemma 3.1 implies $[\![E \setminus \{e_j\}]\!] \subseteq [\![E]\!]$ which contradicts the assumption that E has no redundant inequalities. Thus $k = (i + 1)$ *mod* $n = i'$ ■

The following result also builds on Lemma 3.1 and complements the previous lemma in that it states when adjacent inequalities have feasible intersections points. The lemma is used to show that *extreme* only generates points in $[\![E]\!]$.

Lemma 3.3 *Let* $E = \{e_0, \ldots, e_{n-1}\}$ *be satisfiable, non-redundant and ordered by* θ. *For any* $e_i \in E$ *if* $e_i \measuredangle e_{(i+1) \bmod n} < \pi$ *then* $intersect(e_i, e_{(i+1) \bmod n}) \subseteq [\![E]\!]$.

Proof. Let $e_i \in E$. Since e_i is not redundant in E the boundary of e_i intersects with the non-empty convex body $[\![E \setminus \{e_i\}]\!]$. Let $e_m \in E \setminus \{e_i\}$ such that $\emptyset \neq S = intersect(e_i, e_m) \subseteq [\![E \setminus \{e_i\}]\!]$. Note that if $|S| > 1$ then $E = \{e_i, e_m\}$ with $e_i \measuredangle e_m = \pi$, hence $e_i \measuredangle e_{(i+1) \bmod n} < \pi$ never holds. Let $\{v\} = S$. It remains to show that e_i and e_m are adjacent. Suppose that $e_i \measuredangle e_m < \pi$ ($e_m \measuredangle e_i < \pi$ is analogous). Assume for the sake of a contradiction there exists $e_l \in E \setminus \{e_i\}$ such that $e_i \measuredangle e_l < e_i \measuredangle e_m$. W.l.o.g. $v = \langle 0, 0 \rangle$ and $\theta(e_i) = 0$, hence $0 = \theta(e_i) < \theta(e_l) < \theta(e_m) < \pi$. Since $v \in [\![E \setminus \{e_i\}]\!]$, $v \in [\![e_l]\!]$ and thus $c_l \geq 0$ where

$e_l \equiv a_l x + b_l y \leq c_l$. By Lemma 3.1 e_l is redundant in E which is a contradiction. It follows that $m = (i+1)\ mod\ n$. ■

While the previous lemmata concern points, the following lemma is a statement about rays. In particular, it states when inequalities give rise to rays.

Lemma 3.4 *Let* $E = \{e_0 \equiv a_0 x + b_0 y \leq c_0, \ldots, e_{n-1}\}$ *be satisfiable, non-redundant and ordered by* θ. *Let* $i' = (i+1)\ mod\ n$, $S_1 = cone(\{\langle -b_i, a_i\rangle, \langle b_{i'}, -a_{i'}\rangle\})$ *and* $S_2 = ray([\![E]\!])$. *If* $e_i \measuredangle e_{i'} > \pi$ *or* $|E| \geq 3 \wedge e_i \measuredangle e_{i'} = \pi$ *then* $S_1 = S_2$.

Proof. To show $S_1 \subseteq S_2$. Choose $p \in [\![E]\!]$ such that p saturates e_i. For the sake of a contradiction assume $r_i = \langle -b_i, a_i\rangle \notin ray([\![E]\!])$. Thus there exists $\lambda > 0$ with $p + \lambda r_i \notin [\![E]\!]$. Set $\lambda_{\max} = \max\{\lambda > 0 \mid p + \lambda r_i \in [\![E]\!]\}$. Note that $v = p + \lambda_{\max} r_i$ is a vertex and by Lemma 3.2 there exists $e_k \in E$ with $\{v\} = intersect(e_k, e_{(k+1)\ mod\ n})$ and $e_k \measuredangle e_{(k+1)\ mod\ n} < \pi$. The latter implies that $k \neq i$, therefore $i = (k+1)\ mod\ n$. Hence $e_{(i-1)\ mod\ n} = e_k$ does not contain the ray. W.l.o.g. let $\theta(e_i) = 0$ and $v = \langle 0, 0\rangle$. Then $r_i = \langle 0, 1\rangle$ and $\pi < \theta(e_{(i-1)\ mod\ n}) < 2\pi$, hence $e_{(i-1)\ mod\ n} \equiv a'x + b'y \leq 0$ with $b' < 0$. Thus $a'x + b'(\lambda + y) \leq 0$, hence $p + \lambda r_i \in [\![\{e_{(i-1)\ mod\ n}\}]\!]$ which is a contradiction. Analogously for $r_{i'} = \langle b_{i'}, -a_{i'}\rangle$. Now to show $S_2 \subseteq S_1$. Assume there exists $r \in ray([\![E]\!]) \setminus cone(\{\langle -b_i, a_i\rangle, \langle b_{i'}, -a_{i'}\rangle\})$. Consider $e_i \measuredangle e_{i'} > \pi$. Since $\theta(e_i) \neq \theta(e_{i'})$ there exists $\lambda_1, \lambda_2 \in \mathbb{R}$ with $r = \lambda_1\langle -b_i, a_i\rangle + \lambda_2\langle b_{i'}, -a_{i'}\rangle$. Assume $\lambda_2 < 0$. W.l.o.g. let $\{p\} = intersect(e_i, e_{i'}) = \{\langle 0, 0\rangle\}$ and $\theta(e_i) = 0$ thus let $e_i \equiv x \leq 0$. It follows that $\pi < \theta(e_{i'}) < 2\pi$, $b_{i'} \leq 0$ and we set $e_{i'} \equiv a'_{i'}x - y \leq 0$. Thus $r = \lambda_1\langle 0, 1\rangle + \lambda_2\langle -1, -a'_{i'}\rangle = \langle -\lambda_2, \lambda_1 - \lambda_2 a'_{i'}\rangle$. Since $\lambda_2 < 0$, $p + r \notin [\![\{e_i\}]\!]$ which contradicts $r \in ray([\![E]\!])$. Analogously for $\lambda_1 < 0$ with $\theta(e_{i'}) = 0$. Since $\lambda_1 \geq 0$ and $\lambda_2 \geq 0$, $r \in cone(\{\langle -b_i, a_i\rangle, \langle b_{i'}, -a_{i'}\rangle\})$ which is a contradiction. Now suppose $e_i \measuredangle e_{i'} = \pi$ and $|E| \geq 3$. W.l.o.g. let $\theta(e_i) = 0$, $e_i \equiv x \leq |u|$ and $e_{i'} \equiv -x \leq 0$. Thus $S_1 = cone(\{\langle 0, 1\rangle\})$. Let $\langle x_r, y_r\rangle = r$ and $p \in [\![\{e_i, e_{i'}\}]\!]$. Observe that $x_r = 0$ otherwise there exists $\lambda > 0$ with $p + \lambda r \notin [\![\{e_i, e_{i'}\}]\!]$. Now assume $y_r < 0$. Since $|E| \geq 3$, there exists $e_j \in E$ such that $\pi < \theta(e_j) < 2\pi$ and thus $e_j \equiv a_j x + b_j y \leq c_j$ with $b_j < 0$. Let $p \in [\![\{e_j\}]\!]$, then there exists $\lambda > 0$ such that $p + \lambda r \notin [\![\{e_j\}]\!]$ which is a contradiction. ■

The correctness of the first stage of our algorithm is summarised by the following proposition. Note that the Minkowski sum $conv(V) + \emptyset$ always defines the empty space rather than the polyhedron $conv(V)$. Thus a null ray $\langle 0, 0\rangle$ is required to represent a bounded polyhedron. The function *extreme* avoids adding this null ray for the sake of improved efficiency. In *hull* the translation of points at line 36 by the null ray is replaced by a simple copying step at line 35. However, the null ray still manifests itself in the correctness results.

Proposition 3.1 *Let* $E \subset Lin$ *be non-empty, finite, satisfiable and non-redundant. Then* $extreme(E) = \langle P, R\rangle$ *and* $[\![E]\!] = conv(P) + cone(R \cup \{\langle 0, 0\rangle\})$.

Note that the following proof handles polyhedra that contain lines as a special case. This distinction is not artificial, in fact Klee [5] observed that a closed convex set that does not contain lines is generated by its vertices and its extreme rays. (Extreme rays are rays that cannot be expressed by a linear combination of others.) In order to describe polyhedra which contain lines it is necessary to create points that are not vertices and rays which are not extreme.

Proof. Let $\{e_0, \ldots, e_{n-1}\} = E$ such that $\theta(e_0) < \theta(e_1) < \ldots < \theta(e_{n-1})$. Such an ordering exists since if $\theta(e_i) = \theta(e_j)$ for some $i \neq j$ then either $[\![e_i]\!] \subseteq [\![e_j]\!]$, thus $[\![E \setminus \{e_j\}]\!] = [\![E]\!]$ or vice-versa which contradicts that E is non-redundant. Let $\langle P, R\rangle = \mathit{extreme}(E)$.

- Suppose $\mathit{line}([\![E]\!]) \neq \emptyset$. Note that if there exist $e_i, e_j \in E$ with $e_i \measuredangle e_j < \pi$ then $\mathit{line}([\![E]\!]) = \emptyset$, hence $E = \{e\}$ or $E = \{e_0, e_1\} \wedge e_0 \measuredangle e_1 = \pi$.
  - Consider $E = \{e\}$. The loop is executed only once with $d_{pre} = d_{post} = \mathit{true}$, thus the boundary point p and two rays r_1, r_2 are added in line 12, 9 and 10, respectively. The boundary of $[\![E]\!]$ is $\{p + \lambda_1 r_1 + \lambda_2 r_2 \mid \lambda_i \geq 0\}$. The ray r_3 added in line 4 points into the halfspace, thus $[\![E]\!] = \{p\} + \mathit{cone}(\{r_1, r_2, r_3\})$.
  - Consider $E = \{e_0, e_1\} \wedge e_0 \measuredangle e_1 = \pi$. For each e_i, $d_{pre} = d_{post} = \mathit{true}$, hence for each e_i the loop generates two rays r_i, r_i' along the boundary of e_i (lines 9, 10) and one boundary point p_i (line 11). The set $\mathit{conv}(\{p_1, p_2\})$ is included in $[\![E]\!]$. Note that the rays generated in the second iteration are collinear to those in the first. Thus $[\![E]\!] = \mathit{conv}(\{p_1, p_2\}) + \mathit{cone}(\{r_1, r_1'\})$.
- Suppose E is zero dimensional, hence $[\![E]\!] = \{v\}$. Let $\alpha_i = e_i \measuredangle e_{(i+1) \bmod n}$. Since $\sum_{i=0}^{n-1} \alpha_i = 2\pi$, it follows that $n \geq 3$ since otherwise there exists i such that $\alpha_i \geq \pi$ and by Lemma 3.4 it follows that $\mathit{ray}([\![E]\!]) \neq \emptyset$. By Lemma 3.1, it follows that $\alpha_i + \alpha_{(i+1) \bmod n} \geq \pi$, which for all e_i is $\sum_{i=0}^{n-1}(\alpha_i + \alpha_{(i+1) \bmod n}) \geq n\pi$. However, $n\pi \leq \sum_{i=0}^{n-1}(\alpha_i + \alpha_{(i+1) \bmod n}) = 2\sum_{i=0}^{n-1} \alpha_i = 4\pi$, hence $n \leq 4$.
  - Consider $E = \{e_0, e_1, e_2, e_3\}$. Observe that for all $0 \leq i \leq 3$, $\alpha_i + \alpha_{(i+1) \bmod n} = \pi$ and $\mathit{intersect}(e_i, e_{(i+1) \bmod n}) = \{v\}$, hence $\mathrm{c}(e_0, e_2) \wedge \mathrm{c}(e_1, e_3)$. In all loop iterations $d_{pre} = d_{post} = \mathit{false}$ thus only vertex $\{v\} = P$ is generated (line 11).
  - Consider $E = \{e_0, e_1, e_2\}$. Observe that for all i, $e_i \measuredangle e_{(i+1) \bmod n} < \pi$, otherwise by Lemma 3.4, $\mathit{ray}([\![E]\!]) \neq \emptyset$. Thus in each iteration i, $d_{post} = d_{pre} = \mathit{false}$ and $P = \{v\} = \mathit{intersect}(e_i, e_{(i+1) \bmod n})$ is generated. Hence $[\![E]\!] = \{v\} + \mathit{cone}(\{\langle 0, 0\rangle\})$.
- Suppose E is one dimensional and $\mathit{line}([\![E]\!]) = \emptyset$. Since the boundary of $[\![E]\!]$ contains infinitely many points and $|E| = n$ (which is finite), there exists $e \in E$ such that $p_1, p_2 \in [\![E]\!]$ where $p_1 \neq p_2$ and p_1 and p_2 saturate e. In fact there are infinitely many boundary points on the line between

p_1 and p_2 which saturate e. Observe $[\![E]\!]$ contains no interior points, hence $[\![E]\!]$ consists only of boundary points. Therefore there exists $e' \in E$ that saturates infinitely many of these points and for which $\mathrm{c}(e, e')$ holds. As we assume that $\mathit{line}([\![E]\!]) = \emptyset$, $|E| < 2$. Due to non-redundancy, there exists at most one $e_i \in E$ with $0 < e \measuredangle e_i < \pi$. Similarly for e'. Hence $3 \leq |E| \leq 4$ follows.

- Consider $E = \{e_0, e_1, e_2\}$. Let $i \in [0, 2]$ such that $\mathrm{c}(e_i, e_{(i+1) \bmod n})$ holds. On iteration i the loop generates a ray r along the boundary of $[\![e_i]\!]$ in line 10. A collinear ray is generated in iteration $(i+1) \bmod n$ for $e_{(i+1) \bmod n}$ on line 9. It is in this and iteration $(i+2) \bmod n$ where $\{v\} = \mathit{intersect}(e_{(i+1) \bmod n}, e_{(i+2) \bmod n}) = \mathit{intersect}(e_{(i+2) \bmod n}, e_i)$ is added to P. Thus $[\![E]\!] = \{v\} + \mathit{cone}(\{r\})$.
- Consider $E = \{e_0, e_1, e_2, e_3\}$. Let $i \in [0, 3]$ such that $\mathrm{c}(e_i, e_{(i+2) \bmod n})$ holds. In all four iterations $d_{pre} = d_{post} = \mathit{false}$ holds, resulting in two vertices resulting from the intersection of adjacent inequalities: $\{v_1\} = \mathit{intersect}(e_i, e_{(i+1) \bmod n}) = \mathit{intersect}(e_{(i+1) \bmod n}, e_{(i+2) \bmod n})$, $\{v_2\} = \mathit{intersect}(e_{(i+2) \bmod n}, e_{(i+3) \bmod n}) = \mathit{intersect}(e_{(i+3) \bmod n}, e_i)$. Hence $[\![E]\!] = \mathit{conv}(\{v_1, v_2\}) + \mathit{cone}(\{\langle 0, 0\rangle\})$.

- Suppose that E is two dimensional and that none of the preceeding cases apply.
  - Since c never holds for $|E| > 4$ the previous cases deal with all $e, e' \in E$ where $\mathrm{c}(e, e')$ holds. Because $[\![E]\!]$ does not consist of a single point described by three inequalities, it must be two dimensional.
  - Let $v \in V = \mathit{vert}([\![E]\!])$. By Lemma 3.2 there exists $e_i \in E$ such that $e_i \measuredangle e_{(i+1) \bmod n} < \pi$ and $\{v\} = \mathit{intersect}(e_i, e_{(i+1) \bmod n})$. Then d_{post} is false in iteration i, hence $v \in P$, thus $V \subseteq P$. By Lemma 3.3 $P \subseteq [\![E]\!]$. If for all i, $e_i \measuredangle e_{(i+1) \bmod n} < \pi$ holds then $[\![E]\!] = \mathit{conv}(P) + \mathit{cone}(\langle 0, 0\rangle)$. Otherwise let $i \in [0, m-1]$ such that $e_i \measuredangle e_{(i+1) \bmod n} \geq \pi$. Then d_{post} will be true in iteration i and d_{pre} will hold in iteration $(i+1) \bmod n$ yielding rays that are collinear to $r_1 = \langle -b_i, a_i\rangle$ and $r_2 = \langle b_{(i+1) \bmod n}, -a_{(i+1) \bmod n}\rangle$. Since the previous cases do not apply, whenever $e_i \measuredangle e_{(i+1) \bmod n} = \pi$, $|E| \geq 3$ hence Lemma 3.4 can be applied. It follows that $\mathit{ray}([\![E]\!]) = \mathit{cone}(\{r_1, r_2\})$, hence $[\![E]\!] = \mathit{conv}(P) + \mathit{cone}(\{r_1, r_2\})$.

■

3.3 Reconstruction

One advantage of the point and ray representation (over one that makes lines explicit) is that *extreme* naturally generates lines as two opposing rays in independent iterations, thus an explicit line case is not required. The remainder of

the *hull* function combines the points and rays of the two input polyhedra to construct a corresponding set of inequalities. The advantage of the simplified representation carries over to the reconstruction phase in that opposing rays from different polyhedra need not be recognised and reconstituted as a line.

Theorem 3.2 *Given $E_1, E_2 \subset Lin$ be non-empty, finite, satisfiable and non-redundant, $E_{\text{res}} = hull(E_1, E_2)$ is non-redundant and the smallest $E_{\text{res}} \subset Lin$ with $[\![E_{\text{res}}]\!] \supseteq [\![E_1]\!] \cup [\![E_2]\!]$ such that for all $E \in Lin$, $[\![E]\!] \supseteq [\![E_1]\!] \cup [\![E_2]\!] \Rightarrow [\![E_{\text{res}}]\!] \subseteq [\![E]\!]$.*

Proof. The case for $E_1 = \emptyset$ or $E_2 = \emptyset$ on line 27 is trivial, thus assume $E_1 \cup E_2 \neq \emptyset$ and that lines 28–35 are executed. Note that for all $r \in R$, $|r| = 1$ due to the normalisation in *extreme*. Thus $(P + \{2\sqrt{2}sr \mid r \in R\}) \cap (-s, s)^2 = \emptyset$. Hence after line 12 the predicate $inBox(s, p)$ holds whenever $p \in P = Q \setminus (P + \{2\sqrt{2}sr \mid r \in R\})$. Line 38 handles the zero dimensional case, hence from line 39 onwards $|Q| > 1$. The arithmetic mean q_p of all points is calculated in line 40. This point serves as reference when comparing two points for counter-clockwise ordering. Observe that $q_p \in conv(Q)$, in particular q_p is not on its boundary if $conv(Q)$ is two dimensional. The latter ensures for all boundary points $q_1, q_2 \in conv(Q)$, $\theta(\overline{q_p, q_1}) \neq \theta(\overline{q_p, q_2})$, thus line 18 yields a total ordering on the boundary points in the two dimensional case. Line 44 performs the classic Graham scan which identifies the vertices of $conv(Q)$.

- To show $[\![E_{\text{res}}]\!] \supseteq [\![E_1]\!] \cup [\![E_2]\!]$. In particular, show $[\![E_{\text{res}}]\!] \supseteq [\![E_1]\!]$, i.e. for all $e \in E_{\text{res}}$, $[\![\{e\}]\!] \supseteq [\![E_1]\!]$.
  - To show $conv(Q) \subseteq [\![\{e\}]\!]$. Suppose e is added in line 60. Then the flag *add* was true and $e = connect(q_{k_i}, q_{k_{(i+1) \bmod m}})$. For the sake of a contradiction, suppose there exists $q_{k_j} \notin [\![\{e\}]\!]$ such that $j \notin \{i, (i+1) \bmod m\}$. W.l.o.g. let $\theta(\overline{q_p, q_{k_j}}) < \theta(\overline{q_p, q_{k_i}})$. There exists a line through q_p and q_{k_j} which intersects the boundary of e at a point z. Then $q_{k_j} \in conv(\{z, q_{k_{(i+1) \bmod m}}\})$ which contradicts that q_{k_i} is a vertex. Since all $q_{k_0}, \ldots, q_{k_{m-1}} \in [\![\{e\}]\!]$ it follows that $conv(Q) \subseteq [\![\{e\}]\!]$. Furthermore it is easy to verify that the inequality e added in line 58 or 59 holds for both $\{q_{k_0}, q_{k_1}\} = Q$, hence $conv(Q) \subseteq [\![\{e\}]\!]$.
  - To show $cone(R) \subseteq ray([\![\{e\}]\!])$.
    - Suppose $m > 2$. If either add was set at line 50 or line 54, there exists $q_j \in Q$ which saturates e such that $inBox(s, q_j)$ holds, hence $q_j \in P$. Let $r \in R$. Then $q_j + 2\sqrt{2}sr \in conv(Q) \subseteq [\![\{e\}]\!]$. Since q_j saturates e, it follows that $r \in ray([\![\{e\}]\!])$.
    - Suppose $m = 2$. Assume that $inBox(s, q_{k_0})$ and $inBox(s, q_{k_1})$ both do not hold, then there exist $p \in P$ and $r \in R$ with $q_{k_0} = p + 2\sqrt{2}sr$. Note that since $m = 2$, p saturates e. Continue as above. Assume e was added in lines 58–59. Observe that $q_j = q_{k_i}$ saturates e. Again, continue as in the first case.

Thus $[\![E_1]\!] = conv(P_1)+cone(R_1) \subseteq conv(P)+cone(R) \subseteq [\![\{e\}]\!]$. Similarly for E_2. Thus $[\![E_{\text{res}}]\!] \supseteq [\![E_1]\!] \cup [\![E_2]\!]$.

- To show for all $E \subset Lin$, $[\![E]\!] \supseteq [\![E_1]\!] \cup [\![E_2]\!] \Rightarrow [\![E_{\text{res}}]\!] \subseteq [\![E]\!]$. For the sake of a contradiction suppose there exists $p \in [\![E_{\text{res}}]\!]$ such that $p \notin conv(P) + cone(R)$. Hence for all $e \in E_{\text{res}}$, $p \in [\![\{e\}]\!]$. Let $p' \in conv(P) + cone(R)$ such that $|p - p'|$ is minimal. Observe that p' is unique due to convexity.

  - Suppose $p' \in vert(conv(P)+cone(R))$. Hence there exists $i \in [0, m-1]$ such that $p' = q_{k_i}$. Assume that $inBox(s, q_{k_i})$ does not hold. Then there exists $p'' \in P$, $r \in R$ and $\lambda > 0$ such that $q_{k_i} = p'' + \lambda r$. Since $[\![E_{\text{res}}]\!] \supseteq conv(P) + cone(R)$, $p'' + 2\lambda r \in [\![E_{\text{res}}]\!]$, thus $q_{k_i} \in conv(\{p'', p'' + 2\lambda r\})$ which is a contradiction, hence $inBox(s, q_{k_i})$ holds. Thus the flag *add* is set on line 54 in loop iteration $(i - 1)$ *mod* m and i, hence the inequalities $e_{(i-1) \bmod n} = \overline{q_{k_{(i-1) \bmod m}}, q_{k_i}}$ and $e_i = \overline{q_{k_i}, q_{k_{(i+1) \bmod m}}}$ are added to E_{res}.
    - Suppose $e_{i-1} \measuredangle e_i < \pi$. Then $\{q_{k_i}\} = intersect(e_{i-1}, e_i)$. Due to convexity $|q_{k_{(i-1) \bmod m}} - p| > |q_{k_i} - p|$ and $|q_{k_{(i+1) \bmod m}} - p| > |q_{k_i} - p|$. Hence $p' = q_{k_i}$ is the closest point to p in the space $[\![\{e_{(i-1) \bmod n}, e_i\}]\!]$. But this implies that $p \notin [\![E_{\text{res}}]\!]$ which is a contradiction.
    - Suppose $e_{i-1} \measuredangle e_i = \pi$. Then $q_{k_{(i-1) \bmod m}} = q_{k_{(i+1) \bmod m}}$, thus $m = 2$. Note that $p \in [\![\{e_{i-1}, e_i\}]\!]$ otherwise $p \notin E_{\text{res}}$. Since $q_{k_i} \in P$, $inBox(s, q_{k_i})$ holds and line 58 or 59 adds an inequality $e \in E_{\text{res}}$ with $\{q_{k_i}\} = intersect(e_{i-1}, e) = intersect(e, e_i)$. Observe that $p \notin [\![\{e\}]\!]$ otherwise $p \in conv(\{q_{k_i}, q_{k_{(i+1) \bmod m}}\})$, thus $p \notin [\![E_{\text{res}}]\!]$.
  - Suppose $p' \notin vert(conv(P)+cone(R))$, thus p' is a boundary point of $conv(P) + cone(R)$. There exists a line $L \subset \mathbb{R}^2$ such that $p' \in L$ and $conv(P)+cone(R) \setminus L$ is convex (but not closed). There exists a loop iteration i such that the boundary of $e = connect(q_{k_i}, q_{k_{(i+1) \bmod m}})$ is exactly L. Since $conv(P) + cone(R) \subseteq [\![\{e\}]\!]$, $p \notin [\![\{e\}]\!]$.
    - Suppose $inBox(s, q_{k_i})$ or $inBox(s, q_{k_{(i+1) \bmod m}})$ holds. The flag add is true, thus $e \in E_{\text{res}}$ and $p \notin [\![E_{\text{res}}]\!]$.
    - Suppose neither $inBox(s, q_{k_i})$ nor $inBox(s, q_{k_{(i+1) \bmod m}})$ holds. There exists $q_j \in P$ such that $q_{k_i} = q_j + \lambda r$ for some $\lambda > 0$ and $r \in R$. Due to the ordering of the points, $k_i < j < k_{(i+1) \bmod m}$ whenever $conv(P)+cone(R)$ is two dimensional. Thus *add* is set in line 54. If $conv(P) + cone(R)$ is one dimensional, $m = 2$ and *add* is set in line 49. In both cases $e \in E_{\text{res}}$ and thus $p \notin [\![\{e\}]\!]$.

It remains to show that E_{res} is non-redundant. Note that the loop at lines 46–62 iterates once for each vertex q_{k_i}, creating inequalities $e_i = \overline{q_{k_i}, q_{k_{(i+1) \bmod m}}}$. For all $i \neq j$, $\theta(e_i) \neq \theta(e_j)$ due to the fact that $q_{k_0}, \ldots, q_{k_{m-1}}$ are consecutive vertices of the hull of $conv(Q)$, thus $\{e_0, \ldots, e_{m-1}\}$ has no redundancies. Now consider the inequality e'_i added at line 58 or 59. Since $m = 2$, $e_i \measuredangle e_{(i+1) \bmod m} =$

π and since $\theta(e_i) < \theta(e'_i) < \theta(e_{(i+1) \bmod m})$, the inequality e'_i is not redundant. Hence $E_{\text{res}} \subseteq \{e_0, e'_0, \ldots, e_{m-1}, e'_{m-1}\}$ is non-redundant. ■

The running time of the algorithm is $O(n \log n)$ where $n = |E_1| + |E_2|$. After the sorting step at line 2 in *extreme*, each inequality generates at most two rays and one point. Thus *extreme* is in $O(n \log n)$. The flag d_{pre} is true if the angle between two consecutive inequalities is at least π. Thus d_{pre} can only be true in at most two loop iterations. Similarly for d_{post}. Hence *extreme* returns at most four rays for each polyhedron, thus $|R| \leq 8$ at line 31 and $O(|Q|) = O(n)$. The dominating cost in the *hull* function is the sorting step at line 42 which we assume is in $O(n \log n)$. The scan is linear [4] and partitions the point set Q into vertices and non-vertices. The loop at lines 46–62 runs once for each vertex, whereas the inner loop at lines 52–56 runs at most once for each non-vertex. It follows that the overall running time is in $O(n \log n)$.

4 Conclusion

An $O(n \log n)$ algorithm for calculating the convex hull of planar H-polyhedra has been presented, thereby improving on existing approaches. The algorithm applies a novel box construction which reduces the problem to calculating the convex hull of a set of points. Implementing the algorithm exposed a number of subtleties which motivated a complete proof of the algorithm.

References

[1] Kenneth R. Anderson. A Reevaluation of an Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set. *Information Processing Letters*, 7(1):53–55, January 1978.

[2] D. Avis and K. Fukuda. A Pivoting Algorithm for Convex Hulls and Vertex Enumeration of Arrangements and Polyhedra. *Discrete Computational Geometry*, 8:295–313, 1992.

[3] N. V. Chernikova. Algorithm for Discovering the Set of All Solutions of a Linear Programming Problem. *USSR Computational Mathematics and Mathematical Physics*, 8(6):282–293, 1968.

[4] R. L. Graham. An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set. *Information Processing Letters*, 1(4):132–133, 1972.

[5] V. L. Klee. Some characterizations of convex polyhedra. *Acta Mathematica*, 102:79–107, 1959.

[6] H. Le Verge. A Note on Chernikova's algorithm. Technical Report 1662, Institut de Recherche en Informatique, Campus Universitaire de Beaulieu, France, 1992.

[7] T. S. Motzkin, H. Raiffa, G. L. Thompson, and R. M. Thrall. The Double Description Method. In *Contributions to the Theory of Games*, number 28 in Annals of Mathematics Study. Princeton University Press, 1953.

[8] F. P. Preparata and M. I. Shamos. *Computational Geometry.* Texts and Monographs in Computer Science. Springer Verlag, 1985.

[9] R. Sedgewick. *Algorithms.* Addison-Wesley, 1988.

[10] R. Seidel. Convex Hull Computations. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 361–376. CRC Press, 1997.

[11] G. Ziegler. *Lectures on Polytopes*, volume 152 of *Graduate Texts in Mathematics.* Springer-Verlag, 1994.