Technical Report

Department of Computer Science
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 97-002

On Some Geometric Optimization
Problems in Layered
Manufacturing

by: J. Majhi, R. Janardan, M. Smid,
and P. Gupta

On Some Geometric Optimization Problems in Layered Manufacturing

Jayanth Majhi* Ravi Janardan* Michiel Smid† Prosenjit Gupta‡

January 23, 1997

Abstract

Efficient geometric algorithms are given for optimization problems arising in layered manufacturing, where a 3D object is built by slicing its CAD model into layers and manufacturing the layers successively. The problems considered include minimizing the degree of stair-stepping on the surfaces of the manufactured object, minimizing the volume of the so-called support structures used, and minimizing the contact area between the supports and the manufactured object—all of which are factors that affect the speed and accuracy of the process. The stair-step minimization algorithm is valid for any polyhedron, while the support minimization algorithms are applicable to convex polyhedra only. Algorithms are also given for optimizing supports for non-convex, simple polygons. The techniques used to obtain these results include construction and searching of certain arrangements on the sphere, 3D convex hulls, halfplane range searching, ray-shooting, visibility, and constrained optimization.

Keywords: Layered manufacturing, computational geometry, optimization.

AMS (MOS) subject classifications: 68Q20, 68Q25, 68U05, 68U07.

Abbreviated title: Geometric optimization in layered manufacturing.

1 Introduction

This paper describes efficient algorithms for certain geometric optimization problems arising in layered manufacturing. In *layered manufacturing*, a physical prototype of a 3D object is built from a (virtual) CAD model by orienting and slicing the model with parallel planes and then manufacturing the slices one by one, each on top of the previous one. Layered manufacturing is the basis of an emerging technology called *Rapid Prototyping and Manufacturing* (RP&M). This technology, which is used extensively in the automotive, aerospace, and medical industries, accelerates dramatically the time it takes to bring a product to the market because it allows the designer to create rapidly

*Department of Computer Science, University of Minnesota, Minneapolis, MN 55455, U.S.A. E-mail: {majhi,janardan}@cs.umn.edu . Research supported in part by a Grant-in-Aid of Research from the Graduate School of the University of Minnesota.

†Fakultät für Informatik, Otto-von-Guericke-Universität Magdeburg, D-39106 Magdeburg, Germany. E-mail: michiel@isg.cs.uni-magdeburg.de . Part of this work was done while the author was at the Department of Computer Science, King's College London, UK.

‡Bell Laboratories, 600 Mountain Avenue, Murray Hill NJ 07974, U.S.A. Work done while at MPI-Informatik, Saarbrücken, Germany and at the University of Minnesota, USA. E-mail: prosenjit@lucent.com .

Figure 1: The stereolithography apparatus.

a physical version of the CAD model (literally on the desktop) and to “feel and touch” it, thereby detecting and correcting flaws in the model early on in the design cycle [18].

Although there are many types of layered manufacturing processes, the basic principle underlying them all is as outlined above. Therefore, for concreteness, we will focus here on just one such method, called *StereoLithography*, which dominates the RP&M market [18].¹

1.1 StereoLithography

The input to the StereoLithography process is a surface triangulation of the CAD model in a format called STL. The triangulated model is oriented suitably, sliced by xy -parallel planes, and then built slice by slice in the positive z direction, as follows:

In essence, the StereoLithography Apparatus (SLA) consists of a vat of photocurable liquid resin, a platform, and a laser. Initially, the platform is below the surface of the resin at a depth equal to the slice thickness. The laser traces out the contour of the first slice on the surface and then hatches the interior, which hardens to a depth equal to the slice thickness. In this way, the first slice is created and it rests on the platform. (See Figure 1.)

Next, the platform is lowered by the slice thickness and the just-vacated region is re-coated with resin. The second slice is then built in the same way. Ideally, each slice after the first one should rest in its entirety on the previous one. In general, however, portions of a slice can overhang the previous slice and so additional structures, called *supports*, are needed to hold up the overhangs. Supports are generated automatically during the process itself. For this the CAD model is analyzed beforehand and a description of the supports is generated and merged into the STL file. Supports come in shapes such as wedges, cylinders, and rectangular blocks.

Once the solid has been made, it is postprocessed to remove the supports. Additional postprocessing is often necessary to improve the finish, which has a stair-stepped appearance on certain surfaces due to the non-zero slice thickness used. (See Figure 2, which shows the cross-section of a polyhedral object; the object is normal to the paper and has a uniform cross-section.)

¹In fact, the recent report of the Computational Geometry Task Force explicitly identifies this process as one where geometric techniques could play a significant role [8, page 31].

Figure 2: *Illustrating stair-stepping.*

1.2 Issues

A key step in layered manufacturing is choosing an orientation for the model, i.e., the *build direction* [18]. Among other things, the build direction affects the quantity of supports used and the surface finish—factors which impact the speed and accuracy of the process. As a simple example, consider building the object in Figure 2: If it is built in the direction $\mathbf{d}$ indicated in the figure, then the manufactured solid will have a stair-stepped finish and will require supports. However, if the build direction is normal to the paper, no supports are needed and there is no stair-stepping. In current systems, the build direction is often chosen by the human operator, based on experience, so that the amount of supports used is “small” and the surface finish is “good”. We seek to design computer algorithms that optimize these criteria automatically and lessen the need for human intervention.²

Let us define more formally the parameters of our problem. Throughout the paper, we denote by $\mathcal{P}$ the polyhedral object that we wish to build and by n the number of vertices in $\mathcal{P}$. (Note that there is no loss of generality in assuming a polyhedral model since the input—the STL representation—is polyhedral, even if the original part is not.) We let $\mathbf{d}$ denote the build direction and, for convenience, imagine it to be vertical so that notions such as “above” and “below” have their usual meaning. *Our problem is to find a $\mathbf{d}$ which minimizes the following three parameters, considered independently (i.e., in isolation from one another):*

Degree of stair-stepping: Due to the non-zero slice thickness, the manufactured part will have a stair-stepped finish on any facet f that is not parallel to $\mathbf{d}$. (See Figure 2. Notice that there is no stair-stepping on the facet corresponding to edge $\overline{34}$ since it is parallel to $\mathbf{d}$.) The degree of stair-stepping on a facet f depends on the angle, $\theta_f(\mathbf{d})$, between the facet normal and $\mathbf{d}$, and it can be mitigated by a suitable choice of $\mathbf{d}$. In [4], the notion of an *error-triangle* for a facet is introduced as a way of quantifying stair-stepping (Figure 2). We define the degree of stair-stepping as the maximum area, or height, of the error triangle, taken over all facets.

Volume of supports: The quantity of supports used affects both the building time and the cost. If $\mathcal{P}$ is convex, then the support volume is the volume of the region lying between $\mathcal{P}$ and the platform, i.e., the vertical polyhedral “cylinder” which is bounded below by the platform and above by the facets of $\mathcal{P}$ whose outward normals point downward. If $\mathcal{P}$ is non-convex, then the problem is more complex, since the supports for some facets may actually be attached to other facets instead of to the platform. (Figure 3 illustrates this—in 2D, for convenience.)

²There is a commercial software package called Bridgeworks for generating supports. The algorithms it uses are proprietary and it is unclear whether they optimize the supports in any way.

Figure 3: *Supports for a non-convex polygon.*

Contact area of supports: The amount of $\mathcal{P}$'s surface that is in contact with the supports affects the postprocessing time, since the supports that “stick” to $\mathcal{P}$ must be removed. If $\mathcal{P}$ is convex, then this is the total area of the downward-facing facets. If $\mathcal{P}$ is non-convex, then this is the total facet area that is in contact with the supports (it includes the areas of all downward facing facets and portions of certain upward-facing facets.)

1.3 Results

Our main results include:

1. A simple and practical algorithm for minimizing the maximum area, or height, of the error-triangles of the facets. The algorithm runs in time $O(n \log n)$, with the most involved step being merely the computation of the convex hull of a point-set in 3D. The algorithm works for any polyhedron—even one which is non-convex and has holes.
2. An $O(n^2)$ -time algorithm for computing a build direction $\mathbf{d}$ which minimizes the volume of supports needed by a convex polyhedron $\mathcal{P}$. The algorithm consists of constructing a certain arrangement on the unit-sphere, $\mathbb{S}^2$, whose regions represent directions for which the combinatorial structure of the supports is invariant. We then show how to write the volume formula for a region in a way that allows us to quickly update it in an incremental fashion as we move from region to region. Within each region, we find the best direction using the method of Lagrangian Multipliers.
3. An $O(n^2)$ -time algorithm for minimizing the surface area of a convex polyhedron $\mathcal{P}$ that is in contact with supports. This algorithm involves computing a certain arrangement on $\mathbb{S}^2$, with weighted faces and finding the lightest face. We transform the problem to the plane and solve it using topological sweep [12]. We also give a faster algorithm for the case where $\mathcal{P}$ is built so that an entire facet is in contact with the platform (this is usually the case in practice because it provides more stability to the part). This algorithm runs in roughly $O(n^{4/3})$ time (ignoring polylog factors), and is based on transforming the supports problem to a halfplane range counting problem on weighted points in 2D.
4. Evidence that the quadratic algorithms in (2) and (3) above for support optimization may not be improvable. We show that a related problem belongs to the class of *3SUM-hard problems* [14] for which no subquadratic algorithms have been found despite much effort. Specifically,

we show that, for a convex polyhedron, it is 3SUM-hard to find a build direction $\mathbf{d}$ with a positive z -component which minimizes the number of facets needing support.

5. Efficient support optimization algorithms for non-convex polygons. For an n -vertex polygon, we show that a build direction which minimizes the contact-length (resp. the area) of the supports can be found in time $O(n \log n + np(n))$ (resp. $O(n^2 + nq(n))$), where $p(n)$ and $q(n)$ are the times it takes to minimize certain polynomials of degree $\Theta(n)$ over the real line. (“Area” and “contact-length”, as used here, are the 2D counterparts of “volume” and “contact area”.) If the polygon’s sides have only a constant number of different orientations, then the time bounds improve to $O(n \log n)$ and $O(n^2)$, respectively. These results are based on visibility and ray-shooting techniques and are quite intricate.

We remark that if a planar polygon is built in a direction that is normal to the plane of the polygon, then no supports are needed. Our motivation for studying the polygon problem for build directions that are co-planar with the polygon was to develop techniques that would be applicable to non-convex polyhedra. In principle, our 2D techniques appear to carry over to 3D, but it is not clear at this point how efficient or practical they would be. We are investigating this problem further.

1.4 Prior work

Surprisingly little work has been done by way of efficient geometric algorithms for layered manufacturing. In [3], efficient algorithms are given for deciding if a part can be made by stereolithography without using supports. The problem of generating contact-area-optimal supports is considered in [1] and a heuristic yielding an approximate solution is given, but without any analysis of the running time or the quality of the approximation. The issue of support generation is also addressed in [13], in the context of an expert system, while heuristics are presented in [4, 10] for improving the accuracy and finish of the part.

The rest of the paper is organized as follows: Sections 2–5 present results 1–4 above, one per section. Sections 6 and 7 contain the algorithms in result 5 for minimizing the contact-length and area of supports of non-convex polygons. We conclude in Section 8.

2 Minimizing stair-stepping

Recall our problem: “Given a polyhedron $\mathcal{P}$ with n vertices, find a direction $\mathbf{d}$ which minimizes the maximum error-triangle area or height”. (Figure 2.) Let L denote the slice thickness and let $A_f(\mathbf{d})$ and $h_f(\mathbf{d})$ denote, respectively, the area and height of the error-triangle, $t_f(\mathbf{d})$, for facet f . Let $\theta'_f(\mathbf{d})$ be the angle between $\mathbf{d}$ and the normal, $\mathbf{n}_f$, to f , and let $\theta''_f(\mathbf{d})$ be the angle between $\mathbf{d}$ and $-\mathbf{n}_f$. Let $\theta_f(\mathbf{d}) = \min(\theta'_f(\mathbf{d}), \theta''_f(\mathbf{d}))$. It is easy to check that $A_f(\mathbf{d}) = L^2 \cot \theta_f(\mathbf{d})/2$ and $h_f(\mathbf{d}) = L \cos \theta_f(\mathbf{d})$. We seek a direction $\mathbf{d}$ such that $\max_f A_f(\mathbf{d})$ is minimized, i.e., $\min_f \theta_f(\mathbf{d})$ is maximized. Note that the optimal direction for the area problem is also optimal for the height problem since $h_f(\mathbf{d})$ (like $A_f(\mathbf{d})$) is inversely proportional to $\theta_f(\mathbf{d})$.

Consider the set $S = \{\mathbf{n}_f \cap \mathbb{S}^2, -\mathbf{n}_f \cap \mathbb{S}^2 \mid f \text{ is a facet of } \mathcal{P}\}$. That is, S consists of the points where the facet normals and their negations intersect the unit-sphere $\mathbb{S}^2$. Note that S has $O(n)$ points or *sites*. We wish to find a direction $\mathbf{d}$, i.e., a point $\mathbf{d}$ on $\mathbb{S}^2$, such that the minimum angle between it and the sites is maximized. Define a *cap* on $\mathbb{S}^2$, with *pole* $\mathbf{d}$ and *radius* θ , as the set of all points on $\mathbb{S}^2$ that are at a distance of at most θ from $\mathbf{d}$, as measured along the surface of $\mathbb{S}^2$.

Clearly, our problem is equivalent to finding the largest radius cap which does not contain any site in its interior, i.e., a largest empty cap; the pole of this cap is the desired optimal direction. The following properties of caps are easily shown:

1. Let c be the circle bounding a cap C and let $H(C)$ be the plane such that $c = H(C) \cap \mathbb{S}^2$. If C is empty, then all the sites in S lie on the same side of $H(C)$. Conversely, every such plane which intersects $\mathbb{S}^2$ corresponds to an empty cap.
2. The larger C is, the closer is $H(C)$ to the origin.
3. A largest empty cap must have at least three sites on its boundary.

By 1 and 2 above, we need to find a plane that is (a) closest to the origin and (b) has all the sites on one side of it. Let $CH(S)$ be the convex hull of S . By 3, it suffices to consider only the facets of $CH(S)$ when searching for a candidate plane. This follows because the plane containing any facet of $CH(S)$ must contain at least three sites and, moreover, all the sites of S lie on one side of this plane; on the other hand, the plane containing three or more co-planar sites that are not all on a facet of $CH(S)$ will have sites on both sides of it.

Therefore, our algorithm is as follows: We first compute the set S and then compute $CH(S)$. For each facet of $CH(S)$, we determine the plane containing the facet and find the one closest to the origin. We then compute the normal from the origin to this closest plane. The desired optimal direction $\mathbf{d}$ is the intersection of this normal and $\mathbb{S}^2$. The overall time is dominated by the $O(n \log n)$ time for the convex hull computation.

Theorem 2.1 *Let P be an n -vertex polyhedron. A direction $\mathbf{d}$ which minimizes the maximum error-triangle area, or height, can be found in time $O(n \log n)$.* ■

Remark 2.1 The algorithm makes no assumptions about $\mathcal{P}$ and hence works for any polyhedron. Moreover, it is simple and practical. In fact, we were able to implement a preliminary version of the algorithm in a matter of a few hours using public-domain software for the convex hull computation. (We used the Qhull program [5].) We have tested the algorithm on an actual STL datafile consisting of about 17,000 facets. We plan to refine the implementation and conduct further tests.

3 Minimum-volume supports for a convex polyhedron

For a build direction $\mathbf{d}$, we call the facets of $\mathcal{P}$ whose outward normals make an angle greater than $\pi/2$ with $\mathbf{d}$ the *back facets* of $\mathcal{P}$. Let us call the vertex v of $\mathcal{P}$ that is farthest away in direction $-\mathbf{d}$ the *extreme vertex* of $\mathcal{P}$. Thus, when $\mathcal{P}$ is built in direction $\mathbf{d}$, v rests on the platform and the facets requiring support are the back facets. The support volume is the volume of the polyhedron which is bounded below by the platform, above by the back facets, and on the sides by vertical facets that contain the edges on the boundary of the union of the back facets. (See Figure 4.)

Our approach consists of partitioning 3-space into $O(n^2)$ regions, R , such that the back facets and the extreme vertex are the same for all directions $\mathbf{d} \in R$. This partition can be represented as an arrangement $\mathcal{A}$ on the unit-sphere $\mathbb{S}^2$. We generate a formula for the total support volume w.r.t. any $\mathbf{d} \in R$ and set up an optimization problem for finding the build direction which minimizes the total support volume within R . We solve this problem, in time $O(|R|)$, by using the Lagrangian Multipliers method [19] and by exploiting the convexity of R . This implies that the total time for all regions is $O(n^2)$.

Figure 4: A back facet f of $\mathcal{P}$ and the corresponding support polyhedron $\mathcal{P}_f$. (The entire polyhedron $\mathcal{P}$ is itself not shown.)

However, we must also set up the volume formula within each region R . Doing this in the straightforward way takes $O(n)$ time per region, hence $O(n^3)$ time overall. We circumvent this by visiting the regions of the arrangement in a certain order and updating the formula incrementally. For this, we rewrite the formula as two parts—one based mainly on the extreme vertex v for R and the other based on the back facets. We then show that after doing an $O(n^2)$ -time precomputation, the formula in this new form can be updated incrementally in $O(n^2)$ total time as we visit the regions of $\mathcal{A}$.

Here is our approach in more detail: Following McKenna and Seidel [21], we take for each facet $f \in \mathcal{P}$, the plane h_f which is parallel to f and passes through the origin. The planes h_f partition 3-space into unbounded polyhedral regions called *cones*, each with its apex at the origin and such that for all directions within a cone the set of back facets is the same. We can represent these cones on $\mathbb{S}^2$ as the arrangement, $\mathcal{A}'$, of the great circles $h_f \cap \mathbb{S}^2$, i.e., each cone is in 1-1-correspondence with a region of $\mathcal{A}'$. Note that $\mathcal{A}'$ is composed of arcs of great circles, has size $O(n^2)$, and can be computed in time $O(n^2)$ using the algorithm given in [9]. It is obtained in a canonical form, where the edges incident at each vertex are in sorted order around the vertex. This allows the boundary of each face of $\mathcal{A}$ to be retrieved in time linear in its size.

Next using the algorithm of Bose *et al.* [6], we compute a second arrangement, $\mathcal{A}''$, of great circle arcs on $\mathbb{S}^2$ such that all directions within a region of $\mathcal{A}''$ have the same extreme vertex. As shown in [6], the directions for which a particular vertex $v \in \mathcal{P}$ is the extreme vertex can be obtained as follows: Translate $\mathcal{P}$ such that v is at the origin. For each neighbor vertex w of v , let h_{vw} be the plane normal to vw and passing through v . Let H_{vw} be the closed halfspace bounded by h_{vw} such that it contains vw . Then the set of directions for which v is the extreme vertex is determined by the intersection of all the H_{vw} 's. So we can compute a region r of $\mathcal{A}''$ in time $O(|r| \log |r|)$ and represent it on $\mathbb{S}^2$ as a polygon composed of arcs of great circles. Note that $|r|$ is equal to the number of vertices w that are neighbors of v . Thus we can compute all the regions of $\mathcal{A}''$ in time $O(n \log n)$ and, by sorting the edges at all the vertices in additional $O(n \log n)$ time, we

can obtain it in the above canonical form.

The desired arrangement, $\mathcal{A}$, is the overlay of $\mathcal{A}'$ and $\mathcal{A}''$ and can be obtained in canonical form in $O(n^2)$ time as follows: The edges of $\mathcal{A}''$ are arcs of great circles. It is convenient to think of the great circles of $\mathcal{A}'$ as being colored red and the great arcs of $\mathcal{A}''$ as being colored blue. We extend each blue arc of $\mathcal{A}''$ to the corresponding great circle, where the extensions are colored light blue. We compute the arrangement, $\mathcal{B}$, of the set of $O(n)$ great circles from $\mathcal{A}'$ and $\mathcal{A}''$. $\mathcal{B}$ is obtained in canonical form and its edges are colored red or blue or light blue. We can obtain $\mathcal{A}$ from $\mathcal{B}$ by deleting the light blue edges. Towards this end, we do a depth-first search of $\mathcal{B}$. During the search, whenever we traverse a light blue edge for the second (i.e., last) time, we delete it. In the resulting arrangement, all vertices will have even degree greater than or equal to zero. Vertices of degree zero are the ones in $\mathcal{B}$ that had only light blue edges incident to them; these vertices are deleted. Vertices of degree two must have two red or two blue edges incident to them; these vertices are deleted and the incident edges replaced by a single edge of the same color. Vertices of higher degree are left alone. It is clear that the resulting arrangement $\mathcal{A}$ is indeed in canonical form and that the total time to construct it is $O(n^2)$.

Note that we also need to know for each region $R \in \mathcal{A}$ the corresponding set of back facets and the extreme vertex. Rather than store this explicitly, which would take $O(n^3)$ total space, we compute them incrementally when we update the volume formula, as described in Section 3.4.

3.1 Generating the volume formula

Given a region $R \in \mathcal{A}$, we determine a formula for the volume, $V(R)$, of the supports required by $\mathcal{P}$ for any direction $\mathbf{d} \in R$. Let $\mathbf{d} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ be any unit-vector within R . Let $\mathbf{v}$ be the extreme vertex for R . Consider one of the back facets, say, f . Let the vertices of the facet be $P_0, P_1, \dots, P_{m-1}$, in counterclockwise order looking from outside the polyhedron. Let this facet project onto convex polygon $P'_0, P'_1, \dots, P'_{m-1}$ on the plane which passes through $\mathbf{v}$ and is normal to direction $\mathbf{d}$. The volume of the supports needed by f is then the volume of the polyhedron $\mathcal{P}_f$ shown in Figure 4. We have,

$$\mathbf{P}_i - \mathbf{P}'_i = k_i \mathbf{d},$$

and

$$(\mathbf{P}'_i - \mathbf{v}) \cdot \mathbf{d} = 0, \tag{1}$$

for some constant k_i . From the above equations, we find that $\mathbf{P}'_i = \mathbf{P}_i - k_i \mathbf{d}$, where k_i is given by

$$k_i = (\mathbf{P}_i - \mathbf{v}) \cdot \mathbf{d}.$$

Let the facets of $\mathcal{P}_f$ be $S_0, S_1, \dots, S_{m+1}$, where S_0 is the top facet, S_1 is the bottom facet, and S_i ($2 \leq i \leq m+1$) is a side facet, i.e., $S_i = P_{i-2}P_{i-1}P'_{i-1}P'_{i-2}$. Let $\mathbf{Q}_j$ be any point on S_j , and let $\mathbf{N}_j$ be a unit outward-normal vector for S_j (we mean “outward” w.r.t. $\mathcal{P}_f$). Then, using the formula in [15], the volume, $V_f(R)$, of $\mathcal{P}_f$ is given by

$$V_f(R) = \frac{1}{3} \sum_j (\mathbf{Q}_j \cdot \mathbf{N}_j) \text{Area}(S_j).$$

Let $V_{if}(R)$ be the volume contributed by S_i to $V_f(R)$, $0 \leq i \leq m+1$. Then,

$$V_{0f}(R) = \frac{1}{3}(\mathbf{P}_0 \cdot (-\mathbf{n}_f)) \left(\frac{1}{2}(\mathbf{n}_f \cdot \left(\sum_{i=0}^{m-1} \mathbf{P}_i \times \mathbf{P}_{i+1} \right)) \right),$$

where $-\mathbf{n}_f$ is the unit outward-normal for facet f ("outward" w.r.t. $\mathcal{P}_f$). (For convenience, we take $P_m = P_0$ and $P'_m = P'_0$.) We call $\sum_{i=0}^{m-1} \mathbf{P}_i \times \mathbf{P}_{i+1}$ the *area term* and denote it by Δ_f . Thus,

$$V_{0f}(R) = -\frac{1}{6}(\mathbf{P}_0 \cdot \mathbf{n}_f)(\mathbf{n}_f \cdot \Delta_f).$$

Next, we have

$$V_{1f}(R) = \frac{1}{3}(\mathbf{P}'_0 \cdot (-\mathbf{d})) \left(\frac{1}{2}(-\mathbf{d} \cdot \left(\sum_{i=0}^{m-1} \mathbf{P}'_i \times \mathbf{P}'_{i+1} \right)) \right).$$

Now $\mathbf{P}'_0 \cdot \mathbf{d} = \mathbf{v} \cdot \mathbf{d}$ (from equation (1)) and

$$\begin{aligned} \mathbf{d} \cdot (\mathbf{P}'_i \times \mathbf{P}'_{i+1}) &= \mathbf{d} \cdot ((\mathbf{P}_i - k_i \mathbf{d}) \times (\mathbf{P}_{i+1} - k_{i+1} \mathbf{d})) \\ &= \mathbf{d} \cdot (\mathbf{P}_i \times \mathbf{P}_{i+1} - k_{i+1} \mathbf{P}_i \times \mathbf{d} + k_i \mathbf{P}_{i+1} \times \mathbf{d}) \\ &= \mathbf{d} \cdot (\mathbf{P}_i \times \mathbf{P}_{i+1}). \end{aligned}$$

Therefore,

$$V_{1f}(R) = \frac{1}{6}(\mathbf{d} \cdot \mathbf{v})(\mathbf{d} \cdot \left(\sum_{i=0}^{m-1} \mathbf{P}_i \times \mathbf{P}_{i+1} \right)),$$

which can be written as

$$V_{1f}(R) = \frac{1}{6}(\mathbf{d} \cdot \mathbf{v})(\mathbf{d} \cdot \Delta_f).$$

Recall that S_i , ($2 \leq i \leq m+1$) is the quadrilateral formed by $P_{i-2}, P_{i-1}, P'_{i-1}, P'_{i-2}$. The unit outward-normal to S_i is $\mathbf{n}_i = \frac{\mathbf{d} \times (\mathbf{P}_{i-1} - \mathbf{P}_{i-2})}{|\mathbf{d} \times (\mathbf{P}_{i-1} - \mathbf{P}_{i-2})|}$. The volume, $V_{if}(R)$, contributed by S_i is given by

$$V_{if}(R) = \frac{1}{3}(\mathbf{P}_{i-2} \cdot \mathbf{n}_i) \left(\frac{1}{2}(\mathbf{n}_i \cdot (\mathbf{P}_{i-2} \times \mathbf{P}_{i-1} + \mathbf{P}_{i-1} \times \mathbf{P}'_{i-1} + \mathbf{P}'_{i-1} \times \mathbf{P}'_{i-2} + \mathbf{P}'_{i-2} \times \mathbf{P}_{i-2})) \right).$$

Now,

$$\mathbf{P}_{i-2} \cdot \mathbf{n}_i = \mathbf{P}_{i-2} \cdot \frac{\mathbf{d} \times (\mathbf{P}_{i-1} - \mathbf{P}_{i-2})}{|\mathbf{d} \times (\mathbf{P}_{i-1} - \mathbf{P}_{i-2})|} = -\frac{\mathbf{d} \cdot (\mathbf{P}_{i-2} \times (\mathbf{P}_{i-1} - \mathbf{P}_{i-2}))}{|\mathbf{d} \times (\mathbf{P}_{i-1} - \mathbf{P}_{i-2})|} = -\frac{\mathbf{d} \cdot (\mathbf{P}_{i-2} \times \mathbf{P}_{i-1})}{|\mathbf{d} \times (\mathbf{P}_{i-1} - \mathbf{P}_{i-2})|}.$$

Also,

$$\begin{aligned} \mathbf{P}_{i-1} \times \mathbf{P}'_{i-1} &= \mathbf{P}_{i-1} \times (\mathbf{P}_{i-1} - k_{i-1} \mathbf{d}) \\ &= k_{i-1}(\mathbf{d} \times \mathbf{P}_{i-1}) \\ \mathbf{P}'_{i-1} \times \mathbf{P}'_{i-2} &= (\mathbf{P}_{i-1} - k_{i-1} \mathbf{d}) \times (\mathbf{P}_{i-2} - k_{i-2} \mathbf{d}) \\ &= \mathbf{P}_{i-1} \times \mathbf{P}_{i-2} + k_{i-2} \mathbf{d} \times \mathbf{P}_{i-1} - k_{i-1} \mathbf{d} \times \mathbf{P}_{i-2} \\ \mathbf{P}'_{i-2} \times \mathbf{P}_{i-2} &= (\mathbf{P}_{i-2} - k_{i-2} \mathbf{d}) \times \mathbf{P}_{i-2} \\ &= -k_{i-2} \mathbf{d} \times \mathbf{P}_{i-2}. \end{aligned}$$

So we can write,

$$\begin{aligned}
V_{if}(R) &= \frac{1}{3} \left(-\frac{\mathbf{d} \cdot (\mathbf{P}_{i-2} \times \mathbf{P}_{i-1})}{|\mathbf{d} \times (\mathbf{P}_{i-1} - \mathbf{P}_{i-2})|} \right) \left(\frac{\mathbf{d} \times (\mathbf{P}_{i-1} - \mathbf{P}_{i-2})}{|\mathbf{d} \times (\mathbf{P}_{i-1} - \mathbf{P}_{i-2})|} \cdot \frac{k_{i-2} + k_{i-1}}{2} (\mathbf{d} \times (\mathbf{P}_{i-1} - \mathbf{P}_{i-2})) \right) \\
&= \frac{1}{6} (\mathbf{d} \cdot (\mathbf{P}_{i-1} \times \mathbf{P}_{i-2})) (k_{i-2} + k_{i-1}) \\
&= \frac{1}{6} (\mathbf{d} \cdot (\mathbf{P}_{i-1} \times \mathbf{P}_{i-2})) (\mathbf{d} \cdot (\mathbf{P}_{i-2} + \mathbf{P}_{i-1} - 2\mathbf{v})) \\
&= \frac{1}{6} (\mathbf{d} \cdot (\mathbf{P}_{i-1} \times \mathbf{P}_{i-2})) (\mathbf{d} \cdot (\mathbf{P}_{i-2} + \mathbf{P}_{i-1})) + \frac{1}{3} (\mathbf{d} \cdot (\mathbf{P}_{i-2} \times \mathbf{P}_{i-1})) (\mathbf{d} \cdot \mathbf{v}).
\end{aligned}$$

Therefore,

$$\begin{aligned}
V_f(R) &= \sum_{i=0}^{m+1} V_{if}(R) \\
&= V_{0f}(R) + V_{1f}(R) + \sum_{i=2}^{m+1} V_{if}(R) \\
&= -\frac{1}{6} (\mathbf{P}_0 \cdot \mathbf{n}_f) (\mathbf{n}_f \cdot \Delta_f) + \frac{1}{6} (\mathbf{d} \cdot \mathbf{v}) (\mathbf{d} \cdot \Delta_f) + \\
&\quad \frac{1}{6} \sum_{i=1}^m (\mathbf{d} \cdot (\mathbf{P}_i \times \mathbf{P}_{i-1})) (\mathbf{d} \cdot (\mathbf{P}_i + \mathbf{P}_{i-1})) + \frac{1}{3} (\mathbf{d} \cdot \Delta_f) (\mathbf{d} \cdot \mathbf{v}) \\
&= -\frac{1}{6} (\mathbf{P}_0 \cdot \mathbf{n}_f) (\mathbf{n}_f \cdot \Delta_f) + \frac{1}{6} \sum_{i=1}^m (\mathbf{d} \cdot (\mathbf{P}_i \times \mathbf{P}_{i-1})) (\mathbf{d} \cdot (\mathbf{P}_i + \mathbf{P}_{i-1})) + \frac{1}{2} (\mathbf{d} \cdot \Delta_f) (\mathbf{d} \cdot \mathbf{v}) \\
&= V'_f(R) + \frac{1}{2} (\mathbf{d} \cdot \Delta_f) (\mathbf{d} \cdot \mathbf{v}),
\end{aligned}$$

where $V'_f(R)$ denotes the part of $V_f(R)$ which is independent of $\mathbf{v}$. As we will see in Section 3.4, being able to decompose the formula in this way is crucial to the running time of the algorithm. Therefore, the total support volume, $V(R) = \sum_f V_f(R)$, associated with region $R \in \mathcal{A}$ is:

$$\begin{aligned}
V(R) &= \sum_f V'_f(R) + \frac{1}{2} \sum_f (\mathbf{d} \cdot \Delta_f) (\mathbf{d} \cdot \mathbf{v}) \\
&= \sum_f V'_f(R) + \frac{1}{2} (\mathbf{d} \cdot \sum_f \Delta_f) (\mathbf{d} \cdot \mathbf{v}) \\
&= \sum_f V'_f(R) + \frac{1}{2} (\mathbf{d} \cdot \Delta(R)) (\mathbf{d} \cdot \mathbf{v}),
\end{aligned}$$

where $\Delta(R) = \sum_f \Delta_f$ is the total area term for all the back facets associated with R .

3.2 The optimization problem

When we expand $V(R)$ we get an expression of the form $Ax^2 + By^2 + Cz^2 + Dxy + Eyz + Fxz + G$, where $A, B, \dots, G$ are constants depending only on the P_i 's of the different back facets of R and on the extreme vertex v of R .

Let $\mathbf{d}_R \in \mathbb{S}^2$ be a given point in R 's interior. ($\mathbf{d}_R$ is computed when $\mathcal{A}$ is constructed.) For any great arc a bounding R , let $h(a)$ be the plane containing the corresponding great circle and let $\mathbf{n}_{h(a)}$ be a unit-normal for it. Note that $\mathbf{d}$ is in the interior of R if and only if $\mathbf{d} \cdot \mathbf{n}_{h(a)}$ and $\mathbf{d}_R \cdot \mathbf{n}_{h(a)}$ are both positive or both negative for each great arc a bounding R . Thus our optimization problem within each region, R , is:

$$\text{Minimize } f(x, y, z) = Ax^2 + By^2 + Cz^2 + Dxy + Eyz + Fxz$$

$$\text{Subject to } x^2 + y^2 + z^2 = 1 \quad (\text{Sphere Constraint.})$$

$$\mathbf{d} \cdot \mathbf{n}_{h(a)} \geq 0 \text{ (resp. } \leq 0) \text{ if } \mathbf{d}_R \cdot \mathbf{n}_{h(a)} \geq 0 \text{ (resp. } \leq 0) \text{ for each great arc } a \text{ bounding } R. \quad (\text{Plane Constraints.})$$

3.3 Solving the optimization problem

We use the method of Lagrangian Multipliers [19]. The optimization problem for a region R has $|R| + 1$ constraints, and, in general, could take $O(2^{|R|})$ time to solve. But in our case R is convex and this leads to an $O(|R|)$ -time solution.

We proceed in three stages: (i) We first keep only the sphere constraint active and find the extreme points (i.e., minimum or maximum) over all of $\mathbb{S}^2$. (ii) Next, we take some arc a bounding R and make the corresponding plane constraint active as well. This gives extreme points lying on a 's great circle. We repeat this for each great arc a bounding R . (iii) Finally, we consider arcs a and a' meeting at a vertex v of R and make the corresponding plane constraints active—thus making v an extreme point. Note that it is not necessary to make more than two plane constraints active since there is no point of R that is common to three great circles.

1. All plane constraints are inactive: The Lagrangian is $L(x, y, z, \lambda) = f(x, y, z) + \lambda(1 - x^2 - y^2 - z^2)$, for some parameter λ . The partial derivatives of L , w.r.t. each of x , y , and z , must be zero at an extreme (i.e., minimum or maximum) point. This yields three linear equations in x , y , and z . The values of λ for which these three equations have non-trivial solutions can be found by solving a cubic equation in λ , given by:

$$\begin{vmatrix} 2A - 2\lambda & D & F \\ D & 2B - 2\lambda & E \\ F & E & 2C - 2\lambda \end{vmatrix} = 0.$$

For each such real-valued λ (there are at most three of them) we solve for x , y , and z , using any two of the three linear equations (the remaining one will depend on the two chosen) and the sphere constraint. This will yield (i) two antipodal points on $\mathbb{S}^2$, or (ii) a great circle (if the three equations are the same but not identically zero), or (iii) all of $\mathbb{S}^2$ (if the three equations are identically zero). We can ignore cases (ii) and (iii) since, anyway, we will be covering them in the cases below. If case (i) holds then we check if either of the two points lies in R (by checking the plane constraints) and, if so, we use this point as a candidate for the minimum value of $f(x, y, z)$.

2. One plane constraint is active: Let this plane be defined by $ax + by + cz = 0$. Then the Lagrangian is $L(x, y, z, \lambda_1, \lambda_2) = f(x, y, z) + \lambda_1(1 - x^2 - y^2 - z^2) + \lambda_2(ax + by + cz)$, for some parameters λ_1 and λ_2 . Setting partial derivatives to zero gives three linear equations in x ,

y , z , and λ_2 . Using these equations and the equation $ax + by + cz = 0$ we can compute the values of λ_1 that yield non-trivial solutions, this time by solving a quadratic equation in λ_1 , as given by:

$$\begin{vmatrix} 2A - 2\lambda_1 & D & F & a \\ D & 2B - 2\lambda_1 & E & b \\ F & E & 2C - 2\lambda_1 & c \\ a & b & c & 0 \end{vmatrix} = 0.$$

We can now eliminate λ_2 using one of the linear equations. Using the sphere constraints, the constraint $ax + by + cz = 0$, and any one of the remaining linear equations, we proceed to compute the extreme points and check for feasibility.

3. Two plane constraints are active: In this case we only need to consider the vertices of R , since these are the only points that are common to two great circles bounding R and are in the feasible region.

Analysis: It is reasonable to assume that the cubic and quadratic equations that arise can be solved in $O(1)$ time. Thus, the optimization problem for R takes time $O(|R|)$. Summed over all regions, this is $O(n^2)$.

3.4 Updating the volume formula incrementally

We precompute Δ_f for each facet $f \in \mathcal{P}$. Then, we pick an initial region, $R_0 \in \mathcal{A}$, and compute its back facets, its extreme vertex, and $V(R_0)$ and $\Delta(R_0)$. Clearly, all this can be done in $O(n)$ time. Next, we compute the dual graph, $\hat{\mathcal{A}}$, of $\mathcal{A}$, by placing a vertex in each region of $\mathcal{A}$ and joining two vertices by an edge if the corresponding regions share an edge. We then visit the regions of $\mathcal{A}$ in the order given by a depth-first search of $\hat{\mathcal{A}}$ which starts at the vertex corresponding to R_0 . Suppose that the search moves from region R to region R' . There are three cases:

1. A facet f which was a back facet for R ceases to be a back facet for R' .
2. A facet f which was not a back facet for R becomes a back facet for R' .
3. The extreme vertex v changes to v' .

Note that in cases 1 and 2, the extreme vertex remains unchanged while in case 3 the set of back facets remains unchanged. This follows from the way we constructed $\mathcal{A}$. Also note that cases 1 and 2 may occur simultaneously if $\mathcal{P}$ has parallel facets. In this case, we handle them one after the other, as described below.

In case 1, we obtain $V(R')$ from $V(R)$ in time $O(|f|)$, as follows: We first update the term $\frac{1}{2}(\mathbf{d} \cdot \Delta(R))(\mathbf{d} \cdot \mathbf{v})$ by subtracting $\frac{1}{2}(\mathbf{d} \cdot \Delta_f)(\mathbf{d} \cdot \mathbf{v})$. This can be done in $O(1)$ time since we have precomputed Δ_f . Then, in $O(|f|)$ time, we subtract $V_f(R)$ from $V(R)$. Case 2 is handled similarly.

In case 3, we update $V(R)$ to $V(R')$ in $O(1)$ time by subtracting $\frac{1}{2}(\mathbf{d} \cdot \Delta(R))(\mathbf{d} \cdot \mathbf{v})$ and adding $\frac{1}{2}(\mathbf{d} \cdot \Delta(R'))(\mathbf{d} \cdot \mathbf{v}')$. (Note that, in this case, $\Delta(R') = \Delta(R)$ and is already known.) We also set v' to be the extreme vertex of R' .

How many times can a facet f appear or disappear as a back facet? This will happen each time that we cross, in the depth-first search, any edge of $\mathcal{A}$ which is contained in the great circle

$h_f \cap \mathbb{S}^2$. Since there are $O(n)$ such edges and each is crossed only once, we conclude that f appears or disappears as a back facet $O(n)$ times. Therefore the total update time in case 1 and 2 is $O(\sum_{f \in \mathcal{P}} n|f|) = O(n^2)$. The total time spent in case 3 is clearly $O(n^2)$. We may now conclude:

Theorem 3.1 *Let $\mathcal{P}$ be a convex polyhedron of n vertices. A build direction which minimizes the total volume of supports needed by $\mathcal{P}$ can be computed in time $O(n^2)$.* ■

4 Minimum-contact-area supports for a convex polyhedron

A facet $f \in \mathcal{P}$ needs support w.r.t. a direction $\mathbf{d}$ iff it is a back facet. Thus, the set of directions for which f needs support can be represented on the unit-sphere, $\mathbb{S}^2$, by an open hemisphere, h_f , whose pole is the point $-\mathbf{n}_f$ on $\mathbb{S}^2$.

We associate with h_f a weight equal to the area of f . Clearly, our problem is now equivalent to finding a point on $\mathbb{S}^2$ which is covered by hemispheres, h_f , of minimum total weight. We proceed as follows:

W.l.o.g. assume that the bounding plane of no hemisphere h_f is parallel to the xy -plane; this can be enforced in $O(n)$ time by rotating $\mathcal{P}$ about the x - or y -axis. Let $\mathbb{S}_+^2$ be the portion of $\mathbb{S}^2$ lying above the xy -plane and let $h_f^+ = h_f \cap \mathbb{S}_+^2$. Using *central projection* [22], we map h_f^+ to an open halfplane, ℓ_f^+ , of the same weight, on the plane $z = 1$. Under this map, there is a 1-1 correspondence between the faces in the arrangement of the h_f^+ 's and the faces in the arrangement of the ℓ_f^+ 's. Thus, we can now solve our problem by finding a face in the latter arrangement which is covered by halfplanes of minimum total weight.

In [12, pages 181–182], Edelsbrunner and Guibas consider a similar problem: Given r doublewedges in the plane, find a point which is covered by the maximum number of doublewedges. Using topological sweep, they solve this problem in $O(r^2)$ time and $O(r)$ space. We can use this approach, with a minor modification to handle the halfplane weights, to solve our problem in $O(n^2)$ time and $O(n)$ space (since $r = O(n)$ in our case). We handle the portions of the hemispheres lying below the xy -plane in a symmetric way.

Theorem 4.1 *Let $\mathcal{P}$ be an n -vertex convex polyhedron. A direction $\mathbf{d}$ which minimizes the total surface area of $\mathcal{P}$ in contact with supports can be found in time $O(n^2)$, using $O(n)$ space.* ■

4.1 A faster algorithm when building $\mathcal{P}$ on a facet

To build $\mathcal{P}$ on a facet f , we must choose $-\mathbf{n}_f$ as the build direction. Let h_f be the closed hemisphere on $\mathbb{S}^2$ whose pole is the point $-\mathbf{n}_f$. Then, a facet $f' \neq f$ will need support iff $\mathbf{n}_{f'}$ is not contained in h_f . Let $C = \{\mathbf{n}_f | f \text{ is a facet of } \mathcal{P}\}$ and let $C' = \{-\mathbf{n}_f | f \text{ is a facet of } \mathcal{P}\}$. Associate with each point $\mathbf{n}_f \in C$ a weight equal to f 's area. Thus, our problem is to find a point $-\mathbf{n}_f \in C'$ such that the total weight of the points $\mathbf{n}_{f'} \in C$ that are not in h_f is minimized, or, equivalently, the total weight of the points $\mathbf{n}_{f'} \in C$ that are in h_f is maximized.

It is convenient to reformulate our problem as follows: We are given $r = O(n)$ weighted *blue points* and r *red points* on $\mathbb{S}^2$, corresponding to the points in C and C' , respectively. Each red point is the pole of a closed *red hemisphere*. We wish to find the red hemisphere which contains blue points of maximum total weight.

Let $P = (p_1, p_2, p_3)$ be any blue point. Assume w.l.o.g. $p_2 \neq 0$ for any blue point; this can be enforced in $O(n)$ time by rotating $\mathbb{S}^2$ suitably, about the first or third axis. Let $H : z = ax + by$ be

the bounding plane of a red hemisphere; H passes through the origin. We map H to the red point $H' = (a, b)$ in the plane and map P to the blue line $P' : y = (-p_1/p_2)x + (p_3/p_2)$ in the plane.

Lemma 4.1 *Let P, H, H' , and P' be as above.*

(A): P is on or above H iff $(p_2 > 0$ and H' is on or below P') or $(p_2 < 0$ and H' is on or above $P')$.

(B): P is on or below H iff $(p_2 > 0$ and H' is on or above P') or $(p_2 < 0$ and H' is on or below $P')$.

Proof

(A): P is on or above H iff $p_3 \geq ap_1 + bp_2$; i.e., iff $(p_2 > 0$ and $b \leq (p_3/p_2) - a(p_1/p_2))$ or $(p_2 < 0$ and $b \geq (p_3/p_2) - a(p_1/p_2))$; i.e., iff $(p_2 > 0$ and H' is on or below $P')$ or $(p_2 < 0$ and H' is on or above $P')$.

The proof for (B) is similar. ■

Let us divide the red hemispheres into two sets: those that have their poles above the equator of $\mathbb{S}^2$ (called *upper red hemispheres*) and those that have their poles below (*lower red hemispheres*). Consider the upper red hemispheres.

The blue points P that are contained in any upper red hemisphere all lie on or above the plane H for that hemisphere. Therefore, by Lemma 4.1(A), our goal is to find a red point H' such that the total weight of the blue lines P' with $p_2 > 0$ that are on or above H' plus the total weight of the blue lines P' with $p_2 < 0$ that are on or below H' is maximum. (Similarly, Lemma 4.1(B) applies to the lower red hemispheres.)

We will use the following data structure, due to Matoušek, to solve the above problem.

Theorem 4.2 (Matoušek, [20]) *Let V be a set of n weighted points in $\mathbb{R}^d$, let m be a parameter such that $n \leq m \leq n^d$, let h be an integer such that $1 \leq h \leq d + 1$, and let $\delta > 0$ be any real number. In $O(n^{1+\delta} + m(\log n)^\delta)$ time, we can preprocess the points of V into a data structure of size $O(m)$ such that the total weight of the points of V lying in the intersection of any h halfspaces can be computed in time $O((n/m^{1/d})(\log \frac{m}{n})^{h-(d-h+1)/d})$. ■*

Since the above data structure is designed for halfspace range counting, we need to dualize our problem once again. By a suitable duality transform (see, for instance, [11]), we can write Lemma 4.1 as follows, where H'' is the red line dual to point H' and P'' is the blue point dual to line P' :

Lemma 4.2 (A): P is on or above H iff $(p_2 > 0$ and P'' is on or below H'') or $(p_2 < 0$ and P'' is on or above H'').

(B): P is on or below H iff $(p_2 > 0$ and P'' is on or above H'') or $(p_2 < 0$ and P'' is on or below H''). ■

Note that Lemma 4.2(A) (resp. Lemma 4.2(B)) applies to the upper (resp. lower) red hemispheres.

We now build two data structures D^+ and D^- : D^+ is the structure of Theorem 4.2 built for $d = 2$ and $h = 1$ on the blue points P'' that correspond to points P with $p_2 > 0$. D^- is a similar structure for the blue points P'' with $p_2 < 0$. Thus, these structures can be used for halfplane queries on weighted points. For each upper red hemisphere with bounding plane H , we query

D^+ with the lower halfplane of H'' , query D^- with the upper halfplane of H'' , and sum up the total weights returned. In this way, we find the upper red hemisphere that contains blue points of maximum total weight. We repeat this process with the lower red hemispheres also. This gives the overall optimal red hemisphere and thus yields an area-minimizing direction for building $\mathcal{P}$ on a facet.

Theorem 4.3 *Let $\mathcal{P}$ be an n -vertex convex polyhedron that is to be built on a facet. A build direction minimizing the total area of $\mathcal{P}$ that is in contact with supports can be found in time $O(n^{4/3}(\log n)^\gamma)$ time and space $O(n^{4/3}/(\log n)^{2\gamma})$, where $\gamma > 0$ is an arbitrarily small real number.*

Proof Correctness is clear from the discussion above. For the running time, note that the weight computations and the transformations take $O(n)$ time. For any m , $n \leq m \leq n^2$, the structure of Theorem 4.2 uses $O(m)$ space and can be built in time $O(n^{1+\delta} + m(\log n)^\delta)$. Each of the $r = O(n)$ queries takes time $O((n/m^{1/2})(\log \frac{m}{n})^{1-(2-1+1)/2}) = O(n/m^{1/2})$. Hence, the total time is $O(n^{1+\delta} + m(\log n)^\delta + n^2/m^{1/2})$, which is $O(n^{4/3}(\log n)^{\delta/3})$, if we choose $m = n^{4/3}/(\log n)^{2\delta/3}$. Setting $\gamma = \delta/3$ completes the proof. $\blacksquare$

5 3SUM-hardness of a related problem

In Sections 3 and 4, we gave $O(n^2)$ -time algorithms for minimizing the volume and contact-area of supports for a convex polyhedron. In this section, we show that a related problem belongs to the class of 3SUM-hard problems, introduced by Gajentaan and Overmars [14], which suggests that it may be difficult to improve upon the algorithms of Sections 3 and 4.

The problem 3SUM is defined as follows. Given a set S of n real numbers, decide if there are $a, b, c \in S$ such that $a + b + c = 0$. A problem PR is called 3SUM-hard if, in $o(n^2)$ time, 3SUM can be reduced to PR .

We call a direction $\mathbf{d}$ *positive* if its third coordinate is strictly larger than zero.

Theorem 5.1 *Given a convex polyhedron $\mathcal{P}$ with n vertices, the problem of finding a positive build direction which minimizes the number of facets of $\mathcal{P}$ needing support, is 3SUM-hard.*

Proof Consider the following problem PR . Given n halfplanes and an integer k , decide if there is a point that is contained in at least k halfplanes. (Note that for $k \leq n/2$, the answer is always “yes”.) In [14], it is shown that this problem is 3SUM-hard. Therefore, it suffices to show that we can reduce PR to our polyhedron problem, in $o(n^2)$ time.

Consider an instance of problem PR . It consists of m upper halfplanes U_i , $1 \leq i \leq m$, and $n - m$ lower halfplanes L_i , $1 \leq i \leq n - m$, for some integer m , together with an integer k .

For $1 \leq i \leq m$, let u_i be the bounding line of U_i , and let it be given by the equation $y = a_i x + b_i$. We map each such line u_i to the point

$$u'_i = \left(\frac{a_i}{\sqrt{1 + a_i^2 + b_i^2}}, \frac{1}{\sqrt{1 + a_i^2 + b_i^2}}, \frac{-b_i}{\sqrt{1 + a_i^2 + b_i^2}} \right)$$

on the unit-sphere. Similarly, for $1 \leq i \leq n - m$, let the bounding line l_i of L_i be given by $y = c_i x + d_i$. We map each such line to the point

$$l'_i = \left(\frac{-c_i}{\sqrt{1 + c_i^2 + d_i^2}}, \frac{-1}{\sqrt{1 + c_i^2 + d_i^2}}, \frac{d_i}{\sqrt{1 + c_i^2 + d_i^2}} \right)$$

on the unit-sphere. A point $p = (\alpha, \beta)$ in the plane is mapped to the non-vertical plane $p' : z = \alpha x - \beta y$ in $\mathbb{R}^3$ through the origin.

The following two claims are easy to verify.

1. p is contained in U_i iff u'_i is above p' .
2. p is contained in L_i iff l'_i is above p' .

For each i , $1 \leq i \leq m$, let H_i be the halfspace containing the unit-sphere and whose boundary is the tangent plane at this sphere at point u'_i . Similarly, for each i , $1 \leq i \leq n - m$, let K_i be the halfspace containing the unit-sphere and whose boundary is the tangent plane at this sphere at point l'_i . Finally, let $\mathcal{P}$ be the convex polyhedron that is the intersection of the H_i 's and K_j 's. Note that $\mathcal{P}$ has n facets, and $O(n)$ vertices.

For any build direction $\mathbf{d}$, let $G_{\mathbf{d}}$ be the plane through the origin orthogonal to $\mathbf{d}$, and let $X_{\mathbf{d}}$ be the number of facets of $\mathcal{P}$ not needing support for this build direction. If $\mathbf{d}$ is a positive direction, then $X_{\mathbf{d}}$ is equal to the total number of points in $\{u'_1, \dots, u'_m, l'_1, \dots, l'_{n-m}\}$ that are above $G_{\mathbf{d}}$. This, in turn equals the total number of halfplanes in $\{U_1, \dots, U_m, L_1, \dots, L_{n-m}\}$ that contain the planar point p for which $p' = G_{\mathbf{d}}$. (Note that $G_{\mathbf{d}}$ is non-vertical because the third coordinate of $\mathbf{d}$ is non-zero. Thus the point p exists.)

Conversely, let p be a point in the plane such that it is contained in c halfplanes belonging to the set $\{U_1, \dots, U_m, L_1, \dots, L_{n-m}\}$. Then, if $\mathbf{d}$ is the positive direction normal to the plane p' , we have $X_{\mathbf{d}} = c$. ($\mathbf{d}$ exists, because the plane p' is non-vertical, i.e., not orthogonal to the plane $z = 0$.)

To solve an instance of *PR*, i.e., to decide if there is a point p contained in at least k of the halfplanes U_i and L_j , it suffices to find a point which lies in the maximum number of these halfplanes. By the above discussion, the latter problem is equivalent to finding a positive build direction for $\mathcal{P}$ which maximizes the number of facets of $\mathcal{P}$ not needing support. Clearly, this positive direction minimizes the number of facets that do need support.

Hence, we have reduced the original problem *PR* to our polyhedron problem. Given the input of *PR*, the polyhedron $\mathcal{P}$ can be constructed in $O(n \log n)$ time, i.e., the entire reduction takes $o(n^2)$ time. ■

6 Minimum-contact-length supports for simple polygons

In this section, we consider the problem of computing a direction that minimizes the total *length* of the boundary of a simple polygon to which the support “sticks”. Our approach is similar to that of Arkin *et al.* [2], who solve a related problem.

First, we introduce some notation. Then, we give a precise definition of the problem. Finally, an algorithm for computing the optimal direction is given. Throughout this section and Section 7, we measure angles counterclockwise from 0 to 2π .

Henceforth, we let $\mathcal{P}$ be a simple n -vertex polygon. We assume that $\mathcal{P}$ is in general position, in the sense that no three vertices are collinear. (All our results remain valid for arbitrary simple polygons. The algorithms and correctness proofs, however, need some minor modifications.) For each edge e of $\mathcal{P}$, let $\mathbf{n}_e$ denote its *outer normal*, i.e., the vector that is orthogonal to e and that “points outside of $\mathcal{P}$ ”. Furthermore, let α_e denote the angle between the positive x -axis and the vector $\mathbf{n}_e$. For any direction $\mathbf{d}$, we denote by $\phi_{\mathbf{d}}$ the angle between the positive x -axis and $\mathbf{d}$.

We say that edge e *needs support for direction* $\mathbf{d}$, if the dot product $\mathbf{n}_e \cdot \mathbf{d}$ is negative. In this case, the *entire* edge e needs support.

Figure 5: Edge e has endpoints a_e and b_e . This edge is attached to supports for direction $\mathbf{d}$. The set $S_e(\mathbf{d})$ consists of two segments $a_e A_e(\mathbf{d})$ and $B_e(\mathbf{d}) b_e$. These segments are the parts of edge e that are attached to the supports. Edge f needs support for direction $\mathbf{d}$.

Even if an edge does not need support for a direction $\mathbf{d}$, it may still be a part of some support for this direction, in the following sense. Let e be an edge of $\mathcal{P}$, and let $\mathbf{d}$ be a direction, such that $\mathbf{n}_e \cdot \mathbf{d} \geq 0$. Let q be a point on e , such that the ray emanating from q in the direction $\mathbf{d}$ intersects the interior of the polygon $\mathcal{P}$. Then we say that q is *attached to a support for direction $\mathbf{d}$* . Let $S_e(\mathbf{d})$ be the set of all points on e that are attached to a support for direction $\mathbf{d}$. (See Figure 5.)

Lemma 6.1 *The set $S_e(\mathbf{d})$ is (i) empty, (ii) consists of one segment on e , one of whose endpoints is an endpoint of e , (iii) consists of two segments on e , and each segment has an endpoint which is an endpoint of e , or (iv) is equal to the entire edge e .*

Proof The set $S_e(\mathbf{d})$ consists of line segments on edge e . It suffices to prove that no such segment has both its endpoints in the interior of e .

Assume there is such a segment, and let s and t be its endpoints. Let a_e and b_e be the endpoints of edge e . Assume w.l.o.g. that a_e, s, t , and b_e appear in this order from left to right along e , and that the direction $\mathbf{d}$ is vertically upwards. There is a point x (resp. y) on e , strictly between a_e and s (resp. t and b_e), such that the ray r_x (resp. r_y) emanating from x (resp. y) in the direction $\mathbf{d}$ does not intersect the polygon $\mathcal{P}$. On the other hand, the ray emanating from s in the direction $\mathbf{d}$ does intersect $\mathcal{P}$; let z be the first intersection. By walking along the boundary of $\mathcal{P}$, from a_e to z , we intersect one of the rays r_x and r_y . This is a contradiction. ■

We define $l_e(\mathbf{d})$ as the length of edge e , in case e needs support for direction $\mathbf{d}$. Otherwise, $l_e(\mathbf{d})$ is defined as the length of the at most two segments on e that are determined by $S_e(\mathbf{d})$. Hence, $\sum_e l_e(\mathbf{d})$ is equal to the total length of the boundary of $\mathcal{P}$ to which the supports stick, when $\mathcal{P}$ is manufactured along direction $\mathbf{d}$. Therefore, we want to solve the following problem.

Problem 6.1 *Given a simple polygon $\mathcal{P}$, compute a direction $\mathbf{d}$ which minimizes*

$$L_{\mathcal{P}}(\mathbf{d}) := \sum_e l_e(\mathbf{d}).$$

We need some more notation. (Refer to Figure 5.) Let e be an edge of $\mathcal{P}$, and let $\mathbf{d}$ be a direction such that $S_e(\mathbf{d})$ consists of two segments. Let a_e and b_e denote the two endpoints of e , where a_e is to the left of b_e (w.r.t. direction $\mathbf{d}$), and let $A_e(\mathbf{d})$ and $B_e(\mathbf{d})$ denote the other two endpoints of the segments spanning $S_e(\mathbf{d})$. That is, $S_e(\mathbf{d})$ consists of the two segments $a_e A_e(\mathbf{d})$ and $b_e B_e(\mathbf{d})$. Let $A'_e(\mathbf{d})$ be the vertex of $\mathcal{P}$ that is hit first by shooting a ray from $A_e(\mathbf{d})$ in the direction $\mathbf{d}$. (Note that this ray indeed hits a vertex.) Define $B'_e(\mathbf{d})$ similarly w.r.t. $B_e(\mathbf{d})$. Note that the ray starting at $A_e(\mathbf{d})$ (resp. $B_e(\mathbf{d})$) and containing $A'_e(\mathbf{d})$ (resp. $B'_e(\mathbf{d})$) does not intersect the interior of $\mathcal{P}$. We have

$$l_e(\mathbf{d}) = |a_e, A_e(\mathbf{d})| + |b_e, B_e(\mathbf{d})|,$$

where $|\cdot, \cdot|$ denotes the Euclidean distance function.

If $S_e(\mathbf{d})$ consists of only one segment, then only one of $A_e(\mathbf{d})$ and $B_e(\mathbf{d})$ (and similarly, only one of $A'_e(\mathbf{d})$ and $B'_e(\mathbf{d})$) is defined and, hence, $l_e(\mathbf{d})$ contains only one term. If $S_e(\mathbf{d}) = \emptyset$ or $S_e(\mathbf{d}) = e$, then the points $A_e(\mathbf{d})$, $B_e(\mathbf{d})$, $A'_e(\mathbf{d})$ and $B'_e(\mathbf{d})$ are undefined. In the former case, we have $l_e(\mathbf{d}) = 0$, whereas in the latter case, $l_e(\mathbf{d})$ is equal to the length of e .

We fix an edge e , and consider the behavior of the function $l_e(\mathbf{d})$ as the angle $\phi_{\mathbf{d}}$ varies from 0 to 2π . Consider an angle $\phi_{\mathbf{d}}$, and assume that $B_e(\mathbf{d})$ and $B'_e(\mathbf{d})$ exist. We will express the distance $|b_e, B_e(\mathbf{d})|$ in terms of the angle $\phi_{\mathbf{d}}$.

Let c_e be the orthogonal projection of vertex $B'_e(\mathbf{d})$ on the line through edge e . (See Figure 5.) First assume that c_e lies on e , and $B_e(\mathbf{d})$ lies between a_e and c_e . Also, assume that $0 < \phi_{\mathbf{d}} < \alpha_e < \pi/2$.

The angle between the vectors $B'_e(\mathbf{d})B_e(\mathbf{d})$ and $B'_e(\mathbf{d})c_e$ is equal to $\alpha_e - \phi_{\mathbf{d}}$. Therefore,

$$|b_e, B_e(\mathbf{d})| = |b_e, c_e| + |c_e, B_e(\mathbf{d})| = |b_e, c_e| + |B'_e(\mathbf{d}), c_e| \tan(\alpha_e - \phi_{\mathbf{d}}).$$

If the angle $\phi_{\mathbf{d}}$ does not satisfy $0 < \phi_{\mathbf{d}} < \alpha_e < \pi/2$, or if (i) c_e lies on e , and $B_e(\mathbf{d})$ lies between b_e and c_e , or (ii) c_e does not lie on e , then we get a similar expression for $|b_e, B_e(\mathbf{d})|$. Moreover, if $A_e(\mathbf{d})$ and $A'_e(\mathbf{d})$ also exist, then we can write $|a_e, A_e(\mathbf{d})|$ in a similar fashion.

It follows that we can write $l_e(\mathbf{d})$ in the form

$$l_e(\mathbf{d}) = X_e + Y_e \tan(\alpha_e - \phi_{\mathbf{d}}),$$

where X_e and Y_e are (possibly negative) real numbers that only depend on the edge e and the points $A'_e(\mathbf{d})$ and/or $B'_e(\mathbf{d})$. Clearly, if edge e needs support, or e is completely attached to supports, for direction $\mathbf{d}$, we get the same expression; in these cases, X_e is equal to the length of e , and $Y_e = 0$.

If we vary the angle $\phi_{\mathbf{d}}$ by a small amount, then, in general, the values of X_e and Y_e do not change; hence the above expression for $l_e(\mathbf{d})$ remains the same. For some angles $\phi_{\mathbf{d}}$, however, the values of X_e and Y_e will change. Therefore, we want to partition the interval $[0, 2\pi]$ of directions into subintervals, such that within each subinterval I , the function $l_e(\mathbf{d})$ can be written as

$$l_e(\mathbf{d}) = X_e^I + Y_e^I \tan(\alpha_e - \phi_{\mathbf{d}}),$$

where X_e^I and Y_e^I are constant within I .

How do we find this partition? In order to answer this question, we define for each vertex v of $\mathcal{P}$, its *visibility cone*, $\text{cone}(v)$, as the cone with apex v and maximum angular range in which v can see to infinity.

The following lemma identifies certain directions $\mathbf{d}_0$ where the combinatorial structure of the supports associated with an edge changes.

Lemma 6.2 *Let e be an edge of $\mathcal{P}$, $\mathbf{d}_0$ a direction, ϕ the angle corresponding to $\mathbf{d}_0$, and ϵ a positive real number.*

1. *Assume that for directions $\mathbf{d}$ such that $\phi - \epsilon < \phi_{\mathbf{d}} < \phi$, the vertex $A'_e(\mathbf{d})$ is equal to A' , and for directions $\mathbf{d}$ such that $\phi < \phi_{\mathbf{d}} < \phi + \epsilon$ it is equal to A'' . Then, the line segment $A'A''$ is on one of the bounding rays of $\text{cone}(A')$ and is in the direction $\mathbf{d}_0$.*
2. *Assume that for directions $\mathbf{d}$ such that $\phi - \epsilon < \phi_{\mathbf{d}} < \phi$, the vertex $A'_e(\mathbf{d})$ exists, and for directions $\mathbf{d}$ such that $\phi < \phi_{\mathbf{d}} < \phi + \epsilon$ it does not exist, or vice versa. Then*
 - (a) *the line segment $B'_e(\mathbf{d}_0)A'_e(\mathbf{d}_0)$ is on one of the bounding rays of $\text{cone}(B'_e(\mathbf{d}_0))$ and is in the direction $\mathbf{d}_0$, or*
 - (b) *the line segment $b_eA'_e(\mathbf{d}_0)$ is on one of the bounding rays of $\text{cone}(b_e)$ and is in the direction $\mathbf{d}_0$, or*
 - (c) *the line segment $a_eA'_e(\mathbf{d}_0)$ is on one of the bounding rays of $\text{cone}(a_e)$ and is in the direction $\mathbf{d}_0$, or*
 - (d) *the direction $\mathbf{d}_0$ is parallel to e .*

A similar claim holds for the vertex $B'_e(\mathbf{d})$.

Proof To prove 1, first observe that for all directions $\mathbf{d}$ such that $\phi - \epsilon < \phi_{\mathbf{d}} < \phi + \epsilon$, the set $S_e(\mathbf{d})$ is not equal to the entire edge e . If we increase the direction angle $\phi_{\mathbf{d}}$ from $\phi - \epsilon$ to ϕ , then the ray starting at $A_e(\mathbf{d})$ and going into the direction $\mathbf{d}$ rotates around vertex A' . Moreover, the part of this ray that is beyond A' does not intersect the polygon $\mathcal{P}$. At direction $\mathbf{d}_0$, the ray contains both A' and A'' . If we increase $\phi_{\mathbf{d}}$ further from ϕ to $\phi + \epsilon$, the ray rotates around A'' , and the part beyond A'' does not intersect $\mathcal{P}$. It follows that at direction $\mathbf{d}_0$, the vertex A'' must be beyond A' on this ray. Hence, the line segment $A'A''$ is on one of the bounding rays of $\text{cone}(A')$.

The proof of 2 is similar. In Case (a), while increasing $\phi_{\mathbf{d}}$ from $\phi - \epsilon$ to ϕ , the two endpoints $A_e(\mathbf{d})$ and $B_e(\mathbf{d})$ of the two segments defining the set $S_e(\mathbf{d})$ get closer together, and meet if $\phi_{\mathbf{d}} = \phi$. Note that $A'_e(\mathbf{d}_0)$ is beyond $B'_e(\mathbf{d}_0)$ on the ray from $A_e(\mathbf{d}_0)$ in direction $\mathbf{d}_0$. In Case (b), for $\phi_{\mathbf{d}} \in (\phi - \epsilon, \phi)$, the set $S_e(\mathbf{d})$ only consists of the segment $a_eA_e(\mathbf{d})$. While increasing $\phi_{\mathbf{d}}$ to ϕ , the point $A_e(\mathbf{d})$ moves to the vertex b_e . If $\phi_{\mathbf{d}} = \phi$, $A_e(\mathbf{d})$ and b_e are equal. Case (c) is the case where vertex $A'_e(\mathbf{d})$ changes from undefined to defined. Finally, in Case (d), the edge e starts or stops needing support at direction $\mathbf{d}_0$. ■

Let $\mathbf{D}_a$ be the set of directions $\mathbf{d}$ determined by the bounding rays of the non-empty visibility cones. We also define a set $\mathbf{D}_b$ containing the following directions. For each edge e of $\mathcal{P}$, $\mathbf{D}_b$ contains the two directions that are parallel to e . Let $\mathbf{D} := \mathbf{D}_a \cup \mathbf{D}_b$. Then the set $\mathbf{D}$ contains at most $4n$ directions. The following lemma follows from the discussion above.

Lemma 6.3 *The directions of the set $\mathbf{D}$ partition the interval $[0, 2\pi]$ into $O(n)$ subintervals, such that within each subinterval I , the function $L_{\mathcal{P}}(\mathbf{d})$ is the sum of n functions $l_e(\mathbf{d})$, each having the form*

$$l_e(\mathbf{d}) = X_e^I + Y_e^I \tan(\alpha_e - \phi_{\mathbf{d}}), \quad (2)$$

where X_e^I and Y_e^I are constant within I .

We call *critical* the directions $\mathbf{d}$ at which expression (2) changes for some edge e (i.e., X_e^I or Y_e^I changes). Lemma 6.3 gives a set of $O(n)$ directions that includes all critical directions.

Now we are ready to give an outline of our algorithm for computing the direction $\mathbf{d}$ for which $L_{\mathcal{P}}(\mathbf{d}) = \sum_e l_e(\mathbf{d})$ is minimal. After this outline, we will give the details for each step.

Step 1: Compute the set $\mathbf{D}$ defined above, and sort its elements in counterclockwise order. Preprocess $\mathcal{P}$ into a data structure, such that ray shooting queries can be answered.

At this point, we have a partition of $[0, 2\pi]$ into $O(n)$ subintervals. Within each subinterval I , we know that $L_{\mathcal{P}}(\mathbf{d})$ has the form

$$L_{\mathcal{P}}(\mathbf{d}) = \sum_e l_e(\mathbf{d}) = \sum_e (X_e^I + Y_e^I \tan(\alpha_e - \phi_{\mathbf{d}})), \quad (3)$$

where X_e^I and Y_e^I are constant within I .

Step 2: Obtain expression (3) for the function $L_{\mathcal{P}}(\mathbf{d})$ in the first subinterval I , and compute the minimum of $L_{\mathcal{P}}(\mathbf{d})$ within I .

Step 3: Sweep over the elements of $\mathbf{D}$, thereby visiting the subintervals one by one. At each direction $\mathbf{d} \in \mathbf{D}$, obtain the new expression for $L_{\mathcal{P}}(\mathbf{d})$, by subtracting the functions $l_e(\mathbf{d})$ whose expressions change at this direction, and adding the corresponding new functions $l_e(\mathbf{d})$. Then compute the minimum of $L_{\mathcal{P}}(\mathbf{d})$ within the new subinterval.

Let us consider Step 1 first. Using the algorithm of [16, 23], we compute in $O(n)$ time all vertices v for which $\text{cone}(v) \neq \emptyset$, together with their visibility cones. Then, we obtain the elements of $\mathbf{D}$ in sorted order, in $O(n \log n)$ time. We take the ray shooting data structure of [17]. There it is shown how to preprocess $\mathcal{P}$ in $O(n)$ time, such that ray shooting queries can be answered in $O(\log n)$ time. Hence, Step 1 can be completed in $O(n \log n)$ time.

Next, consider Step 2. For edges e that need support for directions in I , i.e., $\mathbf{n}_e \cdot \mathbf{d} < 0$ for all $\mathbf{d} \in I$, expression (2) is trivial.

In order to find expression (2) for edges e such that $\mathbf{n}_e \cdot \mathbf{d} \geq 0$ for all $\mathbf{d} \in I$, we have to determine the set $S_e(\mathbf{d})$. In particular, if $S_e(\mathbf{d}) \neq \emptyset$ and $S_e(\mathbf{d}) \neq e$, we have to find the vertices $A'_e(\mathbf{d})$ and/or $B'_e(\mathbf{d})$. We do the following.

Choose a direction $\mathbf{d}$ in I . For each vertex v of $\mathcal{P}$, perform a ray shooting query from v in direction $\mathbf{d}$. If this ray does not intersect $\mathcal{P}$, then we also perform a ray shooting query from v in direction $-\mathbf{d}$, provided this ray does not immediately go inside $\mathcal{P}$. If e is the edge that is hit first by this second ray and the ray intersects the interior of e , then $\mathbf{n}_e \cdot \mathbf{d} \geq 0$, $S_e(\mathbf{d}) \neq \emptyset$ and $S_e(\mathbf{d}) \neq e$. Moreover, v is equal to $A'_e(\mathbf{d})$ or $B'_e(\mathbf{d})$, and it is easy to decide whether $v = A'_e(\mathbf{d})$ or $v = B'_e(\mathbf{d})$.

After these ray shooting queries, we have found all edges e such that $\mathbf{n}_e \cdot \mathbf{d} \geq 0$, $S_e(\mathbf{d}) \neq \emptyset$ and $S_e(\mathbf{d}) \neq e$, together with the corresponding vertices $A'_e(\mathbf{d})$ and/or $B'_e(\mathbf{d})$. For all remaining edges e for which $\mathbf{n}_e \cdot \mathbf{d} \geq 0$, we know that either $S_e(\mathbf{d}) = \emptyset$ or $S_e(\mathbf{d}) = e$. Therefore, for each such edge e , we take an arbitrary point x on e , and perform a ray shooting query from x in direction $\mathbf{d}$. If this ray intersects $\mathcal{P}$, then $S_e(\mathbf{d}) = e$; otherwise $S_e(\mathbf{d}) = \emptyset$.

It follows that expression (3) for $L_{\mathcal{P}}(\mathbf{d})$ in the first subinterval I can be computed in $O(n \log n)$ time. The problem that remains is that of computing the minimum of $L_{\mathcal{P}}(\mathbf{d})$ in I . We will consider this in Section 6.1.

We are left with Step 3. Assume, we move from subinterval I to I' . Let $\mathbf{d}$ be the critical direction corresponding to the right endpoint of I (which is the left endpoint of I').

First assume that $\mathbf{d} \in \mathbf{D}_b$, i.e., it is parallel to, say, edge e . There are two possible cases. First, if $\mathbf{n}_e \cdot \mathbf{d}' < 0$ for all $\mathbf{d}'$ in the interior of I' , then edge e needs support in the subinterval I' . Therefore, we subtract the (old) term $l_e(\mathbf{d})$ from $L_{\mathcal{P}}(\mathbf{d})$ and add the new value $l_e(\mathbf{d})$, which is equal to the length of e .

Otherwise, $\mathbf{n}_e \cdot \mathbf{d}' > 0$ for all $\mathbf{d}'$ in the interior of I' . In this case, we know that e needs support in the subinterval I , and the (old) term $l_e(\mathbf{d})$ is equal to the length of e . The new expression for $l_e(\mathbf{d})$ is obtained as follows. Perform a ray shooting query from some point on e in direction $\mathbf{d}$. (Note that this ray is “along” e .) If this ray does not intersect the interior of $\mathcal{P}$, then no point on e is attached to a support in the subinterval I' , and we subtract $l_e(\mathbf{d})$ from $L_{\mathcal{P}}(\mathbf{d})$. Otherwise, the ray hits the interior of some edge of $\mathcal{P}$, and in the subinterval I' , edge e is completely attached to supports. Hence, in I' , the term $l_e(\mathbf{d})$ remains equal to the length of e ; we do not have to update $L_{\mathcal{P}}(\mathbf{d})$.

It remains to consider the case when $\mathbf{d}$ is a “cone direction”, i.e., it coincides with the direction of a bounding ray r of, say, $\text{cone}(v)$. In order to describe what has to be done now, assume w.l.o.g. that $\mathbf{d}$ goes vertically upwards. Let $w \neq v$ be the vertex that is on r . Note that both edges e_w and e'_w that are incident to w are on one side of the line containing r . Assume that they are to the left of this line, and that e_w is above e'_w . (The case when e_w and e'_w are to the right of this line can be handled similarly.) Note that by going down (resp. up) from any interior point on e_w (resp. e'_w), we go into the interior of $\mathcal{P}$. (Assume that by going up from any interior point on e_w , we go into the interior of $\mathcal{P}$. Then by walking along the boundary of $\mathcal{P}$, starting at w and following edge e_w first, we must intersect the ray r . This cannot happen, because r does not intersect the interior of $\mathcal{P}$.)

Let e_v and e'_v be the edges that are incident to v . There are three possible cases. First assume that e_v and e'_v are both to the right of the line containing r . Also, assume w.l.o.g. that e_v is above e'_v . (See Figure 6(a).) Then $\text{cone}(v)$ is also to the right of this line, and by going down (resp. up) from any interior point on e_v (resp. e'_v), we go into the interior of $\mathcal{P}$. Perform a ray shooting query from v in direction $-\mathbf{d}$, and let f be the edge that is hit first. Then, in the subinterval I , w is the vertex $A'_f(\mathbf{d})$, whereas $v = B'_f(\mathbf{d})$. In I' , the edge f is completely attached to supports. This is because e_w and e'_w (resp. e_v and e'_v) are to the left (resp. right) of the line containing r . Therefore, we subtract the term $l_f(\mathbf{d})$ from $L_{\mathcal{P}}(\mathbf{d})$ and add the new term $l_f(\mathbf{d})$, which is equal to the length of f . Also, in I , the set $S_{e_v}(\mathbf{d})$ consists of at most one segment: the vertex $B'_{e_v}(\mathbf{d})$ may exist, but $A'_{e_v}(\mathbf{d})$ does not exist. In I' , the vertex $B'_{e_v}(\mathbf{d})$ is still the same or still does not exist, but we have $w = A'_{e_v}(\mathbf{d})$. Therefore, we update the term $l_{e_v}(\mathbf{d})$.

Next, assume that e_v and e'_v are both to the left of the line containing r . (See Figure 6(b).) Again, assume w.l.o.g. that e_v is above e'_v . By going down (resp. up) from any interior point on e_v (resp. e'_v), we go into the interior of $\mathcal{P}$. Perform a ray shooting query from v in direction $-\mathbf{d}$, and let f be the edge that is hit first. (Assume for the moment that f exists.) Then, in I , we have $v = A'_f(\mathbf{d})$, whereas in I' , we have $w = A'_f(\mathbf{d})$. Also, the vertex $B'_f(\mathbf{d})$ is the same in I and I' , or is undefined in both these intervals. Therefore, we update the term $l_f(\mathbf{d})$. In I , we have $w = A'_{e_v}(\mathbf{d})$, whereas $B'_{e_v}(\mathbf{d})$ does not exist. In I' , edge e_v is completely attached to supports. Therefore, we replace the term $l_{e_v}(\mathbf{d})$ by a new term, which is equal to the length of e_v . If edge f does not exist, then we only have to update the term $l_{e_v}(\mathbf{d})$; in the way just described.

Finally, assume that e_v and e'_v are on different sides of the line containing r . (See Figure 6(c).) Assume w.l.o.g. that e_v is to the left of this line. By going down from any interior point on e_v or e'_v , we go into the interior of $\mathcal{P}$. In subinterval I , the vertex w is equal to $A'_{e_v}(\mathbf{d})$, whereas $B'_{e_v}(\mathbf{d})$ is undefined in this interval. In I' , edge e_v is completely attached to supports. So, we replace the term $l_{e_v}(\mathbf{d})$ by a new term, which is equal to the length of e_v . Also, in I , the vertex $A'_{e'_v}(\mathbf{d})$ is undefined, whereas in I' , it is equal to w . In I and I' , the vertex $B'_{e'_v}(\mathbf{d})$ may be defined or undefined. At direction $\mathbf{d}$, however, its “value” does not change. Therefore, we update the term $l_{e'_v}(\mathbf{d})$.

This concludes the description of Step 3. For each critical direction, we need $O(\log n)$ time to

Figure 6: The three possible cases for a “cone direction” $\mathbf{d}$. We assume that $\mathbf{d}$ goes vertically upwards.

update the expression for $L_{\mathcal{P}}(\mathbf{d})$. As in Step 2, the problem is in computing the minimum of $L_{\mathcal{P}}(\mathbf{d})$ in the new subinterval I' . This problem will be addressed in Section 6.1.

Remark 6.1 The polygon $\mathcal{P}$ may have edges that need support, or are completely attached to supports, for any direction $\mathbf{d}$. These edges are invisible, in the sense that no point on them is visible from the “outside”. Consider such an invisible edge e . In Step 2, we find out that in the first subinterval, $l_e(\mathbf{d})$ is equal to the length of e . During the sweep in Step 3, the term $l_e(\mathbf{d})$ never changes, i.e., it always stays equal to the length of e .

6.1 Minimizing the function $L_{\mathcal{P}}(\mathbf{d})$

The problem that remains is to compute the minimum of the function

$$L_{\mathcal{P}}(\mathbf{d}) = \sum_e (X_e^I + Y_e^I \tan(\alpha_e - \phi_{\mathbf{d}})),$$

in a subinterval I of $[0, 2\pi]$. Recall that X_e^I and Y_e^I are real numbers that are constant for $\phi_{\mathbf{d}} \in I$.

We write this optimization problem in a simpler form. Note that the term $\sum_e X_e^I$ is independent of $\mathbf{d}$. Introducing new variables (a_i for Y_e^I , b_i for α_e , and ϕ for $\phi_{\mathbf{d}}$), leads to the problem of minimizing the function

$$F(\phi) := \sum_{i=1}^n a_i \tan(b_i - \phi),$$

in a subinterval I of $[0, 2\pi]$. Here, the a_i 's and b_i 's are real numbers. Using the formula $\tan(y - x) = (\tan y - \tan x)/(1 + \tan y \tan x)$, and defining $c_i := \tan b_i$, we get

$$F(\phi) = \sum_{i=1}^n a_i \frac{c_i - \tan \phi}{1 + c_i \tan \phi} = \sum_{i=1}^n \left(\frac{a_i(c_i + 1/c_i)}{1 + c_i \tan \phi} - \frac{a_i}{c_i} \right).$$

Let $d_i := a_i(c_i + 1/c_i)$. Then minimizing F is equivalent to minimizing the function

$$\sum_{i=1}^n \frac{d_i}{1 + c_i \tan \phi}.$$

Let $x := \tan \phi$. Then we get the following problem.

Problem $PR(n)$: Given $2n + 2$ real numbers $c_1, c_2, \dots, c_n, d_1, d_2, \dots, d_n, A$ and B , compute the global minimum of the function

$$G(x) := \sum_{i=1}^n \frac{d_i}{1 + c_i x}, \quad (4)$$

in the interval $[A, B]$.

We can solve this problem using standard techniques from calculus. Let us consider the derivative of G :

$$G'(x) = \sum_{i=1}^n \frac{-d_i c_i}{(1 + c_i x)^2} = \frac{-1}{\prod_{k=1}^n (1 + c_k x)^2} \sum_{i=1}^n c_i d_i \prod_{j \neq i} (1 + c_j x)^2.$$

Hence, $G'(x) = 0$ if and only if

$$\sum_{i=1}^n c_i d_i \prod_{j \neq i} (1 + c_j x)^2 = 0. \quad (5)$$

The expression in (5) is a polynomial in x of degree $2(n-1)$. Hence, the original function $L_{\mathcal{P}}(\mathbf{d})$ we are interested in has a linear number of local minima. Using techniques from numerical analysis, we compute (i.e., approximate to any desired precision) the roots of (5), and for each of them that is contained in the interval $[A, B]$, we evaluate G . We also evaluate G for $x = A$ and $x = B$. In this way, we find the global minimum of G in $[A, B]$.

Theorem 6.1 Given a simple polygon with n vertices, Problem 6.1 can be solved in $O(n \log n + np(n))$ time, where $p(n)$ is the time for solving problem $PR(n)$.

Proof Consider our algorithm. Step 1 takes $O(n \log n)$ time. In Step 2, it takes $O(n \log n)$ time, to write down expression (3) for the function $L_{\mathcal{P}}(\mathbf{d})$ in the first subinterval I . Given this expression, we can transform it into (4) in linear time. Then in $p(n)$ time, we compute the minimum of this function. Hence, Step 2 takes total time $O(n \log n + p(n))$. In Step 3, we visit the $O(n)$ critical directions one after another. Going from one direction to the next one, we can update the functions $L_{\mathcal{P}}(\mathbf{d})$ and $G(x)$ in $O(\log n)$ time. The minimum of the updated function $L_{\mathcal{P}}(\mathbf{d})$ can then be computed in $p(n)$ time. ■

As we will show now, the running time can be improved considerably, if the edges of our polygon have only a small number of orientations. A polygon is called *C-oriented* if its edges have at most C different orientations. Suppose that our simple polygon $\mathcal{P}$ is *C-oriented*. In this case, the function $G(x)$ in problem $PR(n)$ can be rewritten such that it contains only C terms: There are at most C different normal angles α_e , hence at most C different values for the c_i 's. If we group these together, we get an expression of the form

$$G(x) = \sum_{i=1}^C \frac{d'_i}{1 + c_i x},$$

for real numbers $d'_1, d'_2, \dots, d'_C$. Therefore, solving $G'(x) = 0$ leads to a polynomial of degree at most $2(C-1)$. Assuming that C is a constant, and that roots of polynomials of constant degree

can be computed in constant time, we can compute the minimum of $G(x)$ in constant time. This proves:

Theorem 6.2 *Given a simple C -oriented polygon with n vertices, where C is a constant, Problem 6.1 can be solved in $O(n \log n)$ time.*

7 Minimum-area supports for simple polygons

We consider the problem of computing a direction that minimizes the total *area* of supports for a simple polygon. Let $\mathcal{P}$ be a simple polygon, and let n denote the number of its vertices. If $\mathbf{d}$ is a direction, then $\phi_{\mathbf{d}}$ denotes the angle between the positive x -axis and $\mathbf{d}$. For any edge e of $\mathcal{P}$, we denote by α_e the angle between the positive x -axis and the outer normal $\mathbf{n}_e$ of e . All angles are measured counterclockwise in the range from 0 to 2π .

Our basic approach is the same as in Section 6. We partition the interval $[0, 2\pi]$ into $O(n)$ subintervals, such that within each subinterval, there is a closed form for the total area of the supports. Then we sweep along these subintervals. As we will see later (see Remark 7.2), there can be a quadratic number of changes in the area expression during the entire sweep. These are caused by the parts of the supports that are connected to the platform. Therefore, rather than updating the area expression at the boundary of each subinterval I , we compute it from scratch, and then compute its minimum within I .

7.1 Determining the area expression

Let I be a sufficiently small interval of angles $\phi_{\mathbf{d}}$. (Later, we will see which conditions I has to satisfy.) We show how to compute the area expression for angles $\phi_{\mathbf{d}} \in I$.

Let $\mathbf{d}$ be any direction such that $\phi_{\mathbf{d}} \in I$. Let $t(\mathbf{d})$ be the vertex of $\mathcal{P}$ that is extreme in direction $-\mathbf{d}$. Hence, the platform is the line through $t(\mathbf{d})$ orthogonal to $\mathbf{d}$, and $\mathcal{P}$ is “above” (w.r.t. direction $\mathbf{d}$) this line. Consider the *trapezoidal decomposition* of the supports for this direction, which is defined as follows. (Later, we will show how to compute it efficiently.) For each vertex v of $\mathcal{P}$, shoot a ray from v in direction $\mathbf{d}$, provided this ray (i) does not immediately go inside $\mathcal{P}$ and (ii) it intersects the boundary of $\mathcal{P}$. The ray stops at the first intersection with $\mathcal{P}$ ’s boundary. Similarly, shoot a ray from v in direction $-\mathbf{d}$, provided this ray (i) does not immediately go inside $\mathcal{P}$ and (ii) it intersects the platform or the boundary of $\mathcal{P}$. The ray stops at the first intersection with the platform or $\mathcal{P}$ ’s boundary.

Doing this for all vertices, we get a partition of the supports into disjoint trapezoids. (A triangle is considered as a degenerate trapezoid.) This partition is the trapezoidal decomposition we are interested in. Clearly, the total area of the supports is equal to the sum of the areas of the trapezoids.

In order to get a linear number of critical directions, we subdivide the trapezoids into two classes. A trapezoid that does not have an edge on the platform is colored *red*, otherwise it is colored *blue*. Hence, we can talk about a *red connected component* of the trapezoids. The area expression for the supports can now be determined by considering the red connected components and the blue trapezoids, and distinguishing three cases.

Case 1: Let $R(\mathbf{d})$ be a red connected component that is not connected to any blue trapezoid.

The red component $R(\mathbf{d})$ contains exactly one edge on its boundary that is not an edge of the polygon $\mathcal{P}$. (Refer to Figure 7.) We denote this edge by $b(\mathbf{d})$. If $\phi_{\mathbf{d}} \in I$ increases by a sufficiently

Figure 7: Illustration for Case 1. $R(\mathbf{d})$ is a red connected component and is not connected to any blue trapezoid. The points c_0 and $c(\mathbf{d})$ lie on edge e . Edge $b(\mathbf{d})$ is on the boundary of $R(\mathbf{d})$, and has a and $c(\mathbf{d})$ as its endpoints. If the angle $\phi_{\mathbf{d}}$ increases, then point $c(\mathbf{d})$ moves along e to the right.

small amount, then one endpoint of $b(\mathbf{d})$ remains fixed (this endpoint is a vertex of $\mathcal{P}$), whereas the other one moves along an edge, say e , of $\mathcal{P}$. We denote the fixed endpoint by a , and the moving endpoint on edge e by $c(\mathbf{d})$. We assume that the interval I is chosen small enough such that for each $\phi_{\mathbf{d}} \in I$, (i) $b(\mathbf{d})$ has the vertex a as one of its endpoints, and (ii) edge e is the same.

We denote the area of $R(\mathbf{d})$ by $K_R(\mathbf{d})$. Let b_0 , c_0 , and K_0 be the edge $b(\mathbf{d})$, the point $c(\mathbf{d})$, and the value of $K_R(\mathbf{d})$, respectively, if $\phi_{\mathbf{d}}$ is the left boundary of the interval I .

For each $\phi_{\mathbf{d}} \in I$, the value of $K_R(\mathbf{d})$ is equal to K_0 plus or minus the area of triangle $ac_0c(\mathbf{d})$. We consider the case of a plus sign. This is the case in Figure 7. The case of a minus sign, e.g. the mirror image of Figure 7 (in which case c_0 is different), can be handled in a similar way. Note that a is above $c(\mathbf{d})$ w.r.t. direction $\mathbf{d}$.

In order to find an expression for the area of triangle $ac_0c(\mathbf{d})$, let a' be the orthogonal projection of vertex a on the line through edge e . Assume that $0 < \phi_{\mathbf{d}} < \alpha_e < \pi/2$. (The other cases can be handled similarly.) Note that the x -coordinate of $c(\mathbf{d})$ is between those of c_0 and a' . The angle between the vectors $ac(\mathbf{d})$ and aa' is equal to $\alpha_e - \phi_{\mathbf{d}}$. Therefore, the area of triangle $ac_0c(\mathbf{d})$ is equal to

$$\begin{aligned} \frac{1}{2} |c_0, c(\mathbf{d})| \cdot |a, a'| &= \frac{1}{2} (|c_0, a'| - |a', c(\mathbf{d})|) \cdot |a, a'| \\ &= \frac{1}{2} (|c_0, a'| - |a, a'| \tan(\alpha_e - \phi_{\mathbf{d}})) \cdot |a, a'|, \end{aligned}$$

which can be written as

$$X_e + Y_e \tan(\alpha_e - \phi_{\mathbf{d}}),$$

Figure 8: Illustration for Case 2. Edge e has endpoints a_e and b_e . The platform is formed by the line through t , $a'_e(\mathbf{d})$, and $b'_e(\mathbf{d})$. The polygon rests on vertex t . If the angle $\phi_{\mathbf{d}}$ increases, then the points $a'_e(\mathbf{d})$ and $b'_e(\mathbf{d})$ move along e to the right.

for some real numbers X_e and Y_e that are independent of $\phi_{\mathbf{d}} \in I$, provided the vertex a and the edge e do not change within I .

We have shown that we can write the area of the red component $R(\mathbf{d})$ as

$$K_R(\mathbf{d}) = K_0 + X_e + Y_e \tan(\alpha_e - \phi_{\mathbf{d}}), \quad (6)$$

where K_0 , X_e , and Y_e are (possibly negative) real numbers that are constant within the interval I .

When does expression (6) change? This happens at a direction $\mathbf{d}$ for which (i) vertex a changes, or (ii) edge e changes.

Let us first consider the case when vertex a changes. In this case, there must be a vertex, say v , $v \neq a$, on the ray from $c(\mathbf{d})$ in direction $\mathbf{d}$. If v is on $b(\mathbf{d})$, i.e., between $c(\mathbf{d})$ and a , then the line segment va —which is parallel to $\mathbf{d}$ —is on one of the bounding rays of $\text{cone}(v)$. Otherwise, if v is not on $b(\mathbf{d})$, then it is on the ray from a in direction $\mathbf{d}$. In this case, av is on one of the bounding rays of $\text{cone}(a)$. Hence, for each visibility cone, each of its two bounding rays forms a critical direction.

Next assume that edge e changes. Then the point $c(\mathbf{d})$ must coincide with an endpoint, say v , of e . In this case, the line segment va is on one of the bounding rays of $\text{cone}(v)$, and we get the same critical directions as above.

Case 2: Let $T(\mathbf{d})$ be a blue trapezoid that contains an entire edge of $\mathcal{P}$ on its boundary.

Let e be the edge of $\mathcal{P}$ that is on the boundary of this trapezoid. (Refer to Figure 8.) Note that e is also an edge of $T(\mathbf{d})$ and that the edge of $T(\mathbf{d})$ opposite from e is on the platform. Also, if we increase the angle $\phi_{\mathbf{d}}$ by a sufficiently small amount, the platform and, hence, also this opposite edge, changes.

Let a_e and b_e be the endpoints of e , and let $a'_e(\mathbf{d})$ (resp. $b'_e(\mathbf{d})$) be the orthogonal projection of a_e (resp. b_e) on the platform. Assume w.l.o.g. that a_e is to the left of b_e . Hence, our trapezoid $T(\mathbf{d})$ has a_e , b_e , $b'_e(\mathbf{d})$ and $a'_e(\mathbf{d})$ as its vertices. Let t be the vertex of $\mathcal{P}$ that rests on the platform.

We assume that the interval I is chosen small enough such that for each $\phi_d \in I$, (i) the entire edge e is an edge of $T(d)$, and (ii) t rests on the platform.

We will determine the expression for the area of $T(d)$ within interval I . Assume that $0 < \phi_d < \alpha_e - \pi < \pi/2$. (The other cases can be handled similarly.)

Let β_t (resp. γ_t) be the angle between ta_e (resp. tb_e) and the positive x -axis. The angle between ta_e and $ta'_e(d)$ is equal to $\pi/2 - \beta_t + \phi_d$. Therefore,

$$|a'_e(d), t| = |a_e, t| \cos(\pi/2 - \beta_t + \phi_d) = |a_e, t| \sin(\beta_t - \phi_d),$$

and

$$|a_e, a'_e(d)| = |a_e, t| \sin(\pi/2 - \beta_t + \phi_d) = |a_e, t| \cos(\beta_t - \phi_d).$$

Similarly, since the angle between tb_e and $tb'_e(d)$ is equal to $\pi/2 - \gamma_t + \phi_d$, we have

$$|b'_e(d), t| = |b_e, t| \cos(\pi/2 - \gamma_t + \phi_d) = |b_e, t| \sin(\gamma_t - \phi_d),$$

and

$$|b_e, b'_e(d)| = |b_e, t| \sin(\pi/2 - \gamma_t + \phi_d) = |b_e, t| \cos(\gamma_t - \phi_d).$$

Hence, twice the area of $T(d)$ is equal to

$$\begin{aligned} & (|a_e, a'_e(d)| + |b_e, b'_e(d)|) \cdot |a'_e(d), b'_e(d)| \\ &= (|a_e, t| \cos(\beta_t - \phi_d) + |b_e, t| \cos(\gamma_t - \phi_d)) \\ & \quad \cdot (|a_e, t| \sin(\beta_t - \phi_d) - |b_e, t| \sin(\gamma_t - \phi_d)) \\ &= |a_e, t| \cdot |b_e, t| (\cos(\gamma_t - \phi_d) \sin(\beta_t - \phi_d) - \cos(\beta_t - \phi_d) \sin(\gamma_t - \phi_d)) \\ & \quad + |a_e, t|^2 \cos(\beta_t - \phi_d) \sin(\beta_t - \phi_d) - |b_e, t|^2 \cos(\gamma_t - \phi_d) \sin(\gamma_t - \phi_d) \\ &= |a_e, t| \cdot |b_e, t| \sin(\beta_t - \gamma_t) + \frac{1}{2} |a_e, t|^2 \sin 2(\beta_t - \phi_d) - \frac{1}{2} |b_e, t|^2 \sin 2(\gamma_t - \phi_d). \end{aligned}$$

Writing $\sin 2(\beta_t - \phi_d) = \sin 2\beta_t \cos 2\phi_d - \cos 2\beta_t \sin 2\phi_d$, and similarly for $\sin 2(\gamma_t - \phi_d)$, it follows that the area of $T(d)$ can be written in the form

$$X_T + Z_{T,1} \sin 2\phi_d + Z_{T,2} \cos 2\phi_d,$$

for some real numbers X_T , $Z_{T,1}$ and $Z_{T,2}$ that are independent of $\phi_d \in I$, provided that for each such ϕ_d , the entire edge e is an edge of $T(d)$, and the vertex t rests on the platform.

Using the equations $\sin 2\phi = 2 \tan \phi / (1 + \tan^2 \phi)$ and $\cos 2\phi = 2 / (1 + \tan^2 \phi) - 1$, the latter expression can be written as

$$X_T - Z_{T,2} + \frac{2Z_{T,1} \tan \phi_d + 2Z_{T,2}}{1 + \tan^2 \phi_d}. \quad (7)$$

Expression (7) changes at a direction d for which (i) the vertex resting on the platform changes, or (ii) the edge e ceases being entirely an edge of $T(d)$.

Consider the case when the vertex that rests on the platform changes from, say, t to t' . Then, the line segment tt' is an edge of the convex hull of $\mathcal{P}$, and the direction d is the inner normal of this convex hull edge. That is, for each convex hull edge, its inner normal forms a critical direction.

The second case is when edge e stops being entirely part of $T(d)$. There are two possibilities for this to happen. First, this happens when e is parallel to direction d . That is, for each edge of $\mathcal{P}$, the two directions that are parallel to it are critical directions. Second, one of the line segments

Figure 9: Illustration for Case 3. Edge e has endpoints a_e and b_e . The platform is formed by the line through t , $A'_e(\mathbf{d})$, and $B''_e(\mathbf{d})$. The polygon rests on vertex t . The blue trapezoid $T(\mathbf{d})$ has vertices $A_e(\mathbf{d})$, $A'_e(\mathbf{d})$, $B'_e(\mathbf{d})$, and $B_e(\mathbf{d})$. The red component $R_1(\mathbf{d})$ (resp. $R_2(\mathbf{d})$) has $A_e(\mathbf{d})A'_e(\mathbf{d})$ (resp. $B_e(\mathbf{d})B'_e(\mathbf{d})$) at its right (resp. left) boundary. The segment $A'_e(\mathbf{d})B'_e(\mathbf{d})$ splits $T(\mathbf{d})$ into $T_1(\mathbf{d})$ and $T_2(\mathbf{d})$.

$a_e a'_e(\mathbf{d})$ and $b_e b'_e(\mathbf{d})$ contains a vertex v , $v \neq a_e$ and $v \neq b_e$; v may be equal to t . (For example, assume in Figure 8 that v is on $b_e b'_e(\mathbf{d})$. Then, by increasing $\phi_{\mathbf{d}}$, the trapezoid $T(\mathbf{d})$ has its right edge on the line through v parallel to $\mathbf{d}$. The top edge of this trapezoid does not contain the vertex b_e . Hence, e is not entirely part of it any more.) In this case, the line segment $a_e v$ or $b_e v$ is on one of the bounding rays of $\text{cone}(a_e)$ or $\text{cone}(b_e)$. Hence, for each visibility cone, the “negative” orientation of each of its two bounding rays forms a critical direction.

Case 3: We are left with red connected components that are connected to some blue trapezoid, and with blue trapezoids that contain only part of an edge of $\mathcal{P}$ on their boundaries. (See Remark 7.1 below, in which we show that this really is the remaining case.)

Let $T(\mathbf{d})$ be a blue trapezoid that contains part of an edge, say e , of $\mathcal{P}$ on its boundary. Let a_e and b_e be the endpoints of e , and assume w.l.o.g. that a_e is to the left of b_e . (Refer to Figure 9.) Let $A_e(\mathbf{d})$ and $B_e(\mathbf{d})$ be the vertices of $T(\mathbf{d})$ that are on e , where $A_e(\mathbf{d})$ is to the left of $B_e(\mathbf{d})$. Then at least one of $A_e(\mathbf{d})$ and $B_e(\mathbf{d})$ is in the interior of e . We assume that both are interior points. (The other case can be handled in a similar way.)

Let $A''_e(\mathbf{d})$ and $B''_e(\mathbf{d})$ be the vertices of $T(\mathbf{d})$ that are on the platform, where $A''_e(\mathbf{d})$ is to the left of $B''_e(\mathbf{d})$. Then the line segment $A_e(\mathbf{d})A''_e(\mathbf{d})$ contains a vertex of $\mathcal{P}$, which we denote by $A'_e(\mathbf{d})$. Similarly, we denote the vertex of $\mathcal{P}$ on the segment $B_e(\mathbf{d})B''_e(\mathbf{d})$ by $B'_e(\mathbf{d})$.

Trapezoid $T(\mathbf{d})$ is connected to exactly two red components, which we denote by $R_1(\mathbf{d})$ and $R_2(\mathbf{d})$. Assume w.l.o.g. that $R_1(\mathbf{d})$ is to the left of $R_2(\mathbf{d})$. Then $A_e(\mathbf{d})A'_e(\mathbf{d})$ is the unique boundary edge of $R_1(\mathbf{d})$ that is not an edge of $\mathcal{P}$. Similarly, $B_e(\mathbf{d})B'_e(\mathbf{d})$ is the unique boundary edge of $R_2(\mathbf{d})$

that is not an edge of $\mathcal{P}$. In particular, each of these red components is connected to only one blue trapezoid; namely $T(\mathbf{d})$.

Let $K_T(\mathbf{d})$ denote the total area of $T(\mathbf{d})$, $R_1(\mathbf{d})$, and $R_2(\mathbf{d})$. We will determine the expression for $K_T(\mathbf{d})$ in interval I . We assume that this interval is chosen small enough such that for each $\phi_{\mathbf{d}} \in I$, (i) the segment $A_e(\mathbf{d})B_e(\mathbf{d})$ is non-empty and strictly contained in edge e , (ii) the vertices $A'_e(\mathbf{d})$ and $B'_e(\mathbf{d})$ are the same, and (iii) the same vertex, say t , rests on the platform.

We begin by splitting $T(\mathbf{d})$ into two trapezoids $T_1(\mathbf{d}) = A'_e(\mathbf{d})A''_e(\mathbf{d})B''_e(\mathbf{d})B'_e(\mathbf{d})$ and $T_2(\mathbf{d}) = A_e(\mathbf{d})A'_e(\mathbf{d})B'_e(\mathbf{d})B_e(\mathbf{d})$. Let K_0 be the total area of $T_2(\mathbf{d})$, $R_1(\mathbf{d})$, and $R_2(\mathbf{d})$, if $\phi_{\mathbf{d}}$ is the left boundary of the interval I . Then for each $\phi_{\mathbf{d}} \in I$, the value of $K_T(\mathbf{d})$ is equal to K_0 plus the area of $T_1(\mathbf{d})$. The area of $T_1(\mathbf{d})$ can be written as in Case 2, whereas the value of K_0 does not change as $\phi_{\mathbf{d}}$ is varied within the interval I .

We have shown that the total area of the blue trapezoid and its two neighboring red components can be written in the form

$$Z_{T,0} + \frac{Z_{T,1} \tan \phi_{\mathbf{d}} + Z_{T,2}}{1 + \tan^2 \phi_{\mathbf{d}}}, \quad (8)$$

for some real numbers $Z_{T,0}$, $Z_{T,1}$ and $Z_{T,2}$ that are independent of $\phi_{\mathbf{d}} \in I$, provided that the conditions (i), (ii), and (iii) above hold for interval I .

Expression (8) changes at a direction $\mathbf{d}$ for which one of (i), (ii), and (iii) does not hold any more. It is easy to check that this gives critical directions of the same type as in Cases 1 and 2.

Remark 7.1 Let $\mathbf{d}$ be a direction, and consider a red connected component $R(\mathbf{d})$ that is connected to a blue trapezoid $T(\mathbf{d})$ having an entire edge, say e , of $\mathcal{P}$ on its boundary. Let a be an endpoint of e , and let a' be the orthogonal projection of a on the platform. Assume that a has been chosen such that the line segment aa' contains an edge, say vw , of $R(\mathbf{d})$. Hence, vw is the unique edge of $R(\mathbf{d})$ that is not an edge of $\mathcal{P}$. Assume w.l.o.g. that a , v , w , and a' appear in this order on aa' . Then, vw , which has direction $-\mathbf{d}$ is on one of the bounding rays of $\text{cone}(v)$. That is, $\mathbf{d}$ is a critical direction. If we vary the angle $\phi_{\mathbf{d}}$ by a small amount, then $R(\mathbf{d})$ or $T(\mathbf{d})$ falls into one of the Cases 1, 2, and 3. Therefore, this configuration of $R(\mathbf{d})$ and $T(\mathbf{d})$ has been treated in one of these three cases.

Putting everything together, we have shown that within a sufficiently small interval I of directions, we can write the total area of the supports as

$$X_0^I + \frac{Z_1^I \tan \phi_{\mathbf{d}} + Z_2^I}{1 + \tan^2 \phi_{\mathbf{d}}} + \sum_e Y_e^I \tan(\alpha_e - \phi_{\mathbf{d}}), \quad (9)$$

where X_0^I , Y_e^I , Z_1^I and Z_2^I are real numbers that are constant for $\phi_{\mathbf{d}} \in I$.

How much time do we need to compute expression (9) for a given interval I ? We take an arbitrary direction $\mathbf{d}$ such that $\phi_{\mathbf{d}} \in I$, and compute, in $O(n)$ time, the trapezoidal decomposition of the supports for this direction, as follows.

First, compute the smallest rectangle B that contains the polygon $\mathcal{P}$ and that has sides parallel and orthogonal to $\mathbf{d}$. Note that for build direction $\mathbf{d}$, the lower edge of B is on the platform. Also, each side of B contains at least one vertex of $\mathcal{P}$. By walking along the boundary of $\mathcal{P}$, we can easily decompose the region outside $\mathcal{P}$ and inside B into pairwise disjoint simple polygons. Using the algorithm of Chazelle [7], we can compute for each such polygon $\mathcal{P}'$ its trapezoidal decomposition, in time proportional to the number of vertices of $\mathcal{P}'$. Hence, overall, we spend $O(n)$ time to compute the trapezoidal decomposition of all these polygons. If we now remove all trapezoids that have an

edge on the upper side of B , then what remains is the trapezoidal decomposition of the supports for direction $\mathbf{d}$.

Given this decomposition, we can in $O(n)$ time color the trapezoids red and blue. Next we compute, in $O(n)$ time, the dual graph of the red trapezoids. This graph has a vertex for each red trapezoid, and any two of them are connected if their boundaries have a non-empty intersection. Computing the connected components of this graph can also be done in $O(n)$ time. Given these red components and the blue trapezoids, we can determine expression (9) in $O(n)$ time.

This proves that given an interval I , we can in $O(n)$ time compute the expression for the area of the supports within I .

Let $\mathbf{D}_a$ be the set of directions $\mathbf{d}$ determined by the bounding rays of the non-empty visibility cones, and let $-\mathbf{D}_a := \{-\mathbf{d} : \mathbf{d} \in \mathbf{D}_a\}$. Also, let the set $\mathbf{D}_b$ contain the following directions. For each edge e of $\mathcal{P}$, $\mathbf{D}_b$ contains the two directions that are parallel to e . Finally, let $\mathbf{D}_c$ be the set containing the inner normals of the convex hull edges of $\mathcal{P}$. Let $\mathbf{D} := \mathbf{D}_a \cup -\mathbf{D}_a \cup \mathbf{D}_b \cup \mathbf{D}_c$. Then the set $\mathbf{D}$ contains all critical directions, and it has at most $7n$ elements. Moreover, in $O(n \log n)$ time, we can determine and sort the elements of $\mathbf{D}$.

We summarize the results of this subsection.

Lemma 7.1 *In $O(n \log n)$ time, we can compute a set $\mathbf{D}$ of critical directions. This set $\mathbf{D}$ partitions the interval $[0, 2\pi]$ into $O(n)$ subintervals, such that within each subinterval I , the total area of the supports can be written in the form (9). Given such an interval I , we can determine this expression in $O(n)$ time.*

7.2 The algorithm for minimizing the area of the supports

We are ready now to give our algorithm.

Step 1: Compute and sort the elements of the set $\mathbf{D}$ as specified in Lemma 7.1.

Step 2: Sweep over the elements of $\mathbf{D}$, thereby visiting the corresponding subintervals one after another. For each subinterval I , compute expression (9) for the area of the supports, and compute its minimum within I . The global minimum of the area over all these subintervals is the desired result.

Consider the minimization problem within one subinterval I . As in Subsection 6.1, we can reduce this to the following problem.

Problem $PR'(n)$: *Given $2n + 4$ real numbers $c_1, c_2, \dots, c_n, d_0, d'_0, d_1, d_2, \dots, d_n, A$ and B , compute the global minimum of the function*

$$H(x) := \frac{d_0 x + d'_0}{1 + x^2} + \sum_{i=1}^n \frac{d_i}{1 + c_i x},$$

in the interval $[A, B]$.

If we solve this problem using standard calculus techniques, then we have to compute the roots of the derivative of H , which leads to a polynomial of degree $2n + 2$.

Theorem 7.1 *Let $\mathcal{P}$ be a simple polygon with n vertices, and let $q(n)$ be the time needed for solving problem $PR'(n)$. In $O(n^2 + n q(n))$ time, we can compute a direction $\mathbf{d}$ for which the total area of the supports is minimal.*

If the polygon $\mathcal{P}$ is C -oriented for some constant C , then this optimal direction can be computed in $O(n^2)$ time.

Proof The lemma follows from our algorithm and Lemma 7.1. Step 1 takes $O(n \log n)$ time. In Step 2, we visit $O(n)$ subintervals. For each subinterval, we need $O(n)$ time to determine expression (9), and $q(n)$ time to find its minimum. If $\mathcal{P}$ is C -oriented, then the summation in the function $H(x)$ contains at most C terms. Therefore, if we assume that the roots of a polynomial of constant degree can be computed in constant time, then we can compute the minimum of $H(x)$ for one subinterval in constant time. ■

Remark 7.2 We indicated in the beginning of Section 7, why we compute at each critical direction the expression for the area from scratch. It is easy to construct a polygon having the following property. There are a linear number of vertices, each of which rests on the platform for a certain interval of directions. Moreover, if such a vertex rests on the platform, then there are a linear number of blue trapezoids in the trapezoidal decomposition of the supports. Each time such a vertex becomes the “platform vertex”, we have to update a linear number of terms in the area function; one term for each blue trapezoid. That is, the total number of changes in the area function can be quadratic in the worst case.

8 Conclusion

We have given efficient geometric algorithms for certain optimization problems arising in layered manufacturing. These include (a) a simple and practical algorithm for minimizing the degree of stair-stepping for any polyhedron, (b) algorithms for minimizing the volume and the contact-area of the support structures for a convex polyhedron, and (c) algorithms for minimizing the area and contact-length of supports for a non-convex, simple polygon.

An interesting open problem that we are pursuing is the design of efficient algorithms for optimizing supports for a non-convex polyhedron. We believe that our results for non-convex polygons will be very useful in this effort. Another open problem is to simultaneously optimize two or more criteria; for instance, among all build directions that minimize stair-stepping, find the one which realizes the minimum volume for the supports.

References

- [1] S. Allen and D. Dutta. Determination and evaluation of support structures in layered manufacturing. *Journal of Design and Manufacturing*, 5:153–162, 1995.
- [2] E.M. Arkin, Y.-J. Chiang, M. Held, J.S.B. Mitchell, V. Sacristan, S.S. Skiena, and T.-C. Yang. On minimum-area hulls. In *Proceedings of the 1996 European Symposium on Algorithms, LNCS 1136, Springer-Verlag, Berlin*, pages 334–348, 1996.
- [3] B. Asberg, G. Blanco, P. Bose, J. Garcia-Lopez, M. Overmars, G. Toussaint, G. Wilfong, and B. Zhu. Feasibility of design in stereolithography. In *Proceedings of the 13th FST&TCS Conference*, 1993.
- [4] M. Bablani and A. Bagchi. Quantification of errors in rapid prototyping processes and determination of preferred orientation of parts. In *Transactions of the 23rd North American Manufacturing Research Conference*, 1995.
- [5] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. The Quickhull algorithm for convex hull. Technical Report GCG53, Geometry Center, Univ. of Minnesota, July 1993. (See also <http://www.geom.umn.edu/software/qhull/>).

- [6] P. Bose, M. van Kreveld, and G. Toussaint. Filling polyhedral molds. In *Proc. 3rd Workshop Algorithms Data Structures*, volume 709 of *Lecture Notes in Computer Science*, pages 210–221. Springer-Verlag, 1993.
- [7] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete & Computational Geometry*, 6:485–524, 1991.
- [8] B. Chazelle et al. Application challenges to computational geometry. Technical Report 521, Princeton University, April 1996. (Also available at <http://www.cs.duke.edu/~jeffe/compgeom/taskforce.html>).
- [9] B. Chazelle, L.J. Guibas, and D.T. Lee. The power of geometric duality. *BIT*, 25:76–90, 1985.
- [10] A. Dolenc and I. Mäkelä. Slicing procedures for layered manufacturing techniques. *Computer-Aided Design*, 26:119–126, 1994.
- [11] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer-Verlag, New York, 1987.
- [12] H. Edelsbrunner and L. Guibas. Topologically sweeping an arrangement. *Journal of Computer and System Sciences*, 38:165–194, 1989.
- [13] D. Frank and G. Fadel. Preferred direction of build for rapid prototyping processes. In *Proceedings of the 5th International Conference on Rapid Prototyping*, pages 191–200, 1994.
- [14] A. Gajentaan and M. H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational Geometry: Theory & Applications*, 5:165–185, 1995.
- [15] R.N. Goldman. Area of planar polygons and volume of polyhedra. In J. Arvo, editor, *Graphics Gems II*, pages 170–171. Academic Press, 1991.
- [16] L. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2:209–233, 1987.
- [17] J. Hershberger and S. Suri. A pedestrian approach to ray shooting: shoot a ray, take a walk. *J. Algorithms*, 18:403–431, 1995.
- [18] P. Jacobs. *Rapid Prototyping & Manufacturing: Fundamentals of StereoLithography*. McGraw-Hill, 1992.
- [19] D.G. Luenberger. *Introduction to linear and non-linear programming*. Addison-Wesley, 1973.
- [20] J. Matoušek. Range searching with efficient hierarchical cuttings. *Discrete & Computational Geometry*, 10:157–182, 1992.
- [21] M. McKenna and R. Seidel. Finding the optimal shadows of a convex polytope. In *Proceedings of the 1st Annual ACM Symposium on Computational Geometry*, pages 24–28, 1985.
- [22] F.P. Preparata and M.I. Shamos. *Computational Geometry—An Introduction*. Springer-Verlag, 1988.
- [23] G.T. Toussaint. A linear-time algorithm for solving the strong hidden-line problem in a simple polygon. *Pattern Recognition Letters*, 4:449–451, 1986.