Surface Approximation and Geometric Partitions *

Pankaj K. Agarwal
Department of Computer Science
Box 1029, Duke University
Durahm, NC 27708-0129.

Subhash Suri
Bell Communications Research
445 South Street
Morristown, NJ 07960.

February 19, 1996

Abstract

Motivated by applications in computer graphics, visualization, and scientific computation, we study the computational complexity of the following problem: Given a set S of n points sampled from a bivariate function $f(x, y)$ and an input parameter $\varepsilon > 0$, compute a piecewise-linear function $\Sigma(x, y)$ of minimum complexity (that is, a xy-monotone polyhedral surface, with a minimum number of vertices, edges, or faces) such that

$$|\Sigma(x_p, y_p) - z_p| \leq \varepsilon, \quad \text{for all } (x_p, y_p, z_p) \in S.$$

We prove that the decision version of this problem is *NP-Hard*. The main result of our paper is a polynomial-time approximation algorithm that computes a piecewise-linear surface of size $O(K_o \log K_o)$, where K_o is the complexity of an optimal surface satisfying the constraints of the problem.

The technique developed in our paper is more general and applies to several other problems that deal with partitioning of points (or other objects) subject to certain geometric constraints. For instance, we get the same approximation bound for the following problem arising in machine learning: given n 'red' and m 'blue' points in the plane, find a minimum number of *pairwise disjoint* triangles such that each blue point is covered by some triangle and no red point lies in any of the triangles.

1 Introduction

In scientific computation, visualization, and computer graphics, the modeling and construction of surfaces is an important area. A small sample of some recent papers [1, 2, 4, 8, 11, 14, 22, 23] on this topic gives an indication of the scope and importance of this problem.

*The first author has been supported by National Science Foundation Grant CCR–93–01259 and an NYI award.

Rather than delve into any specific problem studied in these papers, we focus on a general, abstract problem that seems to underlie them all.

In many scientific and computer graphics applications, computation takes place over a surface in three dimensions. The surface is generally modeled by piecewise linear (or, sometimes piecewise cubic) patches, whose vertices lie either on or in the close vicinity of the actual surface. In order to ensure that all local features of the surface are captured, algorithms for an automatic generation of these models generally sample at a dense set of regularly spaced points. Demands for real-time speed and reasonable performance, however, require the models to have as small a combinatorial complexity as possible. A common technique employed to reduce the complexity of the model is to somehow "thin" the surface by deleting vertices with relatively "flat" neighborhoods. Only *ad hoc* and heuristic methods are known for this key step. Most of the thinning methods follow a set of local rules (such as deleting edges or vertices whose incident faces are almost coplanar), which are applied in an arbitrary order until they are no longer applicable. Not surprisingly, these methods come with no performance guarantee, and generally no quantitative statement can be made about the surface approximation computed by them.

In this paper, we address the complexity issues of the surface approximation problem for surfaces that are xy-monotone. These surfaces represent the graphs of continuous bivariate functions $f(x, y)$, and they arise quite naturally in many scientific applications. One possible approach for handling arbitrary surfaces is to break them into monotone pieces, and apply our algorithm individually on each piece. Let us formally define the main problem studied in our paper.

Let f be a continuous function of two variables x and y, and let S be a set of n points sampled from f. A continuous piecewise-linear function Σ is called an *ε-approximation* of f, for $\varepsilon > 0$, if

$$|\Sigma(x_p, y_p) - z_p| \leq \varepsilon,$$

for every point $p = (x_p, y_p, z_p) \in S$. Given S and ε, the *surface-approximation* problem is to compute a piecewise-linear function Σ that ε-approximates f with a minimum number of breakpoints. The breakpoints of Σ can be arbitrary points of $\mathbb{R}^3$, and they are not necessarily points of S. In many applications, f is generally a function hypothesized to fit the observed data—the function Σ is a computationally efficient substitute for f. The parameter ε is used to achieve a complexity-quality tradeoff — smaller the ε, higher the fidelity of the approximation. (The graph of a piecewise-linear function of two variables is also called a *polyhedral terrain* in computational geometry literature; the breakpoints of the function are the vertices of the terrain.)

Although there is a vast literature on the surface-approximation problem in graphics community, the state of theoretical knowledge on this problem appears to be rather slim. The provable performance bounds are known only for convex surfaces. For this special case, an $O(n^3)$ time algorithm is presented by Mitchell and Suri [17] for computing an

ε-approximation of a convex polytope of n vertices in $\mathbb{R}^3$. Their algorithm produces an approximation of size $O(K_o \log n)$, where K_o is the size of an optimal ε-approximation. Extending their work, Clarkson [7] has proposed an $O(K_o n^{1+\delta})$ expected time randomized algorithm for computing an approximation of size $O(K_o \log K_o)$, where δ can be an arbitrarily small positive number. Recently Brönnimann and Goodrich [5] have refined Clarkson's algorithm and have given a polynomial-time algorithm for computing an approximation of size $O(K_0)$.

In this paper, we study the approximation problem for surfaces that correspond to graphs of bivariate functions. We show that it is *NP-Hard* to decide if a surface can be ε-approximated using k vertices (or facets). The main result of our paper, however, is a polynomial-time algorithm for computing an ε-approximating surface of a guaranteed quality. If an optimal ε-approximating surface of f has K_o vertices, then our algorithm produces a surface with $O(K_o \log K_o)$ vertices. Observe that we are dealing with two approximation measures here: ε, which measures the absolute z difference between f and the "simplified" surface Σ, and the ratio between the sizes of the optimal surface and the output of our algorithm. For the lack of better terminology, we use the term "approximation" for both measures. Notice though that ε is an input (user-specified) parameter, $\log K_o$ is the approximation guarantee provided by the analysis of our algorithm.

The key to our approximation method is an algorithm for partitioning a set of points in the plane by *pairwise disjoint* triangles. This is an instance of the geometric set cover problem, with an additional *disjointness* constraint on the covering objects (triangles). Observe that the disjointness condition on covering objects precludes the well-known *greedy* method for set covering [12, 15]; in fact, we can show that a greedy solution has size $\Omega({K_o}^2)$ in the worst-case. Let us now reformulate our surface-approximation problem as a constrained geometric partitioning problem.

Let $\bar{p}$ denote the orthogonal projection of a point $p \in \mathbb{R}^3$ onto the xy-plane $z = 0$. In general, for any set $A \subset \mathbb{R}^3$, we use $\bar{A}$ to denote the orthogonal projection of A onto the xy-plane. Then, in order to get an ε-approximation of f, it suffices to find a set of triangles in 3-space such that (i) the projections of these triangles on plane $z = 0$ are pairwise disjoint and they cover the projected set of points $\bar{S}$, and (ii) the vertical distance between a triangle and any point of S directly above/below it is at most ε. Our polynomial-time algorithm produces a family of $O(K_o \log K_o)$ such triangles. We stitch together these triangles to produce the desired surface Σ. The "stitching" process introduces at most a constant factor more triangles.

The geometric partition framework also includes several extensions and generalizations of the basic surface-approximation problem. For instance, we can formulate a stronger version of the problem by replacing each sample point by a horizontal triangle (or, any polygon). Specifically, we are given a family of *horizontal* triangles (or polygons) in 3-space, whose projections on the xy-plane are pairwise disjoint. We want a piecewise-linear,

ε-approximating surface whose maximum vertical distance from *any* point on the triangles is ε. Our approximation algorithm works equally well for this variant of the problem—this variant addresses the case when some local features of the surface are known in detail; unfortunately, our method works only for horizontal features.

Finally, let us mention the *planar bichromatic partition* problem, which is of independent interest in the machine learning literature: Given a set R of 'red' points and another set B of 'blue' points in the plane, find a minimum number of pairwise disjoint triangles such that each blue point lies in a triangle and no red point lies in any of the triangles. Our algorithm gives a solution with $O(K_o \log K_o)$ triangles. It can also be used to construct a linear decision tree of size $O(K_o \log K_o)$, which is consistent with respect to R and B, where K_o is the size of an optimal linear decision tree.

The running time of our algorithms, though polynomial, is quite high, and at the moment has only theoretical value. These being some of the first results in this area, however, we expect that the theoretical time complexity of these problems would improve with further work. Perhaps, some of the ideas in our paper may also shed light on the theoretical performance of some of the "practical" algorithms that are used in the trade.

2 A Proof of NP-Hardness

We show that both the planar bichromatic partition problem and the surface-approximation problem are *NP-Hard*, by a reduction from the planar 3-SAT. We do not know whether they are in *NP* since the coordinates of the triangles in the solution may require very high precision. We recall that the 3-SAT problem consists of n variables $x_1, \ldots, x_n$, and m clauses $C_1, \ldots, C_m$, each with three literals C_{i1}, C_{i2}, C_{i3}, where C_{ij} is either x_k or $\bar{x}_k$. The problem is to decide whether the boolean formula

$$F = \bigwedge_{i=1}^{m} (C_{i1} \vee C_{i2} \vee C_{i3})$$

has a truth assignment. An instance of 3-SAT is called *planar* if its variable-clause graph is planar. In other words, F is an instance of the planar 3-SAT if the graph $G(F) = (V, E)$ is planar (see [13]), where V and E are defined as follows:

$$\begin{aligned} V &= \{x_1, x_2, \ldots, x_n\}, \\ E &= \{(x_j, C_i) \mid x_i \text{ or } \bar{x}_i \text{ appears in } C_i\}. \end{aligned}$$

Theorem 2.1 *The planar bichromatic-partition problem is* NP-Hard.

Proof: Our construction is similar to the one used by Fowler et al. [10], who prove the intractability of certain planar geometric covering problems (without the disjointness condition); see also [3, 9] for similar constructions. We first describe our construction for the

bichromatic partition problem. To simplify the proof, our construction allows three or more points to lie on a line—the construction can be modified easily to remove these degeneracies.

Let F be a boolean formula, and let $G = (V, E)$ be a straight-line planar embedding of the graph $G(F)$. We construct an instance of the red-blue point partition problem whose solution determines whether F is satisfiable.

We start by placing a 'blue' point at each clause node C_j, $1 \leq j \leq m$. Let x_i be a variable node, and let $e_{i1}, e_{i2}, \ldots, e_{il}$ be the edges incident to it. In the plane embedding of G, the edges e_{ij} form a "star" (see Figure 1 (i)). We replace this star by its Minkowski sum with a disk of radius δ, for a sufficiently small $\delta > 0$. Before performing the Minkowski sum, we shrink all the edges of the star at x_i by 2δ, so that the "star-shaped polygons" meeting at a clause node do not overlap (see Figure 1 (ii)). Let P_i denote the star-shaped polygon corresponding to x_i. In the polygon P_i, corresponding to each edge e_{ij}, there is a tube, consisting of two copies of e_{ij}, each translated by distance δ, plus a circular arc s_{ij} near the clause node C_j.

We place an even number of (say $2k_i$) 'blue' points on the boundary of P_i, as follows. We put two points a_{ij} and b_{ij} on the circular arc s_{ij} near its tip. If C_j contains the literal x_i, we put six points on the straight-line portion of P_i's boundary, three each on translated copies of the edge e_{ij}. On each copy, we move the middle point slightly inwards so as to replace the original edge of P_i by a path of length two. On the other hand, if C_j contains the literal $\bar{x}_i$, we put four points on straight-line portion of P_i's boundary, two each on translated copies of the edge e_{ij}. Thus, the number of blue points added for edge e_{ij} is either six of eight. ($2k_i$ is the total number of points put along P_i.) Let B denote the set of all blue points placed in this way, and let $k = \sum_{i=1}^{n} k_i$.

Finally, we scatter a large (but polynomially bounded) number of 'red' points so that (*i*) any segment connecting two blue points that are not adjacent along the boundary of some P_i contains a red point, and (*ii*) any triangle with three blue points as its vertices contains at least one red point unless the triangle is defined by $a_{ij}b_{ij}C_j$ for some edge $e_{ij} \in E$. (See Figure 1 (iii).) Let R be a set of red points satisfying the above two properties.

We claim that the set of blue points B can be covered by k disjoint triangles, none of which contains any red point, if and only if the formula F has a truth assignment. Our proof is similar to the one in Fowler et al. [10]; we only sketch the main ideas. The red points act as enforcers, ensuring that only those blue points that are adjacent on the boundary of a P_i can be covered by a single triangle. Thus, the minimum number of triangles needed to cover all the points on P_i is k_i. Further, there are precisely two ways to cover these points using k_i triangles— in one covering, a_{ij} and b_{ij} are covered by a single triangle for those clauses only in which $x_i \in C_j$, and, in the other covering, a_{ij} and b_{ij} are covered by a single triangle for those clauses only in which $\bar{x}_i \in C_j$; see Figure 4 (iv). We regard the first covering as setting $x_i = 1$, and the second covering as setting $x_i = 0$.

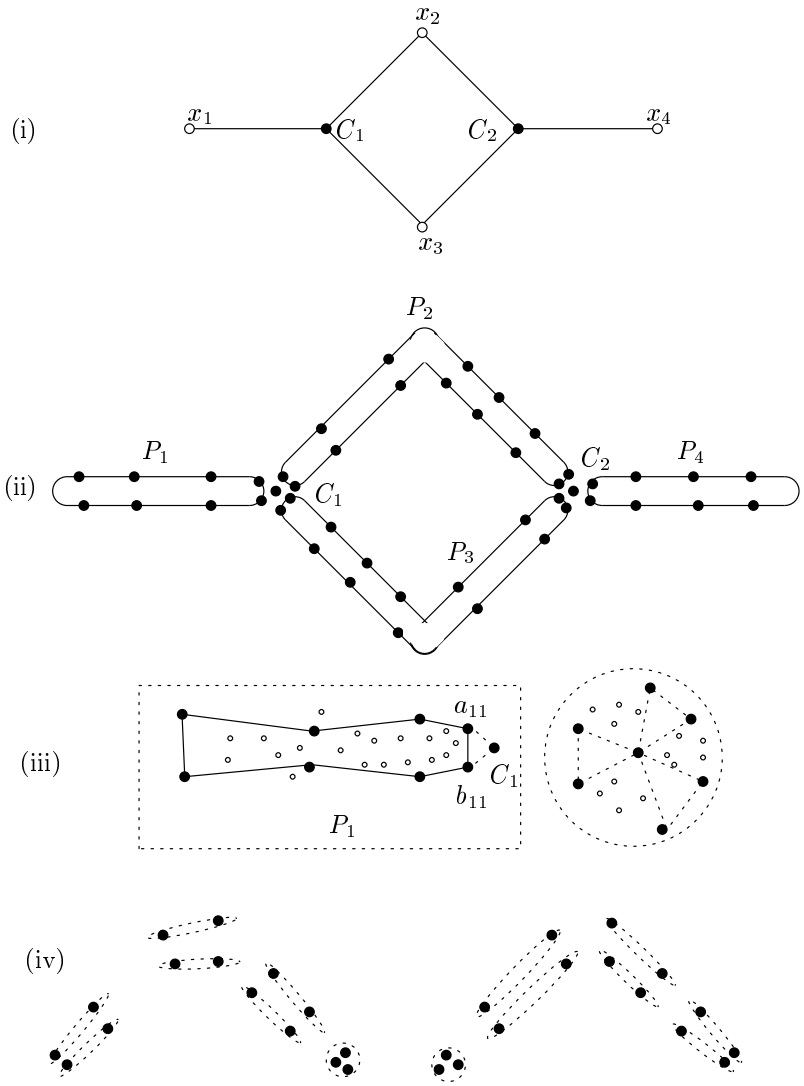

Figure 1: (i) An instance of planar 3-SAT: $F = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_2 \vee \bar{x}_3 \wedge x_4)$; (ii) Corresponding instance of bichromatic partition, (iii) Details of P_1 and C_1, only some of the red points lying near P_1 and C_1 are shown, (iv) Two possible coverings of blue points on P_2.

Suppose $x_i = 1$. For any clause C_j that contains x_i, the points a_{ij} and b_{ij} are covered by a single triangle, and we can cover the clause point corresponding to C_j by the same triangle. The same holds if $x_i = 0$ and the clause C_j contains $\bar{x}_i$. In other words, the clause point of C_j can be covered for free if C_j is satisfied. Thus, the set of blue points B can be covered by k triangles if and only if the clause point for each clause C_j is covered for free, that is, the formula F has a truth assignment. This completes our proof of *NP*-Hardness of the planar bichromatic partition problem. □

Remark: The preceding construction is degenerate in that most of the red points lie on segments connecting two blue points. There are several ways to remove these collinearities; we briefly describe one of them. For each polygon P_i, replace every blue point b on P_i by two blue points b', b'', placed very close to b. (We do not make copies of 'clause points' C_j, $1 \leq j \leq m$.) For every pair of blue points b_j, b_l that we did not want to cover by a single triangle in the original construction, we place a red point in the convex hull of b'_j, b''_j, b'_l, b''_l. If there are $4k_i$ blue points on the boundary of P_i, they can be covered by k_i triangles, and there are exactly two ways to cover these blue points by k_i triangles, as earlier. Following a similar, but more involved, argument, we can prove that the set of all blue points can be covered by $\sum_{i=1}^{n} k_i$ triangles if and only if F is satisfiable.

Theorem 2.2 *The* 3*-dimensional surface-approximation problem is* NP-Hard.

Proof: Our construction is similar in spirit to the one for the bichromatic partition problem, albeit slightly more complex in detail. We use points of three colors: red, white and black. The 'white' points lie on the plane $z = 0$, the 'black' points lie on the plane $z = 2A$, and the 'red' points lie between $z = 0$ and $z = A$, where A is a sufficiently large constant. To maintain a connection with the previous construction, the black and white points play the role of blue points, while the red points play the role of enforcers as before, restricting the choice of "legal" triangles that can cover the black or white points. We will describe the construction in the xy-plane, which represents the orthogonal projection of the actual construction. The actual construction is obtained simply by vertically translating each point to its correct plane, depending on its color.

We start out again by putting a 'black' point at each clause node C_j. Then, for each variable x_i, we construct the "star-shaped" polygon P_i; this part is identical to the previous construction. We replace each of the two straight-line edges of P_i by "concave chains," bent inward, and also make a small "dent" at the tip of the circular arc s_{ij}, as shown in Figure 2. We place 12 points on each arm of P_i, alternating in color black and white, as follows. At the tip of the circular arc s_{ij}, we put a white point c_{ij} at the outer endpoint of the dent and a black point d_{ij} at the inner endpoint of the dent (Figure 2 (ii)). The rest of the construction is shown in Figure 2 (i) — we put two more points a_{ij}, b_{ij} on the circular arc and 4 points $\alpha^l_{ij}, \beta^l_{ij}$ $(1 \leq l \leq 4)$ on each of the two concave chains. The two points

surrounding c_{ij}, d_{ij}, namely, a_{ij} and b_{ij}, are such that any segment connecting them to any point on the two concave chains lies inside P_i. Next, corresponding to each edge e_{ij} of the graph $G(F)$, we put a 'white' point c'_{ij} on the segment joining c_{ij} and the clause point C_j, very close to c_{ij} such that

$$d(c_{ij}, c'_{ij}) \leq \frac{\varepsilon}{2A} d(c_{ij}, C_j)\,.$$

(See Figure 2 (iii).) This condition says that, in the final construction when the black and white points have been translated to their correct z-plane, the vertical distance between c'_{ij} and the segment $C_j c_{ij}$ is no more than ε—recall that ε is the input measure of approximation.

This completes the placement of white and black points. The only remaining part of the construction is the placement of 'red' points, which we now describe.

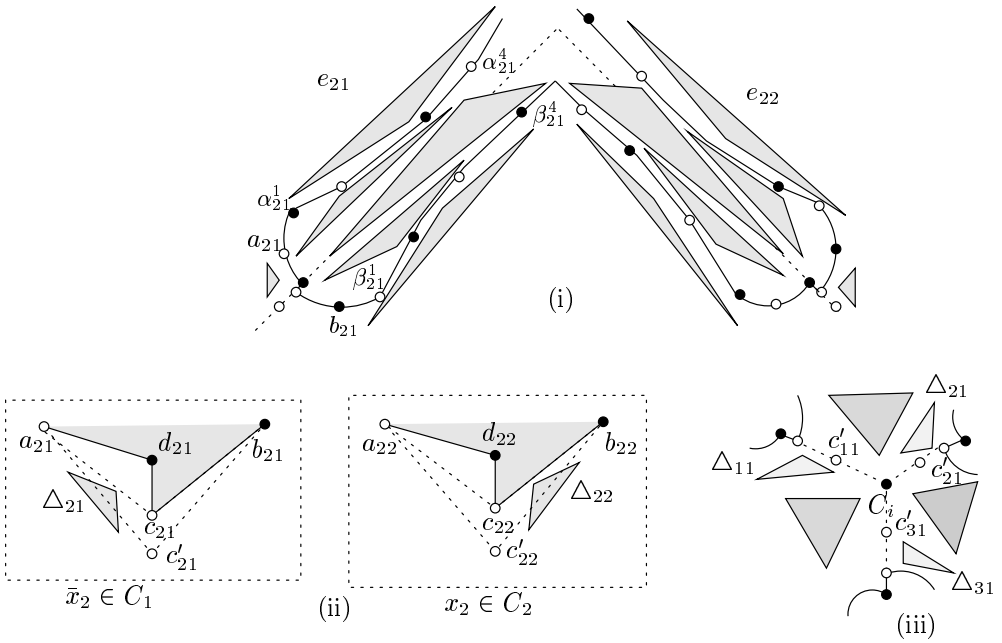

Figure 2: Placing points on the polygon P_2 corresponding to Figure 1: (i) Modified P_2 and points on P_2, (ii) points and triangles in the neighborhood of c_{21}, c_{22}; (iii) points and triangles near C_1.

We add a set of triangles, each containing a large (but polynomially bounded) number of red points—the role of these triangles is to restrict the choice of legal triangles that can cover black/white points. The set of triangles associated with P_i is labeled T_i. The construction of T_i is detailed in Figure 2 (i). Specifically, for an edge e_{ij}, if $x_i \in C_j$, then we put a small triangle that intersects the segment $b_{ij}c'_{ij}$ but not $b_{ij}c_{ij}$. On the other hand, if $\bar{x}_i \in C_j$,

then we put a small triangle that intersects $a_{ij}c'_{ij}$ but not $a_{ij}c_{ij}$. Next, we put a small number of triangles inside P_i, near its concave chains, so that at most three consecutive points along P_i may be covered by one triangle without intersecting any triangle of T_i. We ensure that one of these triangles intersects the triangle $\triangle\alpha^1_{ij}a_{ij}d_{ij}$, so that $\{\alpha^1_{ij}, a_{ij}, d_{ij}\}$ cannot be covered by a single triangle. We also place three triangles near each clause C_j, each containing a large number of red points; see Figure 2 (iii). Finally, we translate black and white points in the z-direction, as described earlier. Let $\{\tau_1, \ldots, \tau_t\}$ be the set of all 'red' triangles. We move all points in τ_i vertically to the plane $z = \frac{A}{4}(1 + \frac{2i}{t})$. There are two ways to cover the points of P_i with $2k_i$ legal (non-intersecting) triangles—one in which a_{ij}, d_{ij}, c_{ij} are covered by a single triangle, and the one in which b_{ij}, d_{ij}, c_{ij} are covered by a triangle. These coverings are associated with the true and false settings of the variable x_i.

Let P denote the set of all points constructed, let t denote the total number of 'red' triangles, and let $k = 2\sum_{i=1}^{n} k_i$. We claim that there exist a polyhedral terrain with $3(k+m+t)$ vertices that ε-approximates P if and only if F has a truth assignment, provided that ε is sufficiently small—recall that m is the number of clauses in F. The claim follows from the observations that it is always better to cover all red points lying in a horizontal triangle τ_i by τ_i itself, and that a clause C_j requires one triangle to cover its points if and only if one of the literals in C_j is set true; otherwise it requires two triangles. (For instance, if C_j contains the literal x_i and x_i is set true, then the triangle a_{ij}, d_{ij}, c_{ij} can be enlarged slightly to cover c'_{ij}. The remaining three points for the clause C_j can be covered by one additional triangle.) The rest of the argument is the same as for the bichromatic partition problem. Finally, we can perturb the points slightly so that no four of them are coplanar. □

3 A Canonical Trapezoidal Partition

We introduce an abstract geometric partitioning problem in the plane, which captures the essence of both the surface-approximation problem as well as the bichromatic points partition problem. The abstract problem deals with trapezoidal partitions under a boolean constraint function $\mathcal{C}$ satisfying the "subset restriction" property. More precisely, let $\mathcal{C}$ be a boolean function from compact, connected subsets of the plane to $\{0, 1\}$ satisfying the following property:

$$\boxed{\mathcal{C}(U) = 1 \implies \mathcal{C}(V) = 1, \qquad \text{for all} \quad V \subseteq U \subseteq \mathbb{R}^2.} \tag{3.1}$$

For technical reasons, we choose to work with "trapezoids" instead of triangles, where the top and bottom edges of each trapezoid are parallel to the x-axis. The trapezoids and triangles are equivalent for the purpose of approximation—each triangle can be decomposed into two trapezoids, and each trapezoid can be decomposed into two triangles.

Given a set of n points P in the plane, a family of trapezoids $\mathbf{\Delta} = \{\Delta_1, \ldots, \Delta_m\}$ is called a *valid trapezoidal partition* (a *trapezoidal partition* for brevity) of P with respect to a boolean constraint function $\mathcal{C}$ if the following conditions hold:

(i) $\mathcal{C}(\Delta) = 1$, for all $\Delta \in \mathbf{\Delta}$,

(ii) $\mathbf{\Delta}$ covers all the points: $P \subset \bigcup_{i=1}^{m} \Delta_i$, and

(iii) The trapezoids in $\mathbf{\Delta}$ have pairwise disjoint interiors.

We can cast our bichromatic partition problem in this abstract framework by setting $P = B$ (the set of 'blue' points) and, for a trapezoid $\tau \subset \mathbb{R}^2$, defining $\mathcal{C}(\tau) = 1$ if and only if τ is empty of red points, that is, $\tau \cap R = \emptyset$. In the surface-approximation problem, we set $P = \bar{S}$ (the orthogonal projection of S on the plane $z = 0$) and a trapezoid $\tau \subset \mathbb{R}^2$ has $\mathcal{C}(\tau) = 1$ if and only if τ can be vertically lifted to a planar trapezoid $\hat{\tau}$ in $\mathbb{R}^3$ so that the vertical distance between $\hat{\tau}$ and any point of S directly above/below it is at most ε.

The space of optimal solutions for our abstract problem is potentially infinite—the vertices of the triangles in our problem can be anywhere in the plane. For our approximation results, however, we show that a restricted choice of trapezoids suffices.

Given a set of n points P in the plane, let $\mathcal{L}(P)$ denote the set consisting of the following lines: the horizontal lines passing through a point of P, and the lines passing through two points of P. Thus, $|\mathcal{L}(P)| = O(n^2)$. We will call the lines of $\mathcal{L}(P)$ the *canonical lines* determined by P. We say that a trapezoid $\Delta \subset \mathbb{R}^2$ is *canonical* if all of its edges belong to lines in $\mathcal{L}(P)$. A trapezoidal partition $\mathbf{\Delta}$ is *canonical* if all of its trapezoids are canonical. The following lemma shows that by limiting ourselves to canonical trapezoidal partitions only, we sacrifice at most a constant (multiplicative) factor in our approximation.

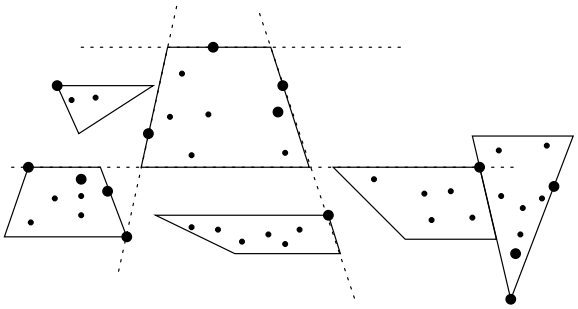

Figure 3: A canonical trapezoidal partition

Lemma 3.1 *Any trapezoidal partition of P with k trapezoids can be transformed into a canonical trapezoidal partition of P with at most $4k$ trapezoids.*

Proof: We give a construction for transforming each trapezoid $\Delta \in \mathbf{\Delta}$ into four trapezoids $\Delta_i \subseteq \Delta$, for $1 \leq i \leq 4$, with pairwise disjoint interiors, so that Δ_i together cover all the points in $P \cap \Delta$. By (3.1), the new set of trapezoids is a valid trapezoidal partition of P. Our construction works as follows.

Consider the convex hull of the points $P_\Delta = P \cap \Delta$. If the convex hull itself is a trapezoid, we return that trapezoid. Otherwise, let ℓ, r, t, b denote the left, right, top and bottom edges of Δ, as shown in Figure 4 (i). We perform the following four steps, which constitute our transformation algorithm.

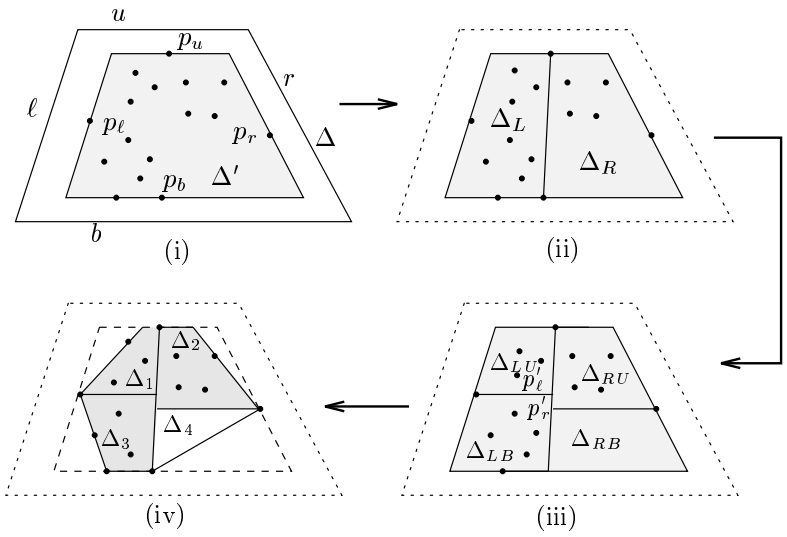

Figure 4: Illustration of the canonicalization.

(i) We shrink the trapezoid Δ by translating each of its four bounding edges towards the interior, until it meets a point of P_Δ. Let $\Delta' \subseteq \Delta$ denote the smaller trapezoid thus obtained (Figure 4 (i)). Let p_ℓ, p_r, p_t, p_b, respectively, denote a point of P_Δ lying on the left, right, top, and bottom edge of Δ'; we break ties arbitrarily if more than one point lies on an edge.

(ii) We partition Δ' into two trapezoids, Δ_L and Δ_R, by drawing the line segment $p_u p_b$, as shown in (Figure 4 (ii)).

(iii) We next partition Δ_L into two trapezoids Δ_{LU} and Δ_{LB}, by drawing the maximal horizontal segment through p_ℓ. Let p'_ℓ denote right endpoint of this segment. Similarly, we partition Δ_R into Δ_{RU} and Δ_{RB}, lying respectively above and below the horizontal line segment $p_r p'_r$.

(iv) We rotate the line supporting the left boundary of Δ_{LU} around the point p_ℓ in clockwise direction until it meets a point of the set $P \cap \Delta_{LU}$. Let q_ℓ denote the intersection

of this line and the top edge of Δ_{LU}. We set $\Delta_1 = p_\ell q_\ell p_u p'_\ell$ (Figure 4 (iv)). (If $q_\ell = p_u$, then Δ_1 is a triangle, which we regard as a degenerate trapezoid; e.g. Δ_4 in Figure 4 (iii).) The top and bottom edges of Δ_1 contain p_u and p_ℓ, respectively, the left edge contains p_ℓ and q_ℓ, and the right edge is determined by the pair of points p_u and p_b. Thus, the trapezoid Δ_1 is in canonical position. The three remaining trapezoids $\Delta_2, \Delta_3, \Delta_4$ are constructed similarly.

In the above construction, if any of the four trapezoids Δ_i does not cover any point of P_Δ, then we can discard it. Thus, an arbitrary trapezoid of the partition $\mathbf{\Delta}$ can be transformed into at most four canonical trapezoids. This completes the proof of the lemma. □

4 Greedy Algorithms

At this point, we can obtain a weak approximation result using the canonical trapezoidal partition. Roughly speaking, we can use the greedy set covering heuristic [7, 15], ignoring the disjointness constraint, and then refine the heuristic output to produce disjoint trapezoids. Unfortunately, the last step can increase the complexity of the solution quite significantly.

Theorem 4.1 *Given a set P of n points in the plane and a boolean constraint function $\mathcal{C}$ satisfying (3.1) that can be evaluated in polynomial time, we can compute an $O({K_o}^2 \log {K_o}^2)$ size trapezoidal partition of P respecting $\mathcal{C}$ in polynomial time, where K_o is the size of an optimal trapezoidal partition.*

Proof: Consider the set Ξ of all valid, canonical trapezoids in the plane—the set Ξ has $O(n^6)$ trapezoids. We form the geometric set-system

$$X = (P, \quad \{ P \cap \Delta \mid \Delta \in \Xi \ \& \ \mathcal{C}(\Delta) = 1 \}) .$$

X can be computed by testing each $\Delta \in \Xi$ whether it is valid. We compute a set cover of X using the greedy algorithm [12, 15] in polynomial time. The greedy algorithm returns a set R consisting of $O(K_o \log K_o)$ trapezoids, not necessarily disjoint. In order to produce a disjoint cover, we first compute the arrangement $\mathcal{A}(R)$ of the plane induced by R. Then, we decompose each face of $\mathcal{A}(R)$ into trapezoids by drawing a horizontal segment through each vertex until the segment hits an edge of the arrangement. The resulting partition is a trapezoidal partition of P. The number of trapezoids in the partition is $O((K_o \log K_o)^2)$ — since the arrangement $\mathcal{A}(R)$ has this size. The total running time of the algorithm is polynomial. □

Remark: (i) The canonical form of trapezoids is used only to construct a finite family of trapezoids to search for an approximate solution. A direct application of the definition in

the previous subsection gives a family of $O(n^6)$ canonical trapezoids. By using a slightly different canonical form, we can reduce the size of canonical triangles to $O(n^4)$.

(ii) One can show that the number of trapezoids produced by the above algorithm is $\Omega({K_o}^2)$ in the worst case.

5 A Recursively Separable Partition

We now develop our main approximation algorithm. The algorithm is based on dynamic programming, and it depends on two key ideas—a *recursively separable* partition and a *compliant* partition.

A trapezoidal partition $\mathbf{\Delta}$ is called *recursively separable* if the following holds:

- $\mathbf{\Delta}$ consists of a single trapezoid, or
- there exists a line ℓ such that (i) ℓ does not intersect the interior of any trapezoid in $\mathbf{\Delta}$, (ii) $\mathbf{\Delta}^+ = \mathbf{\Delta} \cap \ell^+$ and $\mathbf{\Delta}^- = \mathbf{\Delta} \cap \ell^-$ are both nonempty, where ℓ^+ and ℓ^- are the two half-planes defined by ℓ, and (iii) each of $\mathbf{\Delta}^+$ and $\mathbf{\Delta}^-$ is recursively separable.

The following key lemma gives an upper bound on the penalty incurred by our approximation algorithm if only recursively separable trapezoidal partitions are used.

Lemma 5.1 *Let P be a finite set of points in the plane and let $\mathbf{\Delta}$ be a trapezoidal partition of P with k trapezoids. There exists a recursively separable partition $\mathbf{\Delta}^\star$ of P with $O(k \log k)$ trapezoids. In addition, each separating line is either a horizontal line passing through a vertex of $\mathbf{\Delta}$ or a line supporting an edge of a trapezoid in $\mathbf{\Delta}$.*

Proof: We present a recursive algorithm for computing $\mathbf{\Delta}^\star$. Our algorithm is similar to the binary space partition algorithm proposed by Paterson and Yao [18]. We assume that the boundaries of the trapezoids in $\mathbf{\Delta}$ are also pairwise disjoint—this assumption is only to simplify our proof.

At each recursive step of the algorithm, the subproblem under consideration lies in a trapezoid T. (This containing trapezoid may degenerate to a triangle, or it may even be unbounded.) The top and bottom edges of T (if they exist) pass through the vertices of $\mathbf{\Delta}$, while the left and right edges (if they exist) are portions of edges of $\mathbf{\Delta}$. Initially T is set to an appropriately large trapezoid containing the family $\mathbf{\Delta}$. Let $\mathbf{\Delta}_T$ denote the trapezoidal partition of $P \cap T$ obtained by intersecting $\mathbf{\Delta}$ with T, and let V_T be the set of vertices of $\mathbf{\Delta}_T$ lying in the interior of T. An edge of $\mathbf{\Delta}_T$ cannot intersect the left or right edge of T, because they are portions of the edge of T. Therefore, each edge of $\mathbf{\Delta}_T$ either lies in the interior of T, or intersects only the top and bottom edges of T.

If $|\mathbf{\Delta}_T| = 1$, we return $\mathbf{\Delta}_T$ and stop. Otherwise, we proceed as follows. If there is a trapezoid $\triangle \in \mathbf{\Delta}_T$ that completely crosses T (that is, its vertices lie on the top and bottom edges of T), then we do the following. If $\triangle$ is the leftmost trapezoid of $\mathbf{\Delta}_T$, then we partition T into two trapezoids T_1, T_2 by drawing a line through the right edge of Δ, so that T_1 contains $\triangle$ and T_2 contains the remaining trapezoids of Δ_T. If $\triangle$ is not the leftmost trapezoid of $\mathbf{\Delta}_T$, then we partition T into T_1, T_2 by drawing a line through the left edge of Δ.

If every trapezoid Δ in $\mathbf{\Delta}_T$ has at least one vertex in the interior of T, and so the previous condition is not met, then we choose a point $v \in V_T$ with a median y-coordinate. We partition T into trapezoids T_1, T_2 by drawing a horizontal line ℓ_v passing through v. Each trapezoid $\Delta \in \mathbf{\Delta}_T$ that crosses ℓ_v is partitioned into two trapezoids by adding the segment $\Delta \cap \ell_v$. At the end of this dividing step, let $\mathbf{\Delta}_1$ and $\mathbf{\Delta}_2$ be the set of trapezoids of that lie in T_1 and T_2, respectively. We recursively refine $\mathbf{\Delta}_1$ and $\mathbf{\Delta}_2$ into separable partitions $\mathbf{\Delta}_1^\star$ and $\mathbf{\Delta}_2^\star$, respectively, and return $\mathbf{\Delta}_T^\star = \mathbf{\Delta}_1^\star \cup \mathbf{\Delta}_2^\star$. This completes the description of the algorithm.

We now prove that $\mathbf{\Delta}^*$ satisfies the properties claimed in the lemma. It is clear that $\mathbf{\Delta}^*$ is recursively separable and that each separating line of $\mathbf{\Delta}^*$ either supports an edge of $\mathbf{\Delta}$ or it is horizontal. To bound the size of $\mathbf{\Delta}^*$, we charge each trapezoid of $\mathbf{\Delta}^*$ to its bottom-left vertex. Each such vertex is either a bottom-left vertex of a trapezoid of $\mathbf{\Delta}$, or it is an intersection point of a left edge of a trapezoid of $\mathbf{\Delta}$ with the extension of a horizontal edge of another trapezoid of $\mathbf{\Delta}$. There are only k vertices of the first type, so it suffices to bound the number of vertices of the second type. Since the algorithm extends a horizontal edge of a trapezoid of $\mathbf{\Delta}_T$ only if every trapezoid of $\mathbf{\Delta}_T$ has at least one vertex in the interior of T, and we always extend a horizontal edge with a median y-coordinate, it is easily seen that the number of vertices of the second type is $O(k \log k)$. This completes the proof. □

Remark: Given a family $\mathbf{\Delta}$ of k disjoint orthogonal rectangles partitioning P, we can find a set of $O(k)$ recursively separable rectangles that forms a *rectangular* partition of P—this uses the *orthogonal* binary space partition algorithm of Paterson and Yao [19].

6 An Approximation Algorithm

Lemma 5.1 applies to any trapezoidal partition of P. In particular, if we start with a canonical trapezoidal partition $\mathbf{\Delta}$, then the output partition $\mathbf{\Delta}^\star$ is both canonical and recursively separable, and each separating line in $\mathbf{\Delta}^\star$ belongs to the family of canonical lines $\mathcal{L}(P)$. For the lack of a better term, we call a trapezoidal partition of P that satisfies these conditions a *compliant partition*. Lemmas 3.1 and 5.1 together imply the following useful theorem.

Theorem 6.1 *Let P be a set of n points in the plane and let $\mathcal{C}$ be a boolean constraint function satisfying the condition (3.1). If there is a trapezoidal partition of P respecting $\mathcal{C}$ with k trapezoids, then there is a compliant partition of P also respecting $\mathcal{C}$ with $O(k \log k)$ trapezoids.*

In the remainder of this section, we give a polynomial-time algorithm, using dynamic programming, for constructing an optimal compliant partition. By Theorem 6.1, this partition has $O(K_o \log K_o)$ trapezoids. Recall that the set $\mathcal{L} = \mathcal{L}(P)$ consists of all canonical lines determined by P.

Consider a subset of points $R \subseteq P$, and a canonical trapezoid Δ containing R. Let $\sigma(R, \Delta)$ denote the size of an optimal compliant partition of R in Δ; the size of a partition is the number of trapezoids in the partition. Theorem 6.1 gives the following recursive definition of σ:

$$\sigma(R, \Delta) = \begin{cases} 1 & \text{if } \mathcal{C}(\Delta) = 1 \\ \min_{\ell} \{\sigma(R^+, \Delta^+) + \sigma(R^-, \Delta^-)\} & \text{otherwise,} \end{cases}$$

where the minimum is over all those lines $\ell \in \mathcal{L}$ that are either horizontal and intersect Δ, or intersect both the top and bottom edges of Δ; ℓ^+ and ℓ^- denote the positive and negative half-planes induced by ℓ, $R^+ = R \cap \ell^+$ and $R^- = R \cap \ell^-$. The goal of our dynamic programming algorithm is to compute $\sigma(P, T)$, for some canonical trapezoid T enclosing all the points P. We now describe how the dynamic programming systematically computes the required partial answers.

Every canonical trapezoid Δ in the plane can be described (uniquely) by a 6-tuple $(i, j, k_1, k_2, l_1, l_2)$ consisting of integers between 1 and n. The first two numbers fix two points p_i and p_j through which the lines containing the top and bottom edges of Δ pass; the second pair fixes the points p_{k_1}, p_{k_2} through which the line containing the left edge of Δ passes; and the third pair fixes the points p_{l_1}, p_{l_2} through which the line containing the right edge of Δ passes. (In case of ties, we may choose the points closest to the corners of Δ.) We use the notation $\Delta(i, j, k_1, k_2, l_1, l_2)$ for the trapezoid associated with the 6-tuple $(i, j, k_1, k_2, l_1, l_2)$. If the 6-tuple does not produce a trapezoid, then $\Delta(i, j, k_1, k_2, l_1, l_2)$ is undefined.

Let $P(i, j, k_1, k_2, l_1, l_2) = P \cap \Delta(i, j, k_1, k_2, l_1, l_2)$. We use the abbreviated notation $\sigma(i, j, k_1, k_2, l_1, l_2)$ to denote the size of an optimal compliant partition for the points contained in $\Delta(i, j, k_1, k_2, l_1, l_2)$:

$$\sigma(i, j, k_1, k_2, l_1, l_2) = \sigma\Big(P(i, j, k_1, k_2, l_1, l_2),\ \Delta(i, j, k_1, k_2, l_1, l_2)\Big).$$

The quantity $\sigma(i, j, k_1, k_2, l_1, l_2)$ is undefined if the trapezoid $\Delta(i, j, k_1, k_2, l_1, l_2)$ is undefined. If the points in P are sorted in increasing order of their y-coordinates, then $\Delta(i, j, k_1, k_2, l_1, l_2)$ is defined only for $i \geq j$. Our dynamic programming algorithm computes the σ values as follows.

If $\mathcal{C}(\Delta(i,j,k_1,k_2,l_1,l_2)) = 1$, then

$$\sigma(i,j,k_1,k_2,l_1,l_2) = 1\,.$$

Otherwise,

$$\sigma(i,j,k_1,k_2,l_1,l_2) \;=\; \min\left\{ \begin{array}{l} \min\limits_{i\le u\le j}\{\sigma(i,u,k_1,k_2,l_1,l_2)+\sigma(u+1,j,k_1,k_2,l_1,l_2)\}, \\ \min\limits_{v_1,v_2}\{\sigma(i,j,k_1,k_2,v_1,v_2)+\sigma(i,j,v_1,v_2,l_1,l_2)\} \end{array} \right\},$$

where the last minimum varies over all pairs of points v_1, v_2 such that the line passing through them intersects both the top and the bottom edge of $\Delta(i,j,k_1,k_2,l_1,l_2)$.

If Ξ denotes the set of all canonical trapezoids, then $|\Xi| = O(n^6)$—each 6-tuple is associated with at most one unique trapezoid. If $Q(n)$ denotes the time to decide whether $\mathcal{C}(\Delta) = 1$ for an arbitrary trapezoid Δ, then we can initially compute all trapezoids for which $\mathcal{C}(\Delta) = 1$ in total time $O(n^6 Q(n))$ — for these trapezoids, we initially set $\sigma(\Delta) = 1$. For all the remaining trapezoids in Ξ, we use the recursive formula presented above to compute their σ. Computing σ for a trapezoid requires computing the minimum of $O(n^2)$ quantities. Thus the total running time of the algorithm is $O(n^8)$. The following theorem states the main result of our paper.

Theorem 6.2 *Given a set P of n points in the plane and a boolean constraint $\mathcal{C}$ satisfying condition (3.1), we can compute a geometric partition of P with respect to $\mathcal{C}$ using $O(K_o \log K_o)$ trapezoids, where K_o is the number of trapezoids in an optimal partition. Our algorithm runs in worst-case time $O(n^8 + n^6 Q(n))$, where $Q(n)$ is the time to decide whether $\mathcal{C}(R) = 1$ for any subset $R \subseteq P$.*

Remark: By computing σ's in a more clever order and exploiting certain geometric properties of a geometric partition, the time complexity of the above algorithm can be improved by one order of magnitude. This minor improvement, however, doesn't seem worth the effort needed to explain it.

Theorem 6.2 immediately implies polynomial-time approximation algorithms for the surface-approximation and the planar bichromatic-partition problem. In the case of the surface-approximation problem, deciding $\mathcal{C}(\Delta)$ for a trapezoid Δ requires checking whether there is a plane π in $\mathbb{R}^3$ such that the vertical distance between π and the points covered by Δ is at most ε. This problem can be solved in linear time using the fixed-dimensional linear programming algorithm of Megiddo [16]. (A more practical algorithm, running in time $O(n \log n)$, is the following. Let $A \subseteq P$ denote the set of points covered by the trapezoid Δ. For a point $p \in A$, let p^+ and p^-, respectively, denote the point p translated vertically up and down by ε. Let $A^+ = \{p^+ \mid p \in A\}$ and $A^- = \{p^- \mid p \in A\}$. Then, $\mathcal{C}(\Delta) = 1$ if and only if sets A^+ and A^- can be separated by a plane. The two sets are separable if their convex hulls are disjoint. This can be tested in $O(n \log n)$ time—for instance, see the book by Preparata and Shamos [20].)

Theorem 6.3 *Given a set S of n points in $\mathbb{R}^3$ and a parameter $\varepsilon > 0$, in polynomial time we can compute an ε-approximate polyhedral terrain Σ with $O(K_o \log K_o)$ vertices, where K_o is the number of vertices in an optimal terrain. Our algorithm runs in $O(n^8)$ worse-case time.*

In the planar bichromatic partition problem, deciding whether $\mathcal{C}(\Delta) = 1$ requires checking whether Δ contains any point from the red set R. This can clearly be done in $O(n)$ time. Alternatively, R can be preprocessed in time $O(n^{2+\delta} \log^{O(1)} n)$, for any $\delta > 0$, into a triangle range-searching data structure, which can determine in $O(\log n)$ time whether a query trapezoid contains any point of R; see e.g. [6]. Our main purpose, however, is to only establish a polynomial time bound for the approximation algorithm.

Theorem 6.4 *Given a set R of 'red' points and another set B of 'blue' points in the plane, we can find in polynomial time a set of $O(K_o \log K_o)$ disjoint triangles that cover B but do not contain any red point; K_o is the number of triangles in an optimal solution.*

Remark: In view of the remark following Theorem 6.1, given a set R of 'red' points and another set B of 'blue' points in the plane, we can find in polynomial time a set of $O(K_o)$ disjoint orthogonal rectangles that cover B but do not contain any red point. In this case, the time complexity improves by a few orders of magnitude, because there are only $O(n^4)$ canonical rectangle and each rectangle is subdivided into two rectangles by drawing a horizontal or vertical line passing through one of the points. Omitting all the details, the running time in this case is $O(n^5)$.

We conclude this section by noting that Theorem 6.4 can be used to construct small size linear decision trees in the plane. A *linear decision tree* in $\mathbb{R}^d$ is a full binary tree, of which each leaf is labeled $+1$ or -1 and each interior node v is associated with a $(d-1)$-hyperplane h_v; the left (resp. right) child of v is also associated with the halfspace lying above (resp. below) h_v. T is used to classify a point $p \in \mathbb{R}^d$ by traversing a path of T, starting from the root, as follows: Suppose v is the node being currently visited. If v is a leaf, return the label of v. Otherwise, if p lies above (resp. below) h_v, we recursively visit the left (resp. right) child of v. T is *consistent* with B and R if it assigns $+1$ to all the points of B and -1 to all the points of R. Each leaf of T corresponds to a convex polyhedron, which is the intersection of the halfspaces associated with its ancestors, and these regions induce a convex partition of $\mathbb{R}^d$. For $d \leq 3$, a convex polyhedron can be triangulated into linear number of simplices. Hence for $d \leq 3$, the size of an optimal linear decision tree that is consistent with R and B is $\Omega(K_o)$, where K_o is the size of an optimal bichromatic partition for R and B. See [21] for a survey of known results on decision trees.

Using Theorem 6.4, we can construct a linear decision tree of size $O(K_o \log K_o)$ for R and B as follows. Let $\mathbf{\Delta}$ be the partition computed by the above algorithm. If $\mathbf{\Delta}$ consists of a single trapezoid, say τ, then we construct a linear decision tree with 4 interior nodes, each

associated with the line supporting one of the edges of τ. The leaves of the tree partition the plane into 5 regions, of which one is τ itself. The leaf corresponding to τ is labeled $+1$ and the others are labeled -1. If $|\mathbf{\Delta}| > 1$, then there is a separating line ℓ; let $\mathbf{\Delta}^+$ (resp. $\mathbf{\Delta}^-$) be the subset of trapezoids lying above (resp. below) ℓ. We associate the root with ℓ, recursively construct the decision trees for $\mathbf{\Delta}^+$ and $\mathbf{\Delta}^-$, and attach them as the left and right subtrees of the root. Putting everything together, we obtain the following result.

Theorem 6.5 *Given a set R of 'red' points and another set B of 'blue' points in the plane, we can construct in polynomial time a linear decision tree of size $O(K_o \log K_o)$, which is consistent with R and B; K_o is the size of an optimal linear decision tree that is consistent with R and B.*

7 Extensions

We can extend our algorithm to a slightly stronger form of surface approximation. In the basic problem, the implicit function (surface) is represented by a set of sample points S. What if the sample consists of two-dimensional compact, connected pieces? In this section, we show an extension of our algorithm that deals with the case when the sample consists of a set $\mathcal{T}$ of n *horizontal triangles* with pairwise disjoint xy-projection. (Since any polygon can be decomposed into triangles, this case also handles polygons.) Our goal is to compute a polyhedral terrain Σ, so that the vertical distance between any point in $T_i \in \mathcal{T}$ and Σ is at most ε. We produce a terrain Σ with $O(K_o \log K_o)$ vertices, where K_o again is the number of vertices in an optimal such surface.

Let $\mathcal{T} = \{T_1, \ldots, T_n\}$ be the input set of n horizontal triangles in $\mathbb{R}^3$ with the property that their vertical projections on the plane $z = 0$ are pairwise disjoint. We will consistently use the following notational convention: for an object $s \in \mathbb{R}^3$, $\bar{s}$ denotes its orthogonal projection on the plane $z = 0$, and for a subset $g \subseteq \bar{s}$, $\hat{g}$ denotes the portion of s in $\mathbb{R}^3$ such that $g = \bar{\hat{g}}$. Abusing the notation slightly, we say that a set Ξ of trapezoids (or triangles) in $\mathbb{R}^3$ *ε-approximates* $\mathcal{T}$ within a region $Q \subseteq \mathbb{R}^2$ if the vertical distance between $\mathcal{T}$ and Ξ in Q is at most ε and the vertical projections of trapezoids of Ξ are disjoint on $z = 0$.

Let S denote the set of vertices of the triangles in $\mathcal{T}$, and let $\bar{S}$ be their orthogonal projection on $z = 0$. We set $P = \bar{S}$, as the set of points in our abstract problem. The constraint function is defined as follows. Given a trapezoid $\Delta \in \mathbb{R}^3$, we have $\mathcal{C}(\Delta) = 1$ if and only if the vertical distance between Δ and any point in $\bigcup_{i=1}^n T_i$ directly above/below Δ is at most ε. (Thus, while the point set P includes only the vertices of $\mathcal{T}$, the constraint set takes into consideration the whole triangles.) The constraint $\mathcal{C}$ satisfies (3.1), and it can be computed in polynomial time.

It is also clear that the size of an optimal trapezoidal partition of P with respect to $\mathcal{C}$ is a lower bound on the size of a similar partition for $\mathcal{T}$, the set of triangles. We first

apply Theorem 6.3 to obtain a family $\mathbf{\Delta}$ of $O(k \log k)$ trapezoids that ε-approximates P with respect to $\mathcal{C}$; clearly $k \leq K_o$. The next step of our algorithm is to extend $\mathbf{\Delta}$ to a polyhedral terrain that ε-approximates the triangles of $\mathcal{T}$. Care must to be exercised in this step if one wants to add only $O(k \log k)$ new trapezoids. In the second step, we work with the projection of $\mathcal{T}$ and Δ in the plane $z = 0$.

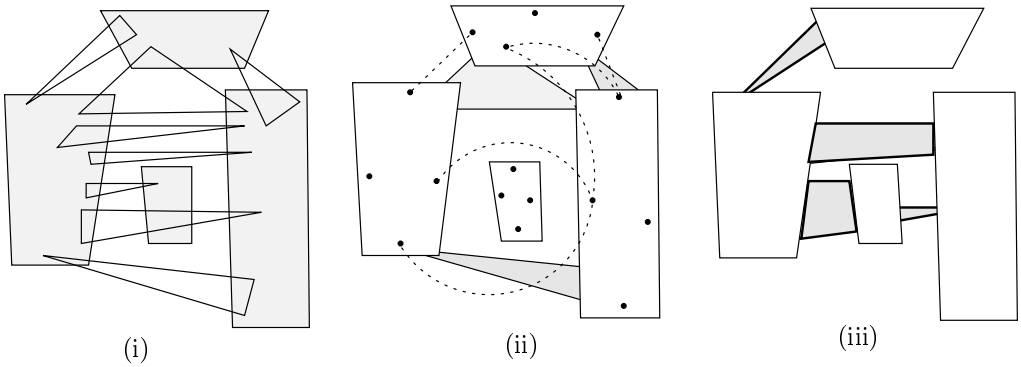

Figure 5: (i) $\bar{\mathcal{T}}$ and $\bar{\mathbf{\Delta}}$, (ii) R_1 and G; (iii) R_2 and Q_i's.

Let $\bar{\mathbf{\Delta}} = \{\bar{\Delta} \mid \Delta \in \mathbf{\Delta}\}$ and $\bar{\mathcal{T}} = \{\bar{T} \mid T \in \mathcal{T}\}$. Let R be the set of connected components of the closure of $\bigcup_{T \in \mathcal{T}} \bar{T} \setminus \bigcup_{\Delta \in \mathbf{\Delta}} \bar{\Delta}$. That is, R is the portion of $\bigcup_{T \in \mathcal{T}} \bar{T}$ lying in the common exterior of $\bar{\mathbf{\Delta}}$, as shown in Figure 5 (i). R is a collection of *simple* polygons, each of which is contained in a triangle of $\bar{\mathcal{T}}$. Since the corners of the triangles of $\bar{\mathcal{T}}$ are covered by $\bar{\mathbf{\Delta}}$, the vertices of all polygons in R lie on the boundary of $\bar{\mathbf{\Delta}}$, and each edge of R is contained in an edge of $\bar{\mathbf{\Delta}}$ or of $\bar{\mathcal{T}}$. Let $R_1 \subseteq R$ be the subset of polygons that touch at least three edges of trapezoids in $\bar{\mathbf{\Delta}}$, and let $R_2 = R - R_1$.

For each polygon $P_i \in R_1$, we compute a set of triangles that ε-approximate $\mathcal{T}$ within P_i. For a vertex $v \in P_i$, lying on the boundary of a trapezoid $\bar{\Delta}$, let $\hat{v}$ denote the point on Δ whose xy-projection is v. Let $\bar{T}_i$ be the triangle containing P_i. We triangulate P_i, and, for each triangle $\triangle abc$ in the triangulation, we pick $\triangle \hat{a}\hat{b}\hat{c}$. Since T_i is parallel to the xy-plane, it can be proved that the maximum vertical distance between $\triangle \hat{a}\hat{b}\hat{c}$ and T_i is ε. We repeat this step for all polygons in R_1. The number of triangles generated in this step is proportional to the number of vertices in the polygons of R_1, which we bound in the following lemma.

Lemma 7.1 *The total number of vertices in the polygons of R_1 is $O(k \log k)$.*

Proof: Each vertex of a polygon in R_1 is either (i) a vertex of a trapezoid in $\bar{\mathbf{\Delta}}$, or (ii) an intersection point of an edge of $\bar{\mathbf{\Delta}}$ with an edge of a triangle in $\mathcal{T}$. There are only $O(k \log k)$ vertices of type (i), so it remains to bound the number of vertices of type (ii).

We construct an undirected graph $G = (V, E)$, as follows. Let $\Gamma = \{\gamma_1, \ldots, \gamma_t\}$ be the set of edges in $\bar{\mathbf{\Delta}}$.[1] To avoid confusion, we will call the edges of Γ *segments* and those of E *arcs*. For each segment γ_i, we place a point i close to γ_i, inside the trapezoid bounded by γ_i. The set of resulting points forms the node set V. If there is an edge pq of a polygon in R_1 such that $p \in \gamma_i$ and $q \in \gamma_j$, we add the arc (i, j) to E; see Figure 5 (ii). It is easily seen that G is a planar graph, and that $|E| = O(k \log k)$. Fix a pair of segments $\gamma_1, \gamma_2 \in \Gamma$ such that $(1, 2) \in E$. Let $E_{12} = \{p_1q_1, p_2q_2, \ldots\}$ be the set of edges in R_1, sorted either left to right or top to bottom, as the case may be, that are incident to γ_1 and γ_2. Let $|E_{12}| = m_{ij}$. Assume that for every $1 \leq i \leq m_{ij}$, p_i lies on γ_1 and q_i lies on γ_2. The number of vertices of type (ii) is obviously $2\sum_{(i,j)\in E} m_{ij}$. We call two edges $p_iq_i,\ p_jq_j \in E_{12}$ *equivalent* if the interior of the convex hull of p_iq_i and p_jq_j does not intersect any trapezoid of $\bar{\mathbf{\Delta}}$. This equivalence relation partitions E_{12} into equivalent classes, each consisting of a contiguous subsequence of E_{12}. Let μ_{ij} denote the number of equivalence classes in E_{ij}.

Claim 1: *There are at most two edges in each equivalence class of E_{12}.*

Proof: Assume for the sake of a contradiction that three edges $p_{i-1}q_{i-1},\ p_iq_i$ and $p_{i+1}q_{i+1}$ belong to an equivalence class. Further, assume that the triangle $\bar{T} \in \bar{\mathcal{T}}$ bounded by p_iq_i lies below p_iq_i (see Figure 6 (i)). Since the quadrilateral Q defined by p_iq_i and $q_{i+1}p_{i+1}$ does not contain any trapezoid of $\bar{\mathbf{\Delta}}$, $p_{i+1}q_{i+1}$ is also an edge of $\bar{T}$ and $Q \subseteq \bar{T}$. But then Q is a connected component of R and it touches only two edges of $\bar{\mathbf{\Delta}}$, thereby implying that p_iq_i is not an edge of a polygon of R_1, a contradiction. □

Thus,

$$\sum_{(i,j)\in E} m_{ij} \;\leq\; 2\sum_{(i,j)\in E} \mu_{ij} \;=\; 2|E| + \sum_{(i,j)\in E} (\mu_{ij} - 1) \;=\; O(k \log k) + \sum_{(i,j)\in E} (\mu_{ij} - 1)\,.$$

Next, we bound the quantity $\sum_{(i,j)\in E}(\mu_{ij} - 1)$. Let E_{12}^j and E_{12}^{j+1} be two consecutive equivalent classes of E_{12}, let p_iq_i be the bottom edge of E_{12}^j, and let $p_{i+1}q_{i+1}$ be the top edge of E_{12}^{j+1}. The quadrilateral $Q_{12}^j = p_iq_iq_{i+1}p_{i+1}$ contains at least one trapezoid $\bar{\Delta}$ of $\bar{\mathbf{\Delta}}$. We call $p_iq_i, p_{i+1}q_{i+1}$ the *triangle edges* of Q, and p_ip_{i+1}, q_iq_{i+1} the *trapezoidal edges* of Q. The triangle edges of Q_{12}^j are adjacent in E_{12}. Let $\mathcal{Q} = \bigcup\{Q_{ij}^l \mid (i, j) \leq E \text{ and } l < \mu_{ij}\}$ be the set of resulting quadrilaterals. Since $|\mathcal{Q}| = \sum_{(i,j)\in E}(\mu_{ij} - 1)$, it suffices to bound the number of quadrilaterals in $\mathcal{Q}$.

Consider the planar subdivision induced by $\mathcal{Q}$ and call it $\mathcal{A}(\mathcal{Q})$. For each bounded face $f \in \mathcal{A}(\mathcal{Q})$, let $Q(f)$ be the smallest quadrilateral of $\mathcal{Q}$ that contains f. Since the boundaries of quadrilaterals do not cross, $Q(f)$ is well defined.

Claim 2: *Every face f of $\mathcal{A}(\mathcal{Q})$ can be uniquely associated with a trapezoid $\Delta \in \bar{\mathbf{\Delta}}$ such*

[1] The segments of Γ may overlap, because the trapezoids of $\bar{\mathbf{\Delta}}$ can touch each other. If a segment γ_i of Γ is an edge of two trapezoids, then no edge of R_1 can be incident to γ_i.

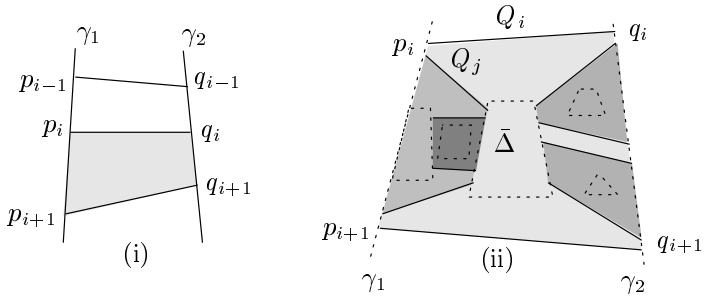

Figure 6: (i) Edges in an equivalent class of E_{12}, and (ii) edges in different equivalent classes

that $\Delta \subseteq f$.

Proof: The claim is obviously true for the unbounded face, so assume that f is a bounded face. If $Q_i = Q(f)$ does not contain any other quadrilateral of $\mathcal{Q}$, then $f = Q_i$, and so by definition f contains a trapezoid of $\bar{\mathbf{\Delta}}$. If $Q_i = Q(f)$ does contain another quadrilateral of $\mathcal{Q}$, let Q_j be the largest trapezoid that lies inside Q_i—that is, ∂Q_j is a part of ∂f. If none of the trapezoidal edges of Q_j lies in the interior of Q_i, then the trapezoidal edges of both Q_i, Q_j lie on the same segments of Γ, say, γ_1, γ_2. Consequently, the triangle edges of both Q_i, Q_j belong to E_{12}, which is impossible because then the triangle edges of Q_i are not adjacent in E_{12}. Hence, one of the trapezoidal edges of Q_j lies in the interior of Q_i. Let $\bar{\Delta}$ be the trapezoid bounded by this edge. Since the triangle edges of Q_j lie outside $\bar{\Delta}$ and the interior of $\bar{\Delta}$ does not intersect any edge of R_1, $\bar{\Delta}$ lies in f. This completes the proof of Claim 2. □

By Claim 2, the number of faces in $\mathcal{A}(\mathcal{Q})$ is at most $|\bar{\mathbf{\Delta}}| = O(k \log k)$. This completes the proof of the lemma. □

Next, we partition the polygons of R_2 into equivalence classes in the same way as we partitioned the edges of E_{12} in the proof of Lemma 7.1. That is, we call two polygons $\tau_1, \tau_2 \in R_2$ *equivalent* if (i) their endpoints lie on the same pair of edges in $\bar{\mathbf{\Delta}}$, and (ii) the interior of the convex hull of $\tau_1 \cup \tau_2$ does not intersect any trapezoid of $\bar{\mathbf{\Delta}}$. Using the same argument as in the proof of the above lemma, the following lemma can be established.

Lemma 7.2 *The edges of R_2 can be partitioned into $O(k \log k)$ equivalence classes.*

For each equivalence class $E_i \subseteq R_2$, let Q_i be the convex hull of E_i—observe that Q_i is a convex quadrilateral, as illustrated in Figure 5 (iii). Each quadrilateral Q_i can be ε-approximated using at most three triangles in $\mathbb{R}^3$ in the same way as we approximated each

polygon P_i of R_1. By Lemma 7.2, the total number of triangles created in this step is also $O(k \log k)$.

Putting together these pieces, we obtain the following lemma.

Lemma 7.3 *The family of trapezoids $\mathbf{\Delta}$ can be supplemented with $O(k \log k)$ additional trapezoids in $\mathbb{R}^3$ so that all the triangles of $\mathcal{T}$ are ε-approximated. The orthogonal projection of all the trapezoids on the plane $z = 0$ is pairwise disjoint.*

The area not covered by the projection of trapezoids found in the preceding lemma, of course, can be approximated without any regards to the triangles of $\mathcal{T}$. The final surface has $O(K_o \log K_o)$ trapezoids and it ε-approximates the family of triangles $\mathcal{T}$. We finish with a statement of our main theorem in this section.

Theorem 7.4 *Given a set of n horizontal triangles in $\mathbb{R}^3$, with pairwise disjoint projection on the plane $z = 0$, and a parameter $\varepsilon > 0$, we can compute in polynomial time a ε-approximate polyhedral terrain of size $O(K_o \log K_o)$ for $\mathcal{T}$, where K_o is the size of an optimal ε-approximate terrain.*

8 Closing Remarks

We presented an approximation technique for certain geometric covering problems with a disjointness constraint. Our algorithm achieves a logarithmic performance guarantee on the size of the cover, thus matching the bound achieved by the "greedy set cover" heuristic for arbitrary sets and no disjointness constraint. Applications of our result include polynomial time algorithms to approximate a monotone, polyhedral surface in 3-space, and to approximate the disjoint cover by triangles of red-blue points. We also proved that these problems are *NP-Hard*.

The surface-approximation problem is an important problem in visualization and computer graphics. The state of theoretical knowledge on this problem appears to be rather slim. Except for the convex surfaces, no approximation algorithms with good performance guarantees are known [7, 17]. For the approximation of convex polytopes, it turns out that one does not need *disjoint* covering, and therefore the greedy set cover heuristic works.

We conclude by mentioning some open problems. An obvious open problem is to reduce the running time of our algorithm for it to be of any practical value. Finding efficient heuristics with good performance guarantees seems hard for most of the geometric partitioning problems, and requires further work. A second problem of great practical interest is to ε-approximate general polyhedra—this problem arises in many real applications of computer modeling. To the best of our knowledge, the latter problem remains open even

for the special case where one wants to find a minimum-vertex polyhedral surface that lies between two monotone surfaces. The extension of our algorithm presented in Section 7 does not work because we do not know how to handle the last fill-in stage.

Acknowledgments

The authors thank Joe Mitchell, Prabhakar Raghavan, and John Reif for several helpful discussions.

References

[1] E. Allgower and S. Gnutzmann, An algorithm for piecewise linear approximation of an implicitly defined two-dimensional surfaces, *SIAM J. Numerical Analysis* 24 (1987), 452–469.

[2] E. Allgower and P. Schmidt, An algorithm for piecewise linear approximation of an implicitly defined manifold, *SIAM J. Numerical Analysis* 22 (1985), 322–346.

[3] E. Arkin, H. Meijer, J. Mitchell, D. Rappaport, S. Skiena, Decision trees for geometric models, *Proceedings 9th Annual Symposium on Computational Geometry*, 1993, 369–378.

[4] J. Bloomenthal, Polygonization of implicit surfaces, *Computer Aided Design* 5 (1988), 341–355.

[5] H. Brönnimann and M. T. Goodrich, Almost optimal set covers in finite VC-dimension, *Proc. 10th ACM Symp. on Computational Geometry*, 1994, pp. 293–302.

[6] B. Chazelle and M. Sharir and E. Welzl, Quasi-optimal upper bounds for simplex range searching and new zone theorems, *Algorithmica*, 8 (1992), 407–429.

[7] K. L. Clarkson, Algorithms for polytope covering and approximation, *Proc. 3rd Workshop on Algorithms and Data Structures*, Lectures Notes in Computer Science, vol. 709, Springer Verlag, pp. 246–252, 1993.

[8] M. DeHaemer and M. Zyda, Simplification of objects rendered by polygonal approximations, *Computers and Graphics* 15 (1992), 175–184.

[9] S. Fekete, Several hardness results on problems of point separation and line stabbing, Tech. Report, Department of Applied Mathematics and Statistics, SUNY Stony Brook, 1993.

[10] R. Fowler, M. Paterson, and L. Tanimoto, Optimal packing and covering in the plane are NP-complete, *Information Processing Letters* 35 (1981), 85–92.

[11] I. Ihm and B. Naylor, Piecewise linear approximations of digitized space curves with applications *Scientific Visualization of Physical Phenomena* pp. 545–569, Springer–Verlag, 1991.

[12] D. S. Johnson., Approximation algorithms for combinatorial problems, *Journal of Computer and System Sciences*, (1974), 256–278.

[13] D. Lichtenstein, Planar formulae and their uses, *SIAM J. Computing* 11 (1982), 329–343.

[14] W. Lorenson and H. Cline, Marching cubes: A high resolution 3D surface construction algorithm, *Computer Graphics* 21 (1987), 163–169.

[15] L. Lovász, On the ratio of optimal integral and fractional cover, *Discrete Mathematics* 13 (1975), 383–390.

[16] N. Megiddo, Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems, *SIAM J. Computing* 12 (1983), 759–776.

[17] J. Mitchell and S. Suri, Separation and approximation of polyhedral surfaces, *Proc. of 3rd ACM-SIAM Symposium on Discrete Algorithms* (1992), 296–306.

[18] M. S. Paterson and F. F. Yao, Efficient binary space partitions for hidden-surface removal and solid modeling, *Discrete and Computational Geometry*, 5 (1990), 485–503.

[19] M. S. Paterson and F. F. Yao, Optimal binary space partitions for orthogonal objects, *J. Algorithms* 13 (1992), 99–113.

[20] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, 1985.

[21] S. Salzberg, R. Chandar, H. Ford, S. Murthy, and R. White, A survey of decision tree classifier methodology, *IEEE Trans. Systems, Man and Cybernatics* 21 (1991), 660–674.

[22] W. Schroeder, J. Zarge, and W. Lorensen, Decimation of triangle meshes, *Computer Graphics* 26 (1992), 65–78.

[23] F. Schmitt, B. Barsky, and W. Hui Du, An adaptive subdivision method for surface fitting from sampled data, *Computer Graphics* 20 (1986), 179–188.

[24] S. Suri, On some link distance problems in a simple polygon, *IEEE Transactions on Robotics and Automation*, 6 (1) (1990), 108–113.

[25] G. Turk, Re-tiling of polygonal surfaces, *Computer Graphics* 26 (1992), 55–64.