SCHOOL OF OPERATIONS RESEARCH
AND INDUSTRIAL ENGINEERING
COLLEGE OF ENGINEERING
CORNELL UNIVERSITY
ITHACA, NEW YORK 14853

TECHNICAL REPORT NO. 948

December 1990

FINDING OPTIMAL BIPARTITIONS OF POINTS AND POLYGONS

By

Joseph S. B. Mitchell
and
Erik L. Wynters

Finding Optimal Bipartitions of Points and Polygons

Joseph S. B. Mitchell *
Erik L. Wynters †

November 28, 1990

Abstract

We give efficient algorithms to compute an optimal bipartition of a set of points or a set of simple polygons in the plane. We examine various criteria involving the perimeter and the area of the convex hulls of the two subsets.

1 Introduction

In the standard "bipartition problem", we are interested in partitioning a set S of n points into two subsets (S_1 and S_2) in such a way as to optimize some function of the "sizes" ($\mu(S_i)$) of the two subsets. Avis ([Av]) gave an $O(n^2 \log n)$ time algorithm to find a bipartition that minimizes the maximum of the diameters of the sets S_1 and S_2. Asano, Bhattacharya, Keil and Yao ([ABKY]) improved the bound on the time complexity of this problem, obtaining an optimal $O(n \log n)$ algorithm. Monma and Suri ([MS]) gave an $O(n^2)$ time algorithm for minimizing the sum of diameters. Most recently, Hershberger and Suri ([HS]) have considered the problem in which the measure of "size" $\mu(S_i)$ is (a). the diameter, (b). the area, perimeter, or diagonal

*SORIE, Cornell University, Ithaca, NY 14853. Partially supported by NSF grants IRI-8710858 and ECSE-8857642, and by a grant from Hughes Research Laboratories.

†Center for Applied Mathematics, Cornell University, Ithaca, NY 14853. Supported by a grant from Hughes Research Laboratories.

of the smallest enclosing axes-parallel rectangle, or (c). the side length of the smallest enclosing axes-parallel square. They provide $O(n \log n)$ time algorithms to find a bipartition that satisfies $\mu(S_i) \leq \mu_i$ $(i = 1, 2)$ for two given numbers μ_1 and μ_2.

Here, we consider the version of the bipartition problem in which $\mu(S_i)$ is the perimeter or area of the convex hull, $conv(S_i)$, and we desire a partition that minimizes the sum $\mu(S_1) + \mu(S_2)$ or the maximum $\max\{\mu(S_1), \mu(S_2)\}$. We also consider the generalization in which S is a set of disjoint polygons.

Funnel trees and hourglasses are fundamental to several of our algorithms, so we discuss their properties next.

2 Funnel Trees and Hourglasses

We use the term *vertex* to mean either a point in S or a polygon vertex. Let n denote the total number of vertices.

The visibility graph of S is the graph whose vertices are the vertices contained in or determined by S and whose edges are pairs of vertices (u, v) such that the open line segment between u and v does not intersect any polygon in S. The visibility graph can be computed in time $O(E + n \log n)$ and space $O(E)$ where E is the number of edges in the graph. Note that E is $O(n^2)$ when S is a set of points, but can be smaller when S is a set of polygons.

A *visible chain* is a path in the visibility graph. It is well known that a shortest path between two points in the plane that avoids polygonal obstacles is a visible chain [Le, LP, LW, Mi, SS]; the subgraph of the visibility graph consisting of all visible chains that are shortest paths from a fixed source vertex s to some other vertex is called the *shortest path tree rooted at* s.

Consider a vertex a that lies to the left of a directed edge (u, v) and is visible from a point b in the interior of the edge. Define the *lower chain* of a determined by u, v, and b to be the unique convex visibile chain from a to u for which the open region bounded by the chain and the line segments $\overline{uv}$ and $\overline{ab}$ contains no vertices. Intuitively, think of taking a rubber band between a and b in Figure 1 and sliding the end at b along $\overline{uv}$ until it reaches u. Define the *upper chain* from a to v analagously. These two chains form a *funnel* with *apex* a and *base* (u, v).

We require the base of a funnel to be an edge on the convex hull of the

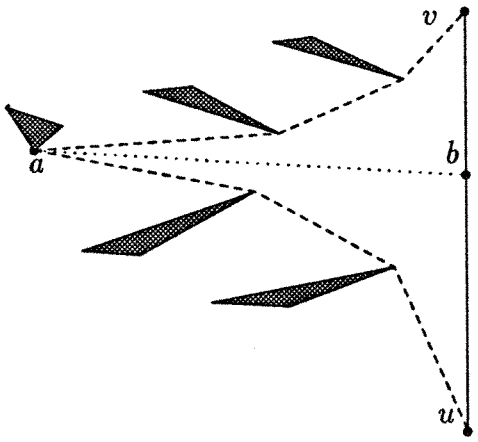

Figure 1: A funnel with base (u, v) and apex a.

set S of points or polygons we wish to partition. In this setting a funnel is completely determined by its apex and by the next vertex on its lower (or upper) chain, i.e., by the first directed edge on its lower (or upper) chain [GM]. Therefore, the total number of funnels in the visibility graph is at most $2E$. Furthermore, we can find all of the directed edges that determine funnels, in time proportional to their number, by traversing part of the visibility graph as explained below.

Consider a funnel f with apex a and base (u, v), and suppose b is the first vertex after a on its lower chain. There is a unique funnel p with apex b and base (u, v) contained in funnel f; its left chain is obtained from that of f by deleting edge (a, b). If we think of p as the parent of f, we see that the funnels with base (u, v) form a tree in the visibility graph (taking the location of a funnel to be that of its apex) rooted at u. Call this the *lower funnel tree* on (u, v). Note that the path from the root of this tree to any node is precisely the lower chain of the funnel corresponding to that node. In an analogous way, the upper chains of funnels with base (u, v) determine an *upper funnel tree* on (u, v) rooted at v. Figure 2 shows a lower funnel tree, and an upper funnel tree on the same base.

Using the representation of the visibility graph generated by their algorithm, Ghosh and Mount show that parents, clockwise and counterclockwise siblings, and extreme clockwise and counterclockwise children of a node in a funnel tree can be found in constant time [GM]. This implies that clockwise

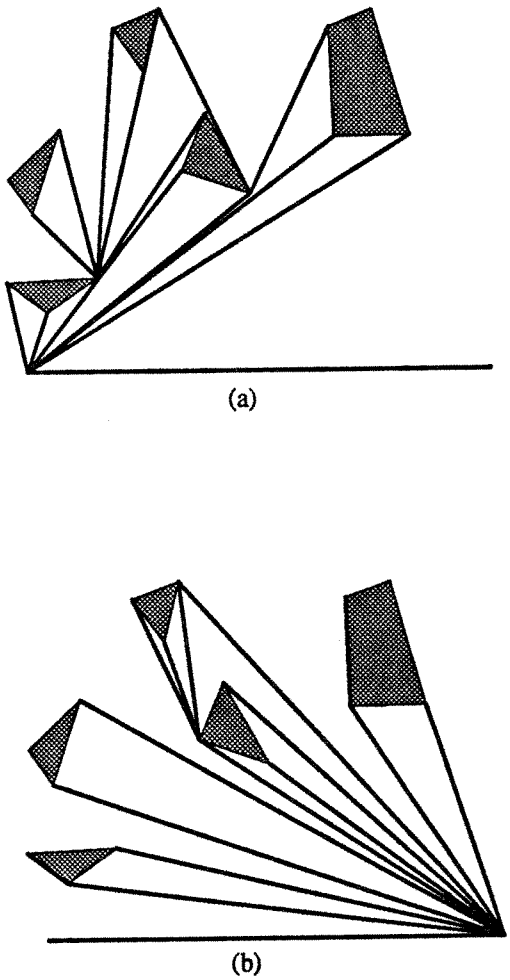

Figure 2: A lower funnel tree (a) and an upper funnel tree (b).

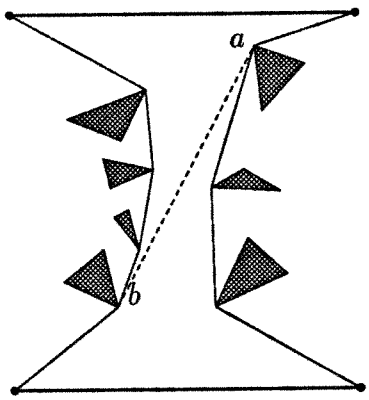

Figure 3: An hourglass formed from two funnels.

and counterclockwise traversals of lower and upper funnel trees can be made in the visibility graph without storing these trees explicitly. Also, we see that, summing over all bases, the total time needed to traverse these trees is $O(E)$.

If f_1 is a funnel determined by directed edge (a, b) and f_2 is a funnel determined by its reversal (b, a), then f_1 and f_2 together form an *hourglass* as shown in Figure 3. Since it takes two funnels to form an hourglass and each funnel is part of at most one hourglass, there are at most E hourglasses contained in the visibility graph. We find hourglasses by traversing the lower (or upper) funnel trees a second time and checking whether the reversal of the directed edge that determines the current funnel also determines a funnel.

Suppose that some function of an hourglass can be computed in constant time if the result of applying the function to the two funnels that form it is known. For example, the length of one of an hourglass's convex chains or the area of the region enclosed by an hourglass is easily computed from the chain lengths or areas of the funnels that form it. If the function is first applied to funnels during the intitial funnel tree traversals, then the function can be applied to all hourglasses during a second iteration of funnel tree traversals in $O(E)$ time. Furthermore, if the function can be evaluated on funnels in amortized constant time, the whole process can be done in $O(E)$ time.

We show below that the lengths of a funnel's upper and lower chains and the area of the region enclosed by a funnel can be calculated in amortized constant time and space during funnel tree traversals. We calculate the

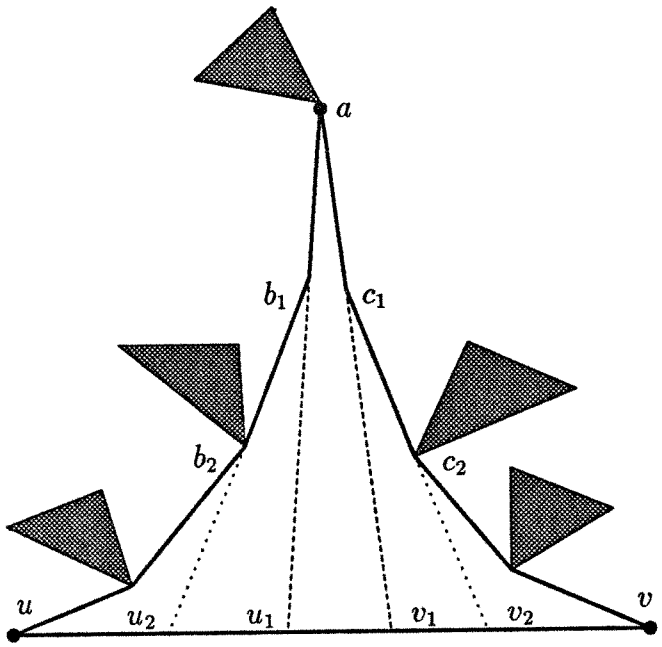

Figure 4: Partitioning the region enclosed by a funnel.

length of a funnel's lower (or upper) chain in constant time during a preorder traversal of the lower (or upper) funnel tree by simply adding the length of its first edge to the length of the lower (or upper) chain of its parent.

Consider the funnel shown in Figure 4. We compute the area of the region enclosed by the funnel by partitioning it into three smaller regions. The central region is a triangle, whose area is easy to calculate. We focus on the region between the lower chain and the segment $\overline{b_1u_1}$. Call the area of this region the *lower area* of the funnel. The lower area of the funnel is just the lower area of its parent funnel in the lower funnel tree plus the area of triangle $\triangle b_1u_1u_2$. The upper area of the funnel is calculated analogously.

Our data structure for directed visibility graph edges allows us to store with each instance of an edge, the result of applying these length or area functions to the funnel or hourglass it determines. Thus the total amount of space needed to store the results of all of the function evaluations is $O(E)$. In the algorithms that follow we also maintain a variable corresponding to the current "best" hourglass, i.e., the one that minimizes the value of the applied function. After completing the funnel tree traversals this variable

holds a representation of the optimal hourglass according to one of several criteria.

In summary, we use funnel tree traversals to find optimal hourglasses in $O(E)$ time and $O(E)$ space.

3 Hourglasses, Funnelglasses, and Bipartitions

Now we show that optimal bipartitions of points correspond to hourglasses and that optimal bipartitions of polygons correspond to hourglasses or "funnelglasses".

Start with an hourglass in the visibility graph of a set of points S and consider the two convex polygons formed by the set of edges obtained as follows: start with the edges of $conv(S)$ and delete the base edges of the hourglass and insert the edges of its two convex chains. Since the two chains (and hence the two polygons) are disjoint and the interior of the hourglass is empty, the polygons are the convex hulls of S_1 and S_2 for some bipartition $S{=}S_1 \cup S_2$ of S.

Now consider a bipartition $S = S_1 \cup S_2$ of a set of points S. Note that it is always beneficial to have both S_1 and S_2 nonempty in the bipartition since letting S_1 be a single vertex of the convex hull $conv(S)$ implies $\mu(S_1) = 0$ and $\mu(S_2) < \mu(S)$. This also implies that we never want $conv(S_1) \subset conv(S_2)$ or vice versa. And it's easy to see that any bipartition in which the convex hulls overlap cannot be optimal because the points in the intersection can be swapped from one set of the bipartition into the other in such a way that a new bipartition is created that reduces the perimeter and the area of each convex hull.

Suppose that we have a bipartition of S satisfying $conv(S_1) \cap conv(S_2) = \emptyset$. Then an inner common tangent to these convex polygons will be a line lying along a visibility graph edge (u, v), and each of the (closed) half-spaces determined by this line will contain one of the sets (S_i). Since the hourglass determined by (u, v) is empty and each of the half-spaces above contains one of its convex chains, it will determine two polygons, as mentioned above, that are precisely $conv(S_1)$ and $conv(S_2)$.

We have shown that in any candidate $S = S_1 \cup S_2$ for an optimal bi-

partition of a point set S, S_1 and S_2 must have disjoint convex hulls. And any bipartition for which the convex hulls are disjoint can be obtained from an hourglass. Furthermore, every hourglass determines such a partition. Therefore, the problem of finding an optimal bipartition of a set of points is equivalent (via a linear-time reduction) to that of finding an optimal hourglass.

We also discuss the problem of finding an optimal bipartition of a set of polygons. Intuitively, when we were bipartitioning point sets, we were using two disjoint closed fences to contain the points, and we were minimizing the perimeters or areas of the fenced-in regions. We generalize this idea to the polygon case by considering only those bipartitions $S = S_1 \cup S_2$ of a set of polygons S that have the property that the shortest closed fence containing the obstacles in S_1 and the shortest closed fence containing the obstacles in S_2 form simple polygons with disjoint interiors. Recall that when partitioning points, if a point p of S_2 was contained in the convex hull of S_1 (or vice versa), it was better to delete p from S_2 and insert it in S_1. This led us to conclude that the fences needed to be disjoint. With polygons, however, a polygon P belonging to S_2 may be partially contained in the convex hull of S_1 in such a way that changing its membership increases the lengths of the fences. This means that optimal fences need not be convex – they may bulge in around polygons not contained in the fence – and they need not be disjoint – they may coincide along some portion of their length.

Now we define the polygon partitioning problem more formally. We say that a bipartition $S = S_1 \cup S_2$ of a set of polygons S is *valid* if S_1 and S_2 can be separated by a simple path between distinct edges of the convex hull of S (note that when partitioning points, valid partitions were separable by a single line). Given a valid bipartition, the *relative convex hull* $rconv(S_1)$ with respect to S_2 is the minimum perimeter polygon that contains every polygon in S_1 but contains no interior point of a polygon in S_2. Figure 5 shows a valid bipartition and Figure 6 shows the resulting relative convex hulls.

If the relative convex hulls $rconv(S_1)$ and $rconv(S_2)$ of a valid bipartition are disjoint, then they are also convex and can be obtained from an hourglass as above. In general, however, they will share a sequence of edges, i.e., they will coincide along a visible chain. In this case, they form what we call a funnelglass. A *funnelglass* is a pair of funnels and a shortest path between their apices that together satisfy the following conditions: the funnels are

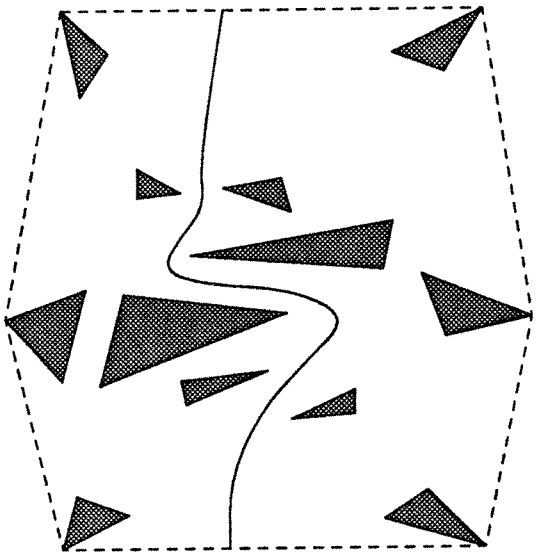

Figure 5: A simple path determines a valid bipartition of a set of simple polygons.

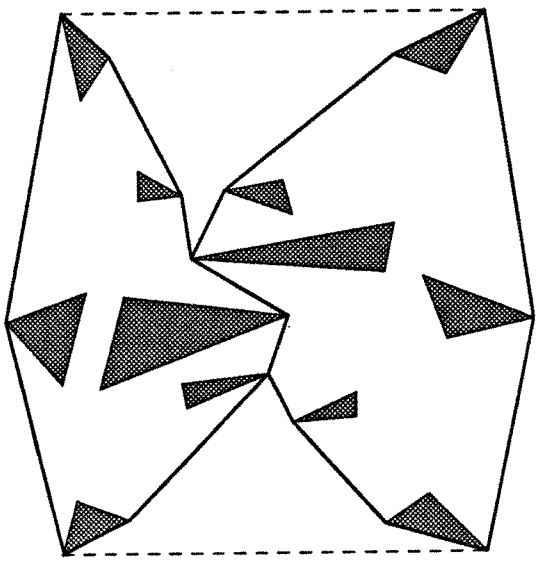

Figure 6: The relative convex hulls corresponding to the bipartition of Figure 5 and their corresponding funnelglass.

disjoint or they intersect only in having a common apex, and the extension of each funnel chain along the inter-apex path is a locally shortest or "taut string" path. The funnel chains and bases and the shortest path form a *weakly simple* polygon (one formed by a path with repeated vertices or edges but no self-crossings) as shown in Figure 6.

The correspondence between funnelglasses and bipartitions with "touching" relative convex hulls is as follows. Start with a funnelglass in the visibility graph of a set of polygons S and consider the two "touching" polygons formed by the set of edges obtained as follows: take the edges of $conv(S)$ and delete the base edges of the funnelglass and insert the edges of its convex chains and its inter-apex path. Since the interior of the funnelglass is empty and since the inter-apex path can be extended through the interior of each funnel to a point on the funnel's base, the funnelglass separates the set of polygons and thus corresponds to a valid bipartition $S = A_1 \cup S_2$ for some S_1 and S_2. Furthermore, the "taut-string" property ensures that the polygons formed from it have minimal perimeter, i.e., they are precisely $rconv(S_1)$ and $rconv(S_2)$.

Conversely, if we start with the optimal valid bipartition $S = S_1 \cup S_2$, and $rconv(S_1)$ and $rconv(S_2)$ coincide along a visible chain, then the edge set $E = conv(S) \triangle rconv(S_1) \cup rconv(S_2)$ forms a funnelglass as follows. The intersection $rconv(S_1) \cap rconv(S_2)$ must be a shortest path between two vertices u and v by optimality of the bipartition, and also the "taut-string" property must hold. And where the edges diverge at u and v they form funnels since relative convex hulls are convex where they don't "touch".

We have shown that an hourglass or a funnelglass determines a valid bipartition of a set of polygons. Furthermore, the optimal bipartition can always be obtained from an hourglass r a funnelglass. Therefore, the problem of finding an optimal bipartition is equivalent (via a linear-time reduction) to that of finding an optimal hourglass or funnelglass.

4 Partitioning Points

Now we consider the case in which S is a set of points in the plane, and the "sizes" $\mu(S_i)$ of the subsets are taken to be the perimeters or areas of their convex hulls. We seek a bipartition that minimizes the sum $\mu(S_1) + \mu(S_2)$ or the maximum $\max\{\mu(S_1), \mu(S_2)\}$. This gives us the following set of four

problems:

1. the min-sum perimeter problem,
2. the min-max perimeter problem,
3. the min-sum area problem, and
4. the min-max area problem.

Note that it is always beneficial to have both S_1 and S_2 nonempty in the bipartition since letting S_1 be a single vertex of the convex hull $conv(S)$ implies $\mu(S_1) = 0$ and $\mu(S_2) < \mu(S)$. We will show later that this is not always the case when partitioning polygons.

Our results about partitioning point sets are summarized in the following theorem.

Theorem 1 *Given a set S of n points in the plane, a bipartition of S, $S = S_1 \cup S_2$, that solves any of problems 1 through 4 above can be found in time $O(n^3)$ using $O(n)$ space. Alternatively, problems 1 through 3 above can be solved in $O(n^2)$ time using $O(n^2)$ space.*

Proof: The algorithm for the first claim is based on the observation that there are only $O(n^2)$ distinct ways to partition n points, after which the convex hulls $conv(S_i)$ can be examined in a straightforward manner:

(0). Sort the points S by x-coordinate. Compute the convex hull of S.

(1). Consider each pair of points $p, q \in S$ $(p \neq q)$ in turn. Each pair (p, q) such that $\overline{pq}$ does not lie on the boundary of the convex hull of S defines a line that partitions S into two nonempty subsets: Use the line $L_{p,q}$ obtained by taking the line through p and q and rotating it clockwise by an infinitesimal amount about the midpoint of the segment $\overline{pq}$.

   (a). For a given pair (p, q), we march through the list of points S (in x order), marking each point as being on the left or the right side of the oriented line through $L_{p,q}$. This yields each of the two sets S_1 and S_2 (for this pair (p, q)) in sorted x order.

   (b). In time $O(n)$, we can then find the convex hulls of S_1 and S_2.

(c). Once we have the convex hulls of each set S_1 and S_2, we can easily compute the sum $\mu(S_1)+\mu(S_2)$ or the maximum $\max\{\mu(S_1), \mu(S_2)\}$ in linear time, where $\mu(S_i)$ denotes either perimeter or area (or any other function on a convex polygon that can be computed in linear time).

By keeping track of the best partitioning so far, we will have an optimal partitioning of S by the time we have examined all pairs (p, q) (since an optimal partitioning must correspond to *some* pair (p, q)).

The above algorithm clearly requires linear time per pair (p, q), so $O(n^3)$ time overall. The space requirement is only linear.

To prove the second claim of the theorem, we rely on methods developed in Section 2. Each pair of points (p, q) determines an hourglass whose bases are edges on the convex hull of S. The two convex chains of the hourglass together with the edges remaining in the convex hull after deleting the two base edges form two convex polygons that correspond to some bipartition of S. Conversely, every bipartition of S corresponds to an hourglass.

The "cost" of a bipartition is easily calculated from the chain lengths of its hourglass in constant time when minimizing either the total or maximum perimeter of $conv(S_1)$ and $conv(S_2)$. Similarly, the total area of the two convex hulls arising from a bipartition is just the area of the entire convex hull minus the area of the hourglass corresponding to the partition. By generating all hourglasses and comparing each newly considered hourglass to the best hourglass seen so far, we obtain the optimal hourglass and hence the optimal bipartition for each of problems 1, 2, and 3 above.

We compute the cost of all $O(n^2)$ hourglasses in time $O(n^2)$, as in Section 2. The space required by this algorithm is quadratic, since we use the (complete) visibility graph of the point set S. ∎

Remark. It is an interesting open question to determine if one can reduce the space complexity of the above algorithm to linear while maintaining the quadratic time bound (e.g., using topological sweep [EG]).

5 Partitioning Polygons

Now we consider the case in which S is a set of (pairwise-disjoint) simple polygons with a total of n vertices. The "size" of S_1 (S_2) will now be the

perimeter of the *relative* convex hull, $rconv(S_1)$ ($rconv(S_2)$), of S_1 (S_2) with respect to the set S_2 (S_1). Again we seek a bipartition that minimizes the sum $\mu(S_1) + \mu(S_2)$ or the maximum $\max\{\mu(S_1), \mu(S_2)\}$, giving us once more the following set of four problems:

1. the min-sum perimeter problem,
2. the min-max perimeter problem,
3. the min-sum area problem, and
4. the min-max area problem.

Note that, in contrast to the case in which S is a set of points, it is *not* always beneficial to have both S_1 and S_2 nonempty here in problems 1 and 2: it may be that every nontrivial bipartition of S results in the perimeters of $rconv(S_1)$ and $rconv(S_2)$ each being greater than the perimeter of $conv(S)$. Therefore, for these problems we distinguish between the case in which we want to find only beneficial bipartitions and the case in which we are forced to find nontrivial bipartitions even if every one of them is non-beneficial. This distinction is not necessary in problems 3 and 4 because the best nontrivial bipartition for these problems will always be beneficial.

Our first polygon partitioning result is contained in the following theorem:

Theorem 2 *Given a set S of (pairwise-disjoint) simple polygons with a total of n vertices, one can find a* beneficial *bipartition of S, $S = S_1 \cup S_2$, that solves the min-sum perimeter problem (or report that none exists) in time $O(En)$, using $O(n)$ space. Alternatively, one can solve this problem in time $O(E + n \log n)$ using $O(E)$ space.*

Proof: The algorithm for the first claim is straightforward:

(0). Triangulate the polygon with holes formed by the convex hull of S and the polygons in S.

(1). Generate the edges of the visibility graph using only $O(n)$ working space and $O(E \log n)$ time [OW]. As each edge (u, v) is found, do the following calculations:

(a). Extend edge (u, v) at both ends until a polygon or the convex hull of S is hit.

(b). If the convex hull was hit on each side, i.e., (u, v) determines an hourglass, then compute the lengths of the hourglass's chains by finding the two funnels that form the hourglass as follows. The funnel with apex u is found by propagating the extension of (u, v) through adjacent cells of the triangulation until the triangle containing the convex hull edge hit by the extension is reached. The shortest paths from u to the endpoints of the convex hull edge that lie within the simple polygon formed from the pierced triangles is found in $O(n)$ time, and these paths are the upper and lower chains of the funnel [LP]. The funnel with apex v is found similarly. Calculate the "cost" of this hourglass, compare it to the current best one, and update if neccessary by storing the edge (u, v) and its cost.

The above algorithm requires linear time per edge (u, v), so $O(En)$ time overall. The space requirement is only linear.

To prove the second claim of the theorem, we again compute hourglasses, but we compute the entire visibility graph first in time $O(E + n \log n)$ using $O(E)$ space [GM]. Then we traverse funnel trees as in Section 2 and find the best hourglass in $O(E)$ time.

∎

Remark. It is interesting to note that partitioning polygons is potentially faster than partitioning points (when E is sparse).

To handle the min-max perimeter problem, it is no longer sufficient to consider only bipartitions corresponding to hourglasses; we must also consider funnelglasses. A *funnelglass* is a pair of funnels connected by a shortest path between the two apices of the funnels. Funnelglasses correspond to partitions in which the two relative convex hulls coincide along some visible chain.

For each funnel in a funnelglass, the total length of the two chains of the funnel must be greater than the length of its base; that's why we didn't need to consider funnelglasses when we were only interested in beneficial solutions to the min-sum perimeter problem. But for the min-max perimeter problem, a funnelglass may correspond to a beneficial partition, so it's no easier to

find beneficial partitions than it is to find forced ones. We find beneficial partitions by finding the best nontrivial partition first. If this funnelglass has a lower cost then that of the entire convex hull, it is the optimal beneficial solution; otherwise, no beneficial solution exists.

Our results for the min-max perimeter problem appear in the following theorem.

Theorem 3 *Given a set S of (pairwise-disjoint) simple polygons with a total of n vertices, one can find a* nontrivial *bipartition of S, $S = S_1 \cup S_2$, that solves the min-max perimeter problem in time $O(E^2 n)$, using $O(n)$ space. Alternatively, one can solve this problem in time $O(E^2 + n^2 \log n)$ time, using $O(E)$ space.*

Proof: The algorithm for the first claim appears below.

Generate the edges of the visibility graph using only $O(n)$ working space and $O(E \log n)$ time [OW]. As each edge (u, v) is found, do the following:

(0). Extend edge (u, v) from u through v until it hits a polygon or an edge of the convex hull of S.

(1). If an edge of the convex hull was hit, i.e., if (u, v) determines a funnel, then

  (a). Determine the chain lengths of the funnel determined by (u, v) in linear time and space as in the proof of Theorem 2.

  (b). Build a shortest path map rooted at u in $O(n^2)$ time and $O(n)$ space [RS].

  (c). Generate the edges of the visibility graph using only $O(n)$ working space and $O(E \log n)$ time [OW] while holding edge (u, v) from the outer loop fixed. As each edge (u', v') is found, if this edge determines a funnel, determine the chain lengths of the funnel, and calculate the cost of the funnelglass formed by this funnel, the funnel determined by (u, v), and the path from u' to u in the shortest path map. Compare the current funnelglass to the stored representation of the best funnelglass found so far and update if necessary.

The time needed by the above algorithm is dominated by the time needed to find all funnelglasses using one fixed funnel. This is an $O(En)$ time computation that needs to be done $O(E)$ times. The space requirement is only linear.

To prove the second claim of the theorem, we again find funnelglasses, but we compute the entire visibility graph first in time $O(E + n \log n)$ using $O(E)$ space [GM]. Then we find all $O(E)$ funnels by traversing funnel trees as in Section 2. While doing this we create a list of all the funnels and a list at each vertex of the funnels having that vertex as their apex. Then for each vertex u we do the following:

(0). Find the shortest path tree rooted at u using Dijkstra's algorithm ($O(E + n \log n)$).

(1). For each vertex v and for each pair of funnels, one with apex u and one with apex v, compute the cost of the funnelglass formed by this pair of funnels and the shortest path from u to v. Update the representation of the best funnelglass found so far if necessary.

This algorithm requires time $O(En + n^2 \log n)$ for the n runs of Dijkstra's algorithm and $O(E^2)$ time to examine all pairs of funnels; thus the total running time is $O(E^2 + n^2 \log n)$ and the total space required for the visibility graph and funnel representations is $O(E)$. ∎

When we are forced to consider non-beneficial partitions in the min-sum perimeter problem, we have to look for funnelglasses as in the min-max case. But there is a difference: in the min-max case, any of the $O(n)$ funnels on each vertex could be part of the optimal funnelglass, but in the min-sum case there is a unique "best" funnel on each apex (taking the cost of a funnel to be the sum of its chain lengths minus the length of its base). This lets us find an optimal funnelglass more efficiently.

Our results for this problem appear in the following theorem.

Theorem 4 *Given a set S of (pairwise-disjoint) simple polygons with a total of n vertices, one can find a* nontrivial *bipartition of S, $S = S_1 \cup S_2$, that solves the min-sum perimeter problem in time $O(n^3)$, using $O(n)$ space. Alternatively, one can solve this problem in time $O(En + n^2 \log n)$ time, using $O(E)$ space.*

Proof: The algorithm for the first claim is as follows:

(0). Find the best hourglass (if one exists) as in the proof of Theorem 2.

(1). Find all funnels in $O(En)$ time and $O(n)$ space as in the proof of Theorem 2 maintaining a representation of the best funnel found at each vertex together with its cost.

(2). Now for each vertex u build the shortest path map rooted at u and as each vertex v is labeled with its distance from u, calculate the cost of the funnelglass formed from the best funnel on u and the best funnel on v in constant time, updating a representation of the best funnelglass found so far if necessary.

(3). Output the bipartition corresponding to the hourglass or funnelglass with minimum cost.

The running time is dominated by the n shortest path map constructions which take a total of $O(n^3)$ time. The total space requirement is linear.

The algorithm for the second claim appears below:

(0). Calculate the visibility graph in $O(E + n \log n)$ time and $O(E)$ space [GM].

(1). Generate all funnels and hourglasses using funnel tree traversals and store with each vertex of the visibility graph a representation of the best and the second best funnel having that vertex as apex. Also store a representation of the best hourglass found.

(2). For each edge e on the convex hull of S do the following:

  (a). Make a copy VG_e of the visibility graph.

  (b). Form an augmented visibility graph VG'_e as follows:

    - Create an additional "funnel" node fn for each funnel f that was determined to be one of the "top two" funnels on its apex.
    - Connect each funnel-node fn to the node v corresponding to the apex of funnel f. Let the length of edge (fn, v) be $c(f)/2$ where $c(f)$, the cost of a funnel is the sum of its chain lengths minus the length of its base.

- Create a "source" node s, linking it with an edge of length 0 to every funnel-node corresponding to a funnel with base e. Similarly, create and link a "sink" node t to every funnel-node not already linked to s.

(c). Find a shortest path from s to t in VG'_e. This determines the optimal funnelglass with one base on e.

(d). If necessary, update the stored representation of the best funnelglass found so far.

(3). Output the bipartition corresponding to the optimal funnelglass or hourglass.

Funnelglasses are also used to solve the min-sum area problem. In this problem, the length of the path connecting the two apices is irrelevant; the cost of the funnelglass is just the area of the entire convex hull minus the sum of the two funnel areas. But there is a difficulty here that doesn't arise in the other problems: the funnelglass of minimal cost may not correspond to a feasible bipartition because its funnels may overlap.

Our result for this case is stated in the following theorem.

Theorem 5 *Given a set S of (pairwise-disjoint) simple polygons with a total of n vertices, one can find a bipartition of S, $S = S_1 \cup S_2$, that solves the min-sum area problem in time $O(E^2n^2)$, using $O(n)$ space.*

The algorithm for this claim is based on the algorithm for the first claim in Theorem 2. We again generate all pairs of funnels and calculate the cost of each pair, but this time a funnel's cost (or benefit) is the area of the region it encloses. Since a funnel can be triangulated in linear time, we can calculate its area in linear time and check that the two funnels in a pair don't intersect in linear time [?]. Therefore this algorithm takes $O(E^2n)$ time and requires only linear space.

∎

Finally, we consider the min-max area problem. We show that this problem is NP-Hard by a polynomial-time reduction from the following NP-Complete problem [GJ]:

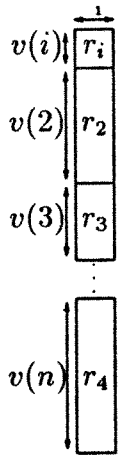

Figure 7: A "stack" of rectangles corresponding to a set of objects.

Partition: Given a set T of n objects with positive integer values $v(t)$, is there a partition $T = T_1 \cup T_2$ such that $\sum_{t \in T_1} v(t) = \sum_{t \in T_2} v(t)$.

Theorem 6 *The min-max area problem is NP-Hard.*

Proof: Given an instance of the partition problem, we generate in linear time a "stack" of rectangles as shown in Figure 7. Rectangle r_t corresponds to object $t \in T$ and has width 1 and length $v(t)$. The vertical separation between rectangles is infinitesimally small. Every bipartition $S = S_1 \cup S_2$ of this set S of rectangles can be separated by a simple path, and, for each subset in a bipartition, the area of its relative convex hull is just the total area of its rectangles. Therefore, the optimal solution of the min-max area problem would be a bipartition $S = S_1 \cup S_2$ satisfying

$$\sum_{r \in S_1} Area(r) = \sum_{r \in S_2} Area(r) = 1/2 \sum_{t \in T} v(t)$$

if and only if there exists a bipartition $T = T_1 \cup T_2$ of the objects satisfying $\sum_{t \in T_1} v(t) = \sum_{t \in T_2} v(t)$. And such a bipartition of rectangles can be transformed into the desired partition of objects in linear time. ∎

6 Conclusion

Our results are summarized in the following table.

	Points	Polygons	
		Beneficial Partitions	Forced Partitions
Min-Sum Perimeter	$O(n^2)$ / $O(n^2)$ or $O(n^3)$ / $O(n)$	$O(E + n \log n)$ / $O(E)$ or $O(En)$ / $O(n)$	$O(En + n^2 \log n)$ / $O(E)$ or $O(n^3)$ / $O(n)$
Min-Max Perimeter	$O(n^2)$ / $O(n^2)$ or $O(n^3)$ / $O(n)$	$O(E^2 + n^2 \log n)$ / $O(E)$ or $O(E^2 n)$ / $O(n)$	$O(E^2 + n^2 \log n)$ / $O(E)$ or $O(E^2 n)$ / $O(n)$
Min-Sum Area	$O(n^2)$ / $O(n^2)$ or $O(n^3)$ / $O(n)$	$O(E^2 n)$ / $O(n)$	$O(E^2 n)$ / $O(n)$
Min-Max Area	$O(n^3)$ / $O(n)$	NP-Hard	NP-Hard

Several questions about these problems remain open. Perhaps the running times given for these algorithms can be improved. For several of these problems our best algorithm uses only linear space. Is there a faster algorithm using a quadratic amount of space? Also it might be interesting to find approximate solutions to the NP-Hard min-max area problem for polygons.

Acknowledgement

We wish to thank Esther Arkin, Robert Freimer, and Christine Piatko for helpful discussions. This research was partially supported by a grant from the Hughes Research Laboratories, Malibu, CA and by NSF Grants IRI-8710858 and ECSE-8857642.

References

[AKM] E.M. Arkin, S. Khuller, and J.S.B. Mitchell, "Optimal Enclosure Problems", Technical Report TR 90-1111, Department of Computer Science, Cornell University, 1990. Presented at the First Canadian Conference on Computational Geometry, Montreal, Canada, August, 1989.

[ABKY] T. Asano, B. Bhattacharya, M. Keil, and F. Yao, "Clustering Algorithms Based on Minimum and Maximum Spanning Trees", *Proc. Fourth Annual ACM Symposium on Computational Geometry*, Urbana-Champaign, IL, June 6-8, 1988, pp. 252-257.

[Av] D. Avis, "Diameter Partitioning", *Discrete and Computational Geometry* **1** (3) (1986), pp. 265-276.

[EG] H. Edelsbrunner and L. Guibas, "Topologically Sweeping an Arrangement", *Journal of Computer and System Sciences* (1989), **38**, pp. 165-194.

[GJ] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, New York, 1979.

[GM] S.K. Ghosh and D.M. Mount, "An Output Sensitive Algorithm for Computing Visibility Graphs", *Proc. 28th Annual IEEE Symposium on Foundations of Computer Science*, pp. 11-19, 1987. To appear: *Algorithmica.*

[HS] J. Hershberger and S. Suri, "Finding Tailored Partitions", *Proc. Fifth Annual ACM Symposium on Computational Geometry*, Saarbrücken, West Germany, June 1989, pp. 255-265.

[Le] D.T. Lee, "Proximity and Reachability in the Plane", Ph.D. Thesis, Technical Report ACT-12, Coordinated Science Laboratory, University of Illinois, Nov. 1978.

[LP] D.T. Lee and F.P. Preparata, "Euclidean Shortest Paths in the Presence of Rectilinear Boundaries", *Networks*, **14** (1984), pp. 393-410.

[LW] T. Lozano-Pérez and M.A. Wesley, "An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles", *Communications of the ACM*, Vol. 22, No. 10 (1979), pp. 560-570.

[Mi] J.S.B. Mitchell, "A New Algorithm for Shortest Paths Among Obstacles in the Plane", Technical Report No. 832, School of Operations Research and Industrial Engineering, Cornell University, October, 1988.

[SS] M. Sharir and A. Schorr, "On Shortest Paths in Polyhedral Spaces", *SIAM Journal on Computing* Vol. 15, No. 1, pp. 193-215, February 1986.

[MS] C. Monma and S. Suri, "Partitioning Points and Graphs to Minimize the Maximum or the Sum of Diameters", In *Proceedings of the Sixth International Conference on the Theory and Applications of Graphs*, John Wiley & Sons, 1988.

[OW] M.H. Overmars and E. Welzl, "New Methods for Computing Visibility Graphs", *Proc. Fourth Annual ACM Symposium on Computational Geometry*, Urbana-Champaign, IL, June 6-8, 1988, pp. 164-171.

[RS] J. H. Reif and J. A. Storer, 'Shortest Paths in Euclidean Space with Polyhedral Obstacles', Technical Report CS-85-121, Computer Science Department, Brandeis University, April 1985.