Shortest Paths Help Solve Geometric Optimization Problems in Planar Regions *

Elefterios A. Melissaratos and Diane L. Souvaine
Department of Computer Science, Rutgers University, New Brunswick, NJ 08903

January 1, 1954

Abstract

The goal of this paper is to show that the concept of the shortest path inside a polygonal region contributes to the design of efficient algorithms for certain geometric optimization problems involving simple polygons: computing optimum separators, maximum area or perimeter inscribed triangles, a minimum area circumscribed concave quadrilateral, or a maximum area contained triangle. The structure for our algorithms is as follows: a) decompose the initial problem into a low-degree polynomial number of optimization problems; b) solve each individual subproblem in constant time using standard methods of calculus, basic methods of numerical analysis, or linear programming. These same optimization techniques can be applied to splinegons (curved polygons). To do this, we first develop a decomposition technique for curved polygons which we substitute for triangulation in creating equally efficient curved versions of the algorithms for the shortest-path tree, ray-shooting and two-point shortest path problems. The maximum-are or perimeter inscribed triangle problem, the minimum area circumscribed concave quadrilateral problem and maximum area contained triangle problem have applications to robotics and stock-cutting. The results of this paper will appear also in [33].

Key Words and Phrases: robotics, stock-cutting, computational geometry, enclosure problems, inclusion problems, separators, geometric optimization, shortest paths, visibility, simple polygons, splinegons.

1 Introduction

The linear-time algorithm for computing the lengths of the shortest paths inside a triangulated simple polygon from a designated start vertex [26] provides a useful tool in developing efficient polygon algorithms for a class of geometric optimization problems. Although our main results refer to the polygon case of the optimization problems, we extend our results to the curvilinear case also. Souvaine and Dobkin have recently argued that, wherever possible, new results should be presented for polygons and curved polygons simultaneously [18]. To make these extensions feasible we need to develop algorithms for shortest paths and visibility problems in curvilinear objects. Unfortunately, shortest paths and visibility represent an area in which little work has been done on curved polygons.

In order to express our results on the optimization problems in as general terms as possible, we begin by focusing on decompositions, shortest paths, and visibility in splinegons, curved polygons in which the region bounded by each curved edge and the line segment joining its endpoints is always convex [18]. The polygonal shortest path and visibility algorithms all require a triangulated polygon. Triangulation, however, is not a viable method on splinegons: it may require adding additional vertices both on the

*This research was supported in part by NSF grant CCR-88-03549, NSF grant CCR-91-04732, and by DIMACS under NSF grant STC-88-09648. Preliminary reports on this work have appeared in the *Proceedings of the 6th ACM Symposium on Computational Geometry* and in the *Proceedings of the 2nd Canadian Conference on Computational Geometry.*

boundary and in the interior; furthermore, curved triangles are not necessarily convex [18], [19], [41]. By substituting a new *bounded degree decomposition* which is linear-time equivalent to triangulation, we generate equally-efficient curved versions of the polygon algorithms for creating shortest paths and factor graphs and for solving visibility from an edge, ray-shooting, and two-point shortest paths.

We then use shortest paths to design algorithms for several types of geometric optimization problems on both polygons and splinegons: separators, inscribed triangles, circumscribed concave quadrilaterals, and contained triangles.

Separators: If two points x and y lie on the boundary of simple polygon or splinegon P and define a directed line segment $xy \subseteq P$ that separates P into two sets P_L and P_R, then xy is called a *separator.*

> **Minimum Length:** The areas of P_L and P_R are defined by constants K_L and K_R. Find a separator of minimum length such that the ratio of the areas of P_L and P_R remains equal to a given constant.
>
> **Minimum Sum of Ratios:** Find a separator that minimizes the sum of the ratio of the area of P_L to the square of its perimeter and the ratio of the area of P_R to the square of its perimeter.

Inscribed Triangles: Given a simple polygon or splinegon P, a triangle T such that $T \subseteq P$ and the vertices of T lie on the boundary of P is an *inscribed triangle.*

> **Maximum Area:** Find the inscribed triangle of maximum area.
>
> **Maximum Perimeter:** Find the inscribed triangle of maximum perimeter.
>
> **Constrained Maximum Area/Perimeter:** Find a maximum area/perimeter inscribed triangle with one edge of given length.

Circumscribed Quadrilateral: Given a simple polygon or splinegon P, a quadrilateral Q such that $P \subseteq Q$ and all four sides of Q intersect the boundary of P, Q is a *circumscribed quadrilateral.*

> **Minimum Area Concave:** Find the circumscribed concave quadrilateral of minimum area.

Contained Triangle: Given a simple polygon or splinegon P, a triangle T such that $T \subseteq P$ is a *contained triangle.*

> **Maximum Area:** Find the contained triangle of maximum area.

In each case, we find the global optimum by using shortest paths to decompose the optimization problem into a low-degree polynomial number of simple continuous optimization problems, solving each in $O(1)$ arithmetic operations by using the methods of calculus analytically, standard methods of numerical analysis, or linear programming, and computing the optimum of all the local optima. For polygons, we solve the separator problems in $O(n^2)$ time, the inscribed triangle problems in $O(n^3)$ time and the contained triangle problem in $O(n^4)$ time, all in linear space. Subsequently we combine our techniques with Hershberger's output sensitive visibility graph technique [29], to create modified algorithms for the area separator and maximum area or perimeter inscribed triangle problems which run in $O(m)$ and $O(n^2 + nm)$ time respectively, where m is the size of the visibility graph of P. The quadrilateral problem can be solved either in $O(n_c^2 n_p^2)$ time and $O(n)$ space or in $O(n_c^2(n_p + k))$ time and $O(n + k)$ space where n_c is the number of vertices of the convex hull of P, $n_p = n - n_c$ and k is an instance-dependent parameter which ranges between $O(n_p)$ and $O(n_p^2)$. Although some of the algorithms for curvilinear objects obtain the same asymptotic complexity as their polygonal counterparts, others do not: the splinegon separator algorithm runs in $O(n^2)$ time; the inscribed triangle algorithm in $O(n^4)$. We conjecture that some curvilinear problems are inherently more difficult than their polygonal counterparts.

The decomposition step used to solve both the inscribed triangle problems and the contained triangle problem focuses on a new type of computationally tractable polygon (resp. splinegon), the *fan-shaped* *polygon* (resp. *fan-shaped splinegon*). Every triangle inscribed (contained) in a simple polygon or splinegon P is also inscribed (contained) in a fan-shaped polygon or splinegon $P' \subseteq P$. We expect that the fan-shaped polygon/splinegon will become a useful tool in other applications as well.

The next two paragraphs recite some of the history of the polygon version of these problems. Lisper [32] posed the first separator problem, citing applications in solid modeling and graph cutting. A linear algorithm exists for convex polygons [36]. Chang posed the second separator problem [10], claiming applications in finite element analysis. Aggarwal ([1]) posed both the contained triangle problem and the concave circumscribed quadrilateral problem. Chang [10] and Chang and Yap [9] previously had posed the contained triangle problem as an open problem. There are numerous results for related problems. The Klee-Laskowski bound of $O(n \log^2 n)$ time for computing the minimum area triangle containing a convex n-gon time [30] was improved to linear time by O' Rourke, Aggarwal, Maddila, and Baldwin [37]. Finding the minimum area circumscribing k-gon of a convex n-gon was solved first in $O(n^2 \log k \log n)$ time [2], next in $O(n^2 \log k)$ time [3], and finally in $O(nk + n \log n)$ by Aggarwal and Park [4]. DePano's bound of $O(n^3)$ time for computing the minimum perimeter triangle circumscribing a convex n-gon [15] was improved by Aggarwal and Park to $O(n \log n)$ time [4]. Note that any convex k-gon circumscribing a simple polygon P also circumscribes the convex hull of P.

Many researchers have studied inclusion problems. Dobkin and Snyder [17] presented a linear-time algorithm for computing the minimum area triangle inscribed in a convex polygon of n vertices. Boyce *et al.* [5] computed the maximum area or perimeter convex k-gon inside a convex n-gon in $O(kn \log n + n \log^2 n)$ time. Aggarwal *et al.* [3] improved the bound to $O(kn + n \log n)$. Chang and Yap solved the general problem of finding the maximum area (perimeter) convex polygon contained within a given simple polygon P in $O(n^7)$ time (resp. $O(n^6)$ time) and $O(n^5)$ space [9],[10]. DePano *et al.* [16] gave an $O(n^3)$ algorithm for maximum area equilateral triangle contained in a simple polygon. This result can be improved using a recent result of Chew and Kedem [14] for the problem of placing the largest similar copy of a convex k-gon in an arbitrary polygonal environment. S. Fortune [22] solved the problem of placing the largest homothetic copy of a k-gon in a simple polygon in $O(kn \log kn)$ time. Some recent research has focused on simultaneous inner and outer approximation of convex polygons by a pair of rectangles [39] or by a pair of similar triangles [21].

We have recently learned of some independent work on shortest paths and visibility in curved regions. Many interesting, non-algorithmic, properties of shortest paths inside curvilinear regions appear in [7],[8]. Furthermore, Bourgin and Howe [6] provide algorithms for shortest paths between two fixed points in a Jordan region which run in $O(nk)$ time where n is the number of distinct sections of the boundary (*i.e.* number of vertices of the boundary) of the region and k is the number of the vertices on the shortest path. Our algorithm computing the lengths of the shortest paths from a fixed point to *all* the vertices of the boundary of the region runs in $O(n)$ time. Our algorithm restricted to computing the shortest path between two fixed points would use $O(n)$ time for the length computation or $O(n + k)$ time for computing the actual path, where k is the number of the vertices of the shortest path.

In the next section, section 2, we review polygon shortest path and visibility results, develop the corresponding splinegon versions, and establish the notation to be used in the paper. Sections 2.2-2.5 are long and detailed and may be omitted by a reader primarily interested in the polygonal versions of the optimization results. Sections 3 to 6 examine each of the optimization problems in turn. Preliminary results of this paper appear in [34], [35]. The results of this paper will appear also in Melissaratos's forthcoming thesis, [33].

2 Shortest paths and visibility in polygons and splinegons

2.1 Shortest paths in simple polygons

In the next few paragraphs we establish our conventions and review necessary facts and definitions. A simple polygon or splinegon P has n edges represented by the integers $1, 2, ..., n$ in clockwise order, and edge j has endpoints p_j and p_{j+1}; whenever a subset of polygon or splinegon vertices (edges) are identified by uppercase (lowercase) letters, alphabetic order implies clockwise order around P; a line l is *tangent* to a polygonal chain C if it intersects the chain in one or more points and C lies entirely in one of the halfplanes defined by l. A point $x \in P$ is *visible* from an edge i of P if there exists a point

y on i such that $xy \subseteq P$. Two edges i, j of P are *visible* from each other if and only if there exist at least two points x, y on i, j respectively such that $xy \subseteq P$. The set of points $x \in P$ which are visible from an edge i form the *visibility polygon* or *visibility splinegon* of P from edge i. If two edges i and j of P are visible from each other, the set of points of j which are visible from i form the *visible part of j with respect to i.*

In [26] the authors describe a linear time and space algorithm for finding the shortest paths from a point v inside or on the boundary of P to all its vertices, if P represents a triangulated simple polygon. The union of these paths form a tree called *the shortest path tree with respect to source v*, or just *the shortest path tree* if v is understood. The shortest path algorithm applied to a triangulated simple polygon P at a designated start vertex v produces a subdivision where each region corresponds to a *funnel* based on some polygon edge i, denoted $F_v(i)$ (see fig. 1a). Extending the edges of each funnel up to their intersection with the funnel's base produces the *extended shortest path tree* which induces a refined subdivision of P, where every region is a triangle, called the *shortest path map with respect to v* or just the *shortest path map* if v is understood. The extended shortest path tree from a vertex v subdivides each edge i of P into *elementary segments.* This set of elementary segments on edge i is denoted by $S_v(i)$ and its size by s_i^v. The union of all $S_v(i)$ over all edges i of P is called the *trace of v.* The closest vertex to x on the shortest path from v to x is called the *anchor* of x with respect to v and is denoted by $anchor^v(x)$. A fundamental property of each shortest path map of a polygon is that all points x in a particular region of the shortest path map have the same anchor (see fig. 1b) [26],[29].

One can compute the visible parts of a given edge i from every other edge of the polygon, as well as the visible parts of every edge from i, in $O(n)$ time and space using the shortest path algorithm. If edges i, j of polygon P are visible from each other then the shortest paths from p_{j+1} to p_i ($SP_{p_{j+1}}(p_i)$) and from p_{i+1} to p_j ($SP_{p_{i+1}}(p_j)$) are inward disjoint convex chains. The region bounded by the above chains and i and j is called an *hourglass* and denoted $H_{i,j}$ [26].

Assuming that polygon P is triangulated, the shortest path algorithm of [26] from a vertex s of P proceeds as follows. Assume without loss of generality that s lies on only one triangle. The computation corresponds to a preorder traversal of the binary tree with one node for each triangle, with an edge joining two nodes whose triangles share an edge, and with the triangle containing s as the root. The algorithm maintains the invariant that all funnels for polygon edges belonging to processed triangles and edges of current triangles (nodes) have been computed and are stored in finger search trees. This statement is trivially true at the outset when we process the first current triangle, the triangle containing s. It has only one edge interior to P, an active edge. Go through that edge, splitting its funnel by computing a tangent from the new vertex to one of the chains of the funnel, to form the funnel for each of the other two edges of the next triangle. If those edges both lie on the boundary of P, then this triangle is a leaf. If both are interior to P, then this triangle has two children. Otherwise, this triangle has one child [26]. The use of the triangulation of the polygon as well as of the funnels, in the computation of shortest paths inside simple polygons, appeared first in a paper by Lee and Preparata [31]. Lee and Preparata, present an algorithm to compute the length of the shortest path between two fixed points inside a simple polygon. The difference is in the following two aspects: a) funnel representation (Lee and Preparata represent the funnels as linked lists and not as finger search trees), b) when a funnel is split the algorithm of [31] recurs only to one of the funnels although the algorithm of [26] may recur to both split funnels.

2.2 Bounded degree decomposition of splinegons

To extend the polygonal shortest path algorithm [26] to work for splinegons, we first need to find an acceptable substitute for triangulation, since Dobkin *et al.* [19] have shown triangulation of splinegons to be infeasible. One candidate decomposition is the horizontal visibility map which would decompose the splinegon into horizontal trapezoids (with curved sides). Fortunately, the Tarjan-Van Wyk algorithm [42] is applicable to splinegons provided that the edges are monotone to at least one of the axes [19], as is the new linear-time Chazelle algorithm [10]. Both algorithms produce a linear number of new vertices. Ordinarily, there will be at most one interior vertex per trapezoid edge. However, in some applications, several vertices may have the same y-coordinate, producing an arbitrary number of vertices within a base of a trapezoid. Thus in theory some trapezoids could have an unlimited number of neighbors. Our goal is to refine the curvilinear trapezoids so that every component has at most three neighbors so that the dual of the decomposition is a binary tree, a key characteristic of polygonal triangulations.

To guarantee that this decomposition is sufficiently general, we need to verify that even in these degenerate cases, the decomposition can be accomplished by adding new vertices only on splinegon boundaries. We call our decomposition of a simple splinegon, into components with at most four sides and with at most three neighbors, the *bounded degree* decomposition. To begin, we preprocess the edges of the splinegon such that each edge is monotone with respect to *both* the x and the y axes, *i.e.* insert the extrema of each splinegon edge with respect to either axes as a new vertex, without duplicating edge endpoints. The convexity of the splinegon edges means that each edge can have at most one minimum and at most one maximum relative to each axis, and, consequently, the additional number of edges or vertices is at most $4n$. If constant time suffices to compute the extrema of any edge, then this preprocessing uses $O(n)$ time.

Next, we separate the set of curved trapezoids and triangles of the horizontal visibility decomposition, into three groups: *group I* contains those which have two side edges concave; *group II*, those with one side concave and the other convex; and *group III*, the ones with both sides convex. Remember that, given our preprocessing, all curved edges are both x-monotone and y-monotone. We focus primarily on *group I*, since the techniques we develop for that group can clearly be applied for the second and third groups also, although simpler procedures for those groups would suffice. We classify each trapezoid $ABCD$ of *group I* with bases AB, CD and side edges AD, BC as having one of three subtypes, by comparing the projections of each of AB, CD to a line parallel to both:

Type 1: The projections intersect but neither is contained in the other
Type 2: The projection of AB is contained in the projection of CD.
Type 3: The projections have empty intersection.

For all three types, call a vertex interior to a base which vertically projects onto a curved side of the trapezoid an $a-point$, and add its projection as a $new-vertex$. All other vertices interior to bases are called $b-points$ (fig. 2).

If $ABCD$ belongs to Type 1, then connect A (C resp.) to any $b-points$ between C (A resp.) and E (F resp.) where E (F resp.) is the projection of A (C resp.) onto CD (AB resp.); if there are no $b-points$ then insert the diagonal AC. Connect each new point on the splinegon edge DA (BC resp.) generated by an $a-point$ on DC (BA resp.) to the next vertex on that edge by a diagonal (fig. 3a).

If $ABCD$ belongs to Type 2, let E, F be the projections of B, A on DC, respectively, such that E, F lie in segment DC. If there exists at least one $b-point$ on CD then the decomposition is done as in fig. 3b. If there are no $b-points$ on CD and there are on AB then let G (H resp.) be the vertical projection on AD (BC resp.) of the last $a-point$ as we move rightwards (leftwards resp.) from D (C resp.). Assume that the y-coordinate of G is larger than that of H. Let I represent the horizontal projection of G on BC. Then what remains is to decompose the trapezoid $ABIG$ if it has $b-points$ on AB by connecting G to all $b-points$ on AB (fig. 3c). If $ABCD$ is of Type 3 then there are no $b-points$ and the decomposition is given in fig. 3d.

Finally, where possible, all remaining quadrilaterals are triangulated using a diagonal. Given that splinegon edges are monotone with respect to both x and y axis, the only class of quadrilaterals that cannot be triangulated by a diagonal consists of trapezoids of type 3: trapezoids with concave-out curved side edges on the boundary of the splinegon and parallel bases with disjoint x-intervals. These quadrilaterals have exactly two neighbors.

Theorem 2.1 *Any simple splinegon can be decomposed in such a way that each component has at most three neighbors, in the same asymptotic time complexity as triangulating a simple polygon.*

Proof: The algorithm described above first computes the horizontal visibility information and then spends at most constant time per vertex refining the decomposition. Fournier and Montuno have proved that triangulating a polygon is linear-time equivalent to computing horizontal visibility information [23]. □

2.3 Shortest Paths in Splinegons

Minor revisions allow the polygonal shortest path algorithm [26] to work for curved polygons, also known as *splinegons*, in comparable time and space bounds. The curved algorithm maintains the corresponding invariant that all funnels for polygon edges belonging to processed and current components of the bounded degree decomposition have been computed and are stored in finger search trees, but there are several notable differences. A convex chain of a funnel is not necessary made of straight line segments but is a concatenation of straight line segments and convex curved segments. If the shortest path map is formed by extending the straight line segments of the funnels and the tangents at the endpoints of the curved segments up to the intersection of the corresponding splinegon edge, for a point x moving within a single component of the shortest path map, $anchor^s(x)$ is not a constant function as in the polygon case. Instead, $anchor^s(x)$ varies over a particular convex section of a single curved edge of the splinegon boundary. Although in the polygon case the funnel splitting operation does not create new vertices on the boundary of the polygon, with splinegons, a new vertex may be created either in the funnel or in the boundary of a new component or on both. All of these differences can be accommodated.

Theorem 2.2 *The shortest path tree inside a simple splinegon with a designated root can be computed in $O(n)$ time, given the bounded degree decomposition.*

Proof: The main step of the polygon algorithm is as follows: given a funnel and a triangle, with one of its sides coincident with the funnel base, split the funnel into at most two funnels which have bases the other two edges of the triangle. The funnel algorithm needs to be revised to accommodate different types of regions: triangles with one or more curved edges; straight-edged triangles; and quadrilaterals with two curved edges. Clearly a region with two children must be a straight-edged triangle. Thus, only regions with at most one child need different processing. If the total contribution of the one-child components to the complexity of the algorithm remains $O(n)$, then the recursive formula used in [26] to prove the linearity of the entire algorithm still applies. There are two main changes to make. One is that splitting the funnels may involve computing tangents from a point to a curve or between a pair of curved edges. But we may assume that each curved operation requires constant time [18], so there is no asymptotic penalty.

More importantly, at a node corresponding to a quadrilateral, there are two splitting points rather than just one (see fig. 4). Let t_1, t_2 be the two splitting points of the current funnel on base AB. Let n_1, n_2, n_3 be the number of funnel vertices between A and t_1, t_1 and t_2, t_2 and B, respectively. In fig. 4, the current funnel is split into three funnels: the first has apex t_1 and consists of the convex chain from A to t_1 together with the common tangent t_1q_1 and has the convex segment Aq_1 as base; the second funnel has apex t_2 and consists of the splinegon boundary segment Dq_1 followed by the common tangent t_1q_1, by the convex subchain t_1t_2, the common tangent t_2q_2, and by the splinegon boundary segment q_2C; the third funnel has apex s, has the splinegon boundary segment q_2B as base, and consists of the common tangent t_2q_2 followed by the convex subchain t_2s and the convex subchain sB. The first and the third of these funnels are not processed further since their bases lie on the splinegon boundary. Thus the vertices of the original funnel between A and t_1 and between t_2 and B will not be used again by the algorithm. Since the funnels are represented by finger trees, the first splitting and tangency operation takes time $O(min(log(n_1), log(n_2 + n_3)))$ and the second $O(min(log(n_2), log(n_3)))$. But the first is $O(n_1)$ and the second $O(n_3)$. Therefore the total complexity for that case is $O(n_1 + n_3)$. But $n_1 + n_3$ is the number of "dead" vertices of the funnel (i.e. the vertices which are not going to be processed further). Therefore summing over all zero or one child cases gives $O(n)$. □

We can also compute the shortest path tree inside a simple splinegon directly from the horizontal visibility decomposition without computing the bounded degree decomposition, producing an alternate proof:

Proof: At any time, we consider the current funnel and a trapezoidal component of which the parallel sides may contain many splinegon vertices (fig. 5). The shortest paths from the apex α of the funnel to $v_1, v_2, ..., v_k$ and to $w_1, w_2, ..., w_l$ create new funnels with bases v_iv_{i+1} for $i = 1, ..., k$ and w_jw_{j+1} for $j = 1, ..., l$. It suffices to solve the following subproblem: given a funnel with apex α, base bc and convex chains F_1 and F_2, a trapezoid $bcde$ and a point x on de, find the shortest path from α to x which lies inside the area defined by the funnel and the trapezoid. Call the curved sides of the trapezoid C_1 and

C_2. Consider the tangent from x to the funnel. Let t be the corresponding tangent point. Without loss of generality assume t lies on F_1. We have the following cases (fig. 6).

1. xt does not intersect any of the C_1 or C_2. Then the shortest path from α to x is the concatenation of the part of F_1 from α to t and the segment xt.

2. xt intersects only one of the C_1 or C_2.

   (a) xt intersects C_1. Let fg be the common outer tangent of F_1 and C_1, where f is on F_1 and g is on F_2.

      i. fg does not intersect C_2. Then let xh be the tangent from x to C_1. Then the shortest path is af, fg, gh, hx.

      ii. fg intersects C_2. Then let ij be the inner common tangent of F_1 and C_2 and kl be the inner common tangent of C_1, C_2. Then the shortest path is ai, ij, jk, kl, lh, hx.

   (b) xt intersects C_2. Use the same steps as in *case 2(a)* but let fg be the inner common tangent of F_1 and C_2.

3. xt intersects both C_1 and C_2.

   (a) As we move from x to t, xt intersects first C_1 and then C_2. Then let fg be the inner common tangent of F_1, C_2 and ij the inner common tangent of C_1 and C_2. Then the shortest path is af, fg, gi, ij, jh, hx.

   (b) As we move from x to t, xt intersects first C_2 and then C_1. The same as above with the only difference that fg is the outer common tangent of F_1, C_1.

We use this funnel splitting operation recursively. Following the above approach a funnel may be split into more than two subfunnels, thus we cannot apply the recurrence formula as in the two-way splitting. But this multiway splitting can be simulated by two-way splittings. The horizontal visibility decomposition of the splinegon is a planar subdivision where its dual is a tree, not necessarily binary. Make that tree a rooted tree, choosing arbitrarily any node as the root. Thus in our rooted tree every node except the root has indegree equal to one. We now apply a well-known transformation which converts any tree to a binary one. Assume that node v has parent node u and children nodes w_1, w_2, w_3, w_4. The transformation constructs a binary tree with root v and leaves w_1, w_2, w_3, w_4 (fig. 7). Then for the complexity analysis the same arguments can apply since we have to work with a binary tree. We must note that the above transformation has nothing to do with the implementation of the algorithm. It is useful only for the complexity analysis. □

The second approach uses the horizontal visibility decomposition directly instead of the bounded degree decomposition and applies multiway splitting of the funnels instead of the original two-way splitting. The two approaches are equivalent from the point of view of asymptotic time complexity but not in practice. Although to get the bounded degree decomposition requires only linear time, the constant may represent computation of intersections of higher degree curves. The second approach avoids these computations, but requires a somewhat complicated proof that multiway splitting does not affect the linearity of the algorithm. It might seem, therefore, that the concept of the bounded degree decomposition is unnecessary. Some visibility problems do not present this option. Although computing the visibility splinegon from an edge depends only on shortest paths, however they are obtained, problems like ray-shooting and the two-point shortest path problem depend on an augmented balanced decomposition tree such as the *factor graph* which is computable from the bounded degree decomposition.

2.4 Factor graphs of splinegons

A triangulation of a polygon can be converted in linear time to a balanced binary decomposition tree in which each node corresponds to a subpolygon P and to a diagonal d which divides P so that neither of the two children subpolygons P_L and P_R contains more than $2/3$ of the triangles of P; d roughly bisects P [26]. As all of the diagonals in the bounded degree decomposition of a splinegon S are straight segments, this algorithm extends directly to splinegons.

Assume that S is the initial polygon or splinegon and that we are given a balanced decomposition tree for S. Let S_l be a polygon or splinegon at level l in the decomposition tree, and let d_l be the roughly bisecting diagonal of S_l. The boundary of S_l consists of some edges of S and some diagonals. The *factor graph* [13] has edges between d_l and the bounding diagonals of S_l; in other words, edges of the factor graph correspond to pairs of bisecting diagonals. Some visibility applications need an augmented factor graph in which each edge is equipped with a representation of the hourglass corresponding to that pair of diagonals. The bottom-up polygon algorithm for creating the (augmented) factor graph extends easily to splinegons. Beginning with the balanced decomposition tree, construct the trivial hourglasses for regions represented by the leaves. Now assume that all hourglass computation up to level k has been completed. Thus, for any splinegon component in levels 1 to k the hourglasses between any pair of bounding diagonals have been computed. To proceed to level $k+1$, "delete" all the diagonals at level k and compute the hourglasses between any bounding diagonal of the left component and any bounding diagonal of the right component by trimming and then concatenating the two hourglasses at level k. The hourglasses may now contain both straight edges and portions of curved edges so that the trimming operation may involve computing tangents to curved edges but the essential procedure is unchanged.

Since representing an hourglass explicitly uses $O(n)$ space, the augmented factor graph could use $O(n^2)$ space overall. By keeping each edge of an hourglass only at the highest level in which it appears in the tree, Chazelle and Guibas [13] demonstrated that the augmented factor graph could be designed to have the following properties: the augmented factor graph has size $O(n)$, each node has degree $O(\log n)$, and the graph can be constructed from the decomposition of the polygon in $O(n)$ time.

Theorem 2.3 *The factor graph and the augmented factor graph of a simple splinegon S can be computed in $O(n)$ time and space, given the bounded degree decomposition.*

2.5 Visibility results for splinegons

In this section, we consider the following three problems:

1. *Visibility from an edge:* Given an edge j of splinegon S, find the points x on the boundary of S for which there exists at least one point y on j such that $xy \subseteq S$.

2. *Ray-shooting:* Given a simple splinegon S, a query point q and a ray passing through q find the first intersection of the ray with the splinegon.

3. *Two point shortest path problem:* Given a simple splinegon S preprocess it in order to construct a data structure such that given any two query points p and q the length and the shortest path itself can be computed efficiently.

Each of them can be solved efficiently using either shortest paths or factor graphs. The last two, however, also use planar point location. To preprocess S for this purpose, first construct the convex hull of S, $CH(S)$, in linear time [18],[40]. Given the bounded degree decomposition both of S and of the pockets identified in the process of computing the convex hull, a *layered dag* can be constructed in linear time, allowing the location of a query point in a component of the decomposition to be determined in logarithmic time [20].

Theorem 2.4 *Computing the part of the boundary of a simple splinegon S of n vertices which is visible from an edge requires $O(n)$ time given the horizontal visibility decomposition of S.*

Proof: The linear-time polygon algorithm uses the fact that if edges i, j are visible from each other then the shortest paths from p_{j+1} to p_i ($SP_{p_{j+1}}(p_i)$) and from p_{i+1} to p_j ($SP_{p_{i+1}}(p_j)$) are inward disjoint convex chains [26]. For splinegons, this fact does not hold (fig. 8). We present a new method based on local computations for computing the visibility of an edge in either a polygon or a splinegon.

To compute the visible region from edge j, find the shortest path maps from p_j and p_{j+1} respectively. Merge $trace(p_j)$ and $trace(p_{j+1})$ into a linear-sized subdivision M. If x moves along an elementary segment I of M, the anchors of x with respect to the endpoints of j remain unchanged. Thus we can unambiguously refer to $anchor^{p_j}(I)$ and $anchor^{p_{j+1}}(I)$. For each I of M, perform the following simple test: if $anchor^{p_{j+1}}(I) <> anchor^{p_j}(I)$ then for every point x on I, x is visible from edge j; call such a segment I a *valid* segment. Merge adjacent valid segments and then report the results. □

Theorem 2.5 *Given the bounded degree decomposition, the factor graph, and the layered dag of a simple splinegon S and all of its pockets, a query point q and a ray passing through q, the first intersection of the ray with the splinegon can be reported in $O(\log n)$ time.*

Proof: Locate the query point q in the decomposition using the layered dag [20]. If q is outside $CH(S)$, then determine the convex hull edge first crossed by the shooting ray in logarithmic time [12],[18]. If it is an edge of S, report it. If not, perform rayshooting as described below in the pocket having that edge as a lid, a new splinegon. As in [13], if q lies within S, find the diagonal crossed by the shooting ray which is closest to the root of the decomposition tree. Descend the augmented factor graph as follows: at each node visited check either its $L(v)$ or $R(v)$ list; for each w in $L(v)$, test if the ray from q avoids the hourglass corresponding to the edge (v, w); at a leaf, no such hourglass exists but the edge of the splinegon intersected by the ray can be computed in $O(1)$ time. At first glance, it seems we need $O(\log^2 n)$ time. To achieve the $O(\log n)$ complexity, transform the factor graph so that it has bounded degree. Then, using fractional cascading, the $O(\log n)$ intersection tests between convex chains and the line can all be accomplished in $O(\log n)$ time. This algorithm differs from the original polygon algorithm [13] in only one respect. In the polygon algorithm, the test of whether a line intersects an hourglass is transformed to the dual problem of point inclusion in a convex polygon, solvable using a variant of binary search. Since no duality transforms are known to apply to curved objects, we solve the line-hourglass intersection problem directly using binary search on the two convex chains bounding the hourglass. □

Theorem 2.6 *Given two query points p and q and the bounded degree decomposition, the factor graph, and the layered dag of a simple splinegon S and its pockets, the shortest path from p to q and its length can be reported in $O(\log n + k)$, where k is the number of segments in the path.*

Proof: The polygon algorithm of [25] extends directly. □

3 Separators

In this section we solve two optimum polygon separator problems and then generalize those solutions to accommodate splinegons. It should be clear that an area separator does not always exist. For example, there are polygons (splinegons) that cannot be bisected by a single segment. Our algorithm finds an optimum separator if one exists or reports the non-existence otherwise. Given that we solve both problems in a uniform way, we describe the solution to the minimum length separator problem in depth and then refer briefly to the minimum sum of ratios problem.

If x and y delimit a separator for simple polygon (splinegon) P, then x and y are visible in P. Thus, if x lies on edge i and y on edge j, then i and j are visible in P. Furthermore, the line segment xy lies in the hourglass $H_{i,j}$ defined by the shortest paths from p_{j+1} to p_i ($SP_{p_{j+1}}(p_i)$) and from p_{i+1} to p_j ($SP_{p_{i+1}}(p_j)$). Thus our goal is to reduce the optimum separator problem for a simple polygon to a series of optimum separator problems on hourglasses which are simpler to solve.

Define as a_L (a_R resp.) the area of P to the left (right resp.) of the directed segment xy (see fig. 9). Not every hourglass will admit a separator satisfying the constraints $a_L = K_L$ and $a_R = K_R$ for constants K_L and K_R where $K_L + K_R = A$, where A is the area of P. Note, however, that $SP_{p_i}(p_{j+1})$ cuts the polygon into two or more pieces some to the left and some to the right. The area of P to the left (resp. right) of $SP_{p_{j+1}}(p_i)$ is denoted $AL_{p_{j+1}}(p_i)$ (resp. $AR_{p_{j+1}}(p_i)$) . Therefore a necessary and sufficient condition for H_{ij} to contain a separator xy is $AL_{p_{j+1}}(p_i) \leq K_L$, and $AR_{p_j}(p_{i+1}) \leq K_R$. Thus when we consider hourglass H_{ij} we need to know both $AL_{p_{j+1}}(p_i)$ and $AR_{p_j}(p_{i+1})$. It is clear that computing these quantities for a specific H_{ij} could use $\Theta(n)$ time, leading to a time complexity of $O(n^3)$ for all $O(n^2)$ hourglasses. For a fixed vertex v, however, the shortest path map from v divides P

into a linear number of triangles where the funnel $F_v(i)$ on edge i is the disjoint union of some subset of these triangles. Therefore, we can compute all $AL_v(p_i)$ for $i = 1, ...n$ incrementally in $O(n)$ time.

Below, we present a high level description of our algorithm for computing the minimum-length separator, where *shortestpath(v)* represents the procedure which computes the shortest path tree (map) from vertex v, $a(F)$ denotes the area of funnel F, and $hourglass(i, j, locmin)$ computes the minimum-length separator for $H_{i,j}$.

For $j = 1$ to n do begin
    $globmin = \infty$;
    $shortestpath(p_j)$;
    $shortestpath(p_{j+1})$;
    for $i = 1$ to n do
        if i, j are visible and $AL_{p_{j+1}}(p_i) \leq K_L$ and $AR_{p_j}(p_{i+1}) \leq K_R$
        then do begin
            hourglass(i,j,locmin)
            $globmin = MIN(globmin, locmin)$;
        end;
end;

Next, we need a procedure *hourglass(i,j,locmin)* to solve the following problem:

> **Problem:** Given an hourglass $H_{i,j}$ find x and y on i and j, respectively, such that: a) $xy \subseteq H_{i,j}$; b) the area bounded by $SP_{p_{j+1}}(p_i)$, $p_i x$, xy, and yp_{j+1} equals $K_L - AL_{p_{j+1}}(p_i)$; and c) the length of xy is minimum.

First, we simplify the test of condition b). For each hourglass $H_{i,j}$, define $C_{p_{j+1}}(p_i)$ as the area of the region bounded by $SP_{p_{j+1}}(p_i)$ and the segment $p_i p_{j+1}$. Depth-first-search traversal of the shortest path tree of P from vertex p_{j+1} produces all of the $C_{p_{j+1}}(p_i)$ in linear time:

procedure $convex - area(v, s)$;
Begin
for all neighbors w of v do
    if the path from s to w is a counterclockwise convex chain
    then begin
        $C_s(w) := C_s(v) + area(\Delta svw)$;
        $convex - area(s, w)$;
    end;
End;

The test of condition b) now reduces to determining whether the quadrilateral $p_i x y p_{j+1}$ has area $K_L - AL_{p_{j+1}}(p_i) + C_{p_{j+1}}(p_i)$, from now on referred to as K.

Condition a) is satisfied if and only if the closed halfplane to the left of the $\overrightarrow{xy}$ contains all the vertices of $SP_{p_{j+1}}(p_i)$ and the one to the right contains all the vertices of $SP_{p_{i+1}}(p_j)$. This condition could produce a linear number of constraints. We exploit the fact that shortest paths are convex to decompose the problem further, reducing the number of constraints. First, trim edge i to create an edge i' so that all of i' is visible from j. Next, subdivide i' into elementary segments by merging $S_{p_j}(i)$ and $S_{p_{j+1}}(i)$. As x moves from p'_i to p'_{i+1}, $anchor^{p_{j+1}}(x)$ and/or $anchor^{p_j}(x)$ changes only when x moves from one subinterval to the next. Thus, it is reasonable to refer to $anchor^{p_j}(I)$ and $anchor^{p_{j+1}}(I)$. Consequently, we can reduce an arbitrary hourglass problem to a series of problems defined on elementary hourglasses:

> **Problem:** For an elementary segment I of i, find points $x = (x_1, y_1)$ and $y = (x_2, y_2)$ on I and j respectively such that a) the $anchor^{p_j}(I)$ and $anchor^{p_{j+1}}(I)$ do not lie in the same open halfspace defined by $\overrightarrow{xy}$, b) the area of the quadrilateral $p_i x y p_{j+1}$ equals K, and c) length of xy is minimum.

This is a continuous optimization problem, rather than a combinatorial one, which generates the following constraints, where $p_i = (k_1, l_1)$, $p_{j+1} = (k_2, l_2)$ and (a_i, b_i) represents the slope and y-intercept of the line containing edge i.

$area(p_i x y p_{j+1}) = x_1 y_2 - x_2 y_1 + x_2 l_2 - k_2 y_2 + k_2 l_1 - k_1 l_2 + k_1 y_1 - x_1 l_1 = K$ (1)

x lies on line containing i: $y_1 = a_i x_1 + b_i$ (2)

y lies on line containing j: $y_2 = a_j x_2 + b_j$ (3)

Substitution of (2) and (3) in (1) produces constants B_i such that:

$$x_2 = \frac{B_1 x_1 + B_2}{B_3 x_1 + B_4} \quad (4)$$

Substitution of (2), (3), and (4) in the expression for the length of xy,

$$L = \sqrt{((x_1 - x_2)^2 + (y_1 - y_2)^2)},$$

gives the length as the square root of a rational function of one variable x_1. The domain of x_1 is restricted by the constraints that x must lie within the elementary segment I, the point y lies on edge j, and the line through x and y must keep the $anchor^{p_j}(I)$ and $anchor^{p_{j+1}}(I)$ on opposite sides. The length function L is the square root of a rational function of x_1. The degrees of the numerator and denominator of the rational function permit analytical solution for finding the optimum in constant time, even in the restricted domain. The global minimum for the original hourglass is the minimum of all the minima obtained from the continuous problems on elementary hourglasses. Therefore:

Theorem 3.1 *For an hourglass H_{ij} the minimum length separator problem can be solved in $O(h_{ij})$ time and space where h_{ij} is the number of vertices of H_{ij}.*

Since the size of an hourglass H_{ij} is in the worst case $O(n)$ where n is the number of vertices of polygon P and since we call the hourglass algorithm at most $O(n^2)$ times, the time complexity of the entire separator algorithm is at most $O(n^3)$. We exploit the linearity of the shortest path trees to improve that bound.

Theorem 3.2 *The minimum length area separator of a simple polygon can be computed in $O(n^2)$ time and $O(n)$ space.*

Proof: We have $O(n)$ calls to shortest path algorithm for a total of $O(n^2)$ time. $O(n^2)$ time is spent computing the $AL_v(w)$. Finally, for a particular hourglass H_{ij}, we spend $O(s_j^{p_i} + s_j^{p_{i+1}})$. Thus the whole algorithm takes:

$O(\sum_{i=1}^{n} \sum_{j=1}^{n} (s_j^{p_i} + s_j^{p_{i+1}}))$ which is clearly $O(n^2)$ since

$$\sum_{k=1}^{n} s_k^v = O(n)$$

for any vertex v. □

An alternate proof of the theorem uses the following lemma which is interesting in its own right:

Lemma 3.3 *The sum of the sizes of all (open) hourglasses of a simple polygon P is $O(n^2)$.*

Proof: Let h_{ij} be the size of the hourglass defined by the visible edges i and j and let ad, bc the inner common tangent segments of the convex chains of the hourglass (see fig.10). But

$$s_j^{p_i} = ab + p_{j+1}b + p_j d \quad (1)$$
$$s_i^{p_j} = cd + p_{i+1}c + p_i a \quad (2)$$

(1) and (2) imply $s_j^{p_i} + s_i^{p_j} = h_{ij}$

Then

$$\sum_{i=1}^{n}\sum_{j=1}^{n} h_{ij} = \sum_{i=1}^{n}\sum_{j=1}^{n}(s_j^{p_i} + s_i^{p_j})$$

The last sum is $O(n^2)$ since

$$\sum_{k=1}^{n} s_k^v = O(n)$$

for any vertex v. □

The alternate algorithm works are follows. Find the shortest path maps from p_j and p_{j+1} respectively. Merge $S_{p_j}(i)$ and $S_{p_{j+1}}(i)$ for every i into a linear-sized subdivision M. If x moves along an elementary segment I of M, the anchors of x with respect to the endpoints j remain unchanged. Thus we can unambiguously refer to $anchor^{p_j}(I)$ and $anchor^{p_{j+1}}(I)$. For each I of M, perform the following simple test: If $anchor^{p_{j+1}}(I) <> anchor^{p_j}(I)$ then for every point x on I, x is visible from edge j; for each segment I solve the continuous optimization problem discussed above. Repeat the same process for all edges j.

We now generalize this result to splinegons:

Theorem 3.4 *The minimum length area separator of a simple splinegon can be computed in $O(n^2)$ time and $O(n)$ space.*

Proof: The hourglass problem for splinegons becomes somewhat more difficult since the anchor can be a curved segment, rather than a single vertex. Nonetheless, it is possible to alter the constraints of the elementary hourglass problem to require that a curved segment lie above a line rather than a point lie above a line. The revised problem can still be solved in constant time. For a particular fixed edge i of P we have to scan all elementary segments I of the merged subdivision M. For each such segment we have to solve a continuous optimization problem which takes $O(1)$ time to solve. Because of the linearity of the shortest path trees we have $O(n)$ elementary segments, that implies $O(n)$ time to find a separator xy when y lies on a fixed edge i. Summing all over edges i we get an $O(n^2)$ time algorithm.□

Theorem 3.5 *The minimum sum of ratios separator of a simple polygon P of n vertices can be computed in $O(n^2)$ time and $O(n)$ space.*

Proof: The minimum-sum-of-ratios separator problem can be solved in a similar fashion to the minimum-length separator problem. A new easily computed parameter is needed: $LP_{p_{j+1}}(p_i)$ represents the length of the boundary of P from p_{j+1} to p_i in clockwise order. All of the constraints in the continuous optimization problem remain unchanged, but we need to minimize a more complicated expression. The area a_L to the left of xy equals area$(p_i x y p_{j+1}) + AL_{p_{j+1}}(p_i) - C_{p_{j+1}}(p_i)$. The perimeter p_L to the left of xy is $LP_{p_{j+1}}(p_i) + length(p_i x) + length(y p_{j+1}) - length($all polygon edges on $xy)$. Compute a_R and p_R comparably. We need to minimize $\frac{a_L}{{p_L}^2} + \frac{a_R}{{p_R}^2}$. The expression can be reduced to an optimization problem in two variables which can be solved with classical methods. □

We can combine our method with that of Hershberger's of finding the visibility graph of a simple polygon to improve our result to $O(m)$ time where m is the size of the visibility graph of polygon P. In order to do that, we fix a particular edge j and find the shortest path map from one of its endpoints, say p_j. Then starting from the other endpoint p_{j+1}, we construct the shortest path map from p_{j+1} incrementally, as in [29]. In order to combine it with our method, at each step, update the appropriate areas and solve the corresponding continuous optimization problem. Repeat the whole process for every j. Thus:

Theorem 3.6 *The minimum length area separator for a simple polygon can be solved in $O(m)$ time and $O(n)$ space where m is the size of the visibility graph of the polygon.*

Using the same reasoning we can prove that

Theorem 3.7 *The minimum sum of ratios separator of a simple polygon P of n vertices can be computed in $O(m)$ time and $O(n)$ space.*

4 Inscribed Triangles

For three points x, y, z on the boundary of a simple polygon P to define an inscribed triangle, it is necessary and sufficient that they be pairwise visible. If x, y, z lie on edges k, i, j, respectively, then the points are pairwise visible if and only if xy, yz and zx lie inside $H_{k,i}$, $H_{j,i}$ and $H_{k,j}$, respectively. Thus the boundary of the triangle xyz is interior both to P and to the union of the three hourglasses. Since P is simple, the entire triangle must be interior to P. It is also contained in the polygon $F_{i,j,k} \subseteq P$ bounded by i, $SP_{p_{i+1}}(p_k)$, k, $SP_{p_{k+1}}(p_j)$, j and $SP_{p_{j+1}}(p_i)$.

$F_{i,j,k}$ is called a *fan-shaped polygon* with *bases* i, j, k (see fig. 11). $F_{i,j,k}$ is *legal* iff the visible parts of each of its bases with respect to the two others have non-empty intersections. A triangle T is *inscribed* in $F_{i,j,k}$ if and only if $T \subseteq F_{i,j,k}$ and the vertices of T lie on the bases. Every triangle inscribed in a simple polygon P is also inscribed in a fan-shaped polygon $F_{i,j,k} \subseteq P$. Below we present a high level description of our algorithm for computing the maximum area/perimeter inscribed triangle, where *shortestpath(v)* represents the procedure which computes the shortest path tree (map) from vertex v, and $fan(i, j, k, locmax)$ computes the maximum triangle inscribed in a legal $F_{i,j,k}$.

```
globmax = 0;
for i = 1 to n do begin
    shortestpath(p_i); shortestpath(p_{i+1});
    for j = 1 to n do begin
        shortestpath(p_j); shortestpath(p_{j+1});
        for k = 1 to n do begin
            if F_{i,j,k} is legal (if edges i, j, k are pairwise visible)
            then fan(i,j,k, locmax);
            globmax = MAX(globmax, locmax);
        end;
    end;
end;
```

It remains to develop the procedure $fan(i, j, k, locmax)$.

Lemma 4.1 *For a fan-shaped polygon $F_{i,j,k}$ the maximum-area inscribed triangle with vertices x, y, z on the bases, must have at least two sides tangent to the convex chains of the fan.*

Proof: By contradiction. In specific, assume that neither xy nor xz is tangent to the boundary of $F_{i,j,k}$. Tangents from y and z to the chains $SP_{p_i}(p_{k+1})$ and $SP_{p_{j+1}}(p_k)$, respectively, intersect k at points v and w such that x must lie between them. Then it is clear that one of vyz or wyz must have area greater than or equal to the area of xyz (the equality happens when k is parallel to yz) (see fig. 12). □

Lemma 4.2 *Given two line segments AB and CD. Let x be a point on CD. Then the triangle xAB has maximum perimeter L when either $x = C$ or when $x = D$.*

Proof: The locus of points x such that triangle xAB has perimeter L is an ellipse with foci A and B which contains CD but has at least one point of CD on its boundary. Either C or D lie on the boundary. □

Lemma 4.3 *For a fan-shaped polygon, the maximum-perimeter inscribed triangle with vertices on the bases must have at least two sides tangent to the convex chains of the fan.*

Proof: Bring tangents as we did in Lemma 4.1 and then apply Lemma 4.2. □

Assume, without loss of generality, that the optimum triangle Δxyz has xy tangent to $SP_{p_{i+1}}(p_k)$, and xz tangent to $SP_{p_j}(p_{k+1})$. To find the optimum, trim edge k to create a maximal edge k', every point of which is visible from both i and j. If k' is empty, then no inscribed triangle exists. Perform comparable operations on edges i and j, creating i' and j' respectively. Next, subdivide k' by merging $S_{p_{j+1}}(k)$, $S_{p_j}(k)$, $S_{p_i}(k)$, $S_{p_i}(k)$. Call the resulting subdivision M. As x moves from p'_k to p'_{k+1}, $anchor^{p_{i+1}}(x)$

and/or $anchor^{p_j}(x)$ changes only when x moves from one subinterval to the next. Consequently, we can reduce an arbitrary fan-shaped polygon problem to a series of problems defined on simpler fan-shaped polygons:

> **Problem:** For each interval K of k', find points $x \in K$, $y \in i$, $z \in j$ such that a) $anchor^{p_{i+1}}(K) \in xy$ and $anchor^{p_j}(K) \in xz$, b) y and z are mutually visible, and c) area/perimeter of xyz is maximum.

To test condition b), we must be able to detect possible intersections of yz with the boundary of P, in particular with the convex chain $SP_{p_{i+1}}(p_j)$.[1] This process could still be difficult, so we choose to decompose the problem further. Subdivide i' according to $S_{p_j}(i)$ and subdivide j' according to $S_{p_{i+1}}(j)$. As y (z resp.) moves from p'_i to p'_{i+1} (p'_j to p'_{j+1} resp.) $anchor^{p_{j+1}}(y)$ ($anchor^{p_i}(z)$ resp.) changes only when y (z resp.) moves from one subinterval to the next. Let W be the number of vertices of either $S_{p_{j+1}}(p_i)$ or $S_{p_i}(p_{j+1})$. To each interval of i' (resp. j') assign a number called its rank which corresponds to the position of its anchor in $SP_{p_i}(p_{j+1})$ (resp. $SP_{p_{j+1}}(p_i)$), assuming that the first position is 0.

Refine the subdivision of k' further so that whenever $x \in K$, y and z each have constant rank. Let the rank of K equal the sum of those ranks. The algorithm is straightforward. If $rank(K) < W$ then do nothing. Since y, z are not visible from each other. If $rank(K) > W$, solve Problem I; if $rank(K) = W$ solve Problem II.

> **Problem I:** Given three non intersecting line segments AB, CD, and EF and two points p, q such that p (q resp.) lies on AE and BF (AC and BD resp.), find s,t,u on AB,EF,CD with p (q resp.) on st (su resp.), such that the area of stu is maximum.

> **Problem II:** Add the constraint that the line through t and u should be always above a constant point (x_0, y_0).

Solutions to both problems can be computed analytically.

Solution of Problem I: The objective is to compute the coordinates of s, t and u. Call the coordinates of these points x_1, y_1, x_2, y_2, and x_3, y_3 respectively. Let (k_1, l_1) ((k_2, l_2) resp.) be the coordinates of point p (q resp.).

Points s, t, u lie on three different lines:

$$\begin{aligned} y_1 &= a_1x_1 + b_1 \quad (1)\\ y_2 &= a_2x_2 + b_2 \quad (2)\\ y_3 &= a_3x_3 + b_3 \quad (3)\end{aligned}$$

Segments st and su pass through points p and q respectively:

$$\begin{aligned} x_1y_2 - x_2y_1 - k_1(y_2 - y_1) - l_1(x_1 - x_2) &= 0 \quad (4)\\ x_1y_3 - x_3y_1 - k_2(y_3 - y_1) - l_2(x_1 - x_3) &= 0 \quad (5)\end{aligned}$$

Algebraic manipulation of the equations 2, 3, 3, 5, 5 produces constants A_i and B_i such that:

$$\begin{aligned} x_2 &= \frac{A_4 - A_2x_1}{A_1x_1 - A_3} \quad (6)\\ x_3 &= \frac{B_4 - B_2x_1}{B_1x_1 - B_3} \quad (7)\end{aligned}$$

The area of a triangle is given by:

$$A = x_3y_2 - x_2y_3 + x_1y_3 - x_3y_1 + x_2y_1 - x_1y_2 \quad (8)$$

[1] The segment yz cannot intersect chains $SP_{p_i}(p_k)$ and $SP_{p_{j+1}}(p_k)$, since both chains lie outside of the convex angle yxz.

Substitution of the above equations into the last one produces a rational function of one variable x_1 where the numerator is a polynomial of degree three and the denominator is of degree two. The extrema can be computed using calculus: the derivative of this function is a rational function where the numerator is of degree at most four, but the roots of any algebraic equation on one variable up to degree four can be found in closed form. The domain of variable x_1 is fixed by the facts a) that s, t, u lie on AB, EF, CD respectively and b) that points k and l are always on sides st and su respectively. □

Solution to Problem II: In addition to the equations above, we have the new constraint:

$$y_0 \leq \frac{y_2 - y_1}{x_2 - x_1} x_0 + \frac{x_1 y_2 - x_2 y_1}{x_2 - x_1} \tag{9}$$

If we express everything in terms of x_1 as we did before we get:

$$(a_2 - a_1) x_1 \frac{A_4 - A_2 x_1}{A_1 x_1 - A_3} + (b_2 - a_1 x_0 + y_0) x_1 + (a_2 x_0 - b_1 - y_0) \frac{A_4 - A_2 x_1}{A_1 x_1 - A_3} - b_1 x_0 \geq 0 \tag{10}$$

which is a rational function with numerator a polynomial of degree two and the denominator a polynomial of degree one. Therefore the zeroes and the intervals of interest can be computed analytically. □

The perimeter optimization problem also generates two types of subproblem with these same constraints, but a different, and more complicated, objective function. As rewritten as a function of x_1, the roots of the first derivative cannot be found analytically; one of the classical methods for root finding from numerical analysis must be used. We assume that finding the roots of an equation by a numerical method takes $O(1)$ time.

We conclude that the continuous rotation of the triangle xyz produced as x is moving along the 'legal' portion of AB can be discretized into a finite number of problems discussed below each of which can be solved analytically in $O(1)$ time.

Lemma 4.4 *The maximum inscribed triangle in a fan-shaped polygon can be found in* $O(s_k^{p_i} + s_k^{p_{j+1}} + s_j^{p_{i+1}} + s_i^{p_j})$ *which is* $O(n)$ *where* n *is the number of vertices of the fan-shaped polygon.*

Since we decompose the simple polygon into at most $O(n^3)$ fan-shaped polygons, computing the maximum inscribed triangle in the simple polygon uses at most $O(n^4)$ time. Careful analysis produces a better bound.

Theorem 4.5 *The maximum triangle inscribed in a simple polygon* P *can be found in* $O(n^3)$ *arithmetic operations where* n *is the number of vertices of* P*. The space required is* $O(n)$*.*

Proof: The total time spent in the shortest path computation is $O(n^3)$ since the shortest path procedure is called $O(n^2)$ times; Each **if** statement takes $O(1)$ time since whether $F_{i,j,k}$ is legal can be decided from the shortest path computation; According to Lemma 4.4, the procedure *fan(i,j,k)* takes $O(s_j^{p_i} + s_i^{p_{j+1}} + s_k^{p_{i+1}} + s_k^{p_j})$. Thus the total time spent on the fan-shaped polygons corresponding to all triples (i, j, k) of edges of P is:

$O(\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} (s_j^{p_i} + s_i^{p_{j+1}} + s_k^{p_{i+1}} + s_k^{p_j}))$ which is $O(n^3)$ since $\sum_{k=1}^{n} s_k^v = O(n)$ for any vertex v of P [26]. □

An alternative way of proving the above bound is to prove the following combinatorial result of independent interest:

Lemma 4.6 *The sum of the sizes of all legal fan-shaped polygons with* 3 *bases of a simple polygon* P *is* $O(n^3)$*.*

Proof: Let f_{ijk} be the size of a legal 3-fan-shaped polygon defined by the edges i, j, k of a simple polygon P. Consider all pairs of hourglasses defined by the edges i, j, k. These hourglasses can be formulated by considering the common tangents of the chains of the fan-shaped polygons. Let h_{ij}, h_{kj}, h_{ik} be the sizes of the corresponding hourglasses. For our proof, $C_{p_i p_{j+1}}$ will represent both the chain $C_{p_i p_{j+1}}$ and its length. Consequently,

$h_{ij} = p_{i+1}a + 1 + p_j b + C_{p_i p_{j+1}}$ (1)
$h_{ik} = p_{k+1}d + 1 + p_i c + C_{p_k p_{i+1}}$ (2)
$h_{kj} = p_{j+1}f + 1 + p_k e + C_{p_j p_{k+1}}$ (3)
summing up (1), (2) and (3) produces
$h_{ij} + h_{ik} + h_{kj} = 3 + p_{i+1}a + p_j b + p_{k+1}d + p_i c + p_{j+1}f + p_k e + f_{ijk}$
which implies $f_{ijk} < h_{ij} + h_{ik} + h_{kj}$ (4)
Then

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} f_{ijk} < \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}(h_{ij} + h_{ik} + h_{kj})$$

.

This last summation is $O(n^3)$ since according to Lemma 3.3,

$$\sum_{i=1}^{n}\sum_{j=1}^{n} h_{ij} = O(n^2).$$

□

This lemma generalizes:

Lemma 4.7 *The sum of the sizes of all legal fan-shaped polygons with k bases of a simple polygon P is $O(n^k)$.*

Proof: By induction on k. □

The complexity of our method depends on two components: a) shortest path computation and b) computation done in all fan-shaped polygons. At all times, either we have computed the visible part of P from both edges i and j or we are in the process of computing it. In each of a) and b) we expend $O(n^3)$ time. Now we are going to see how the Hershberger technique [29] of creating shortest paths incrementally can profitably be combined with our method of the previous section.

First, we define some terms and notation. Let i and j be two polygon edges. The number of edges between i and j as we move clockwise from i to j is called the *distance* between i and j. Let D^i denote the symmetric difference of the shortest path maps from the endpoints of edge i. Also let d^i_j be the part of D^i which intersects edge j.

Assuming that the shortest path map from vertex p_1 has already been computed [26], Hershberger [29] shows that the shortest path map from p_2 can be computed incrementally by scanning the boundary of P in counterclockwise order starting from p_2 and going back to p_1, in time proportional to the symmetric difference of the two shortest path maps D^1. Consequently he proves that the total number of differences of all shortest path maps $\sum_{i=1}^{n} |D^i|$ is $O(m)$ where m is the size of the visibility graph of P. Additionally, at any moment his algorithm satisfies the following invariant: upon reaching a particular point x on the boundary of P, the shortest path map from p_1 within the area of P to the left of $SP_{p_1}(x)$ is known, and the shortest path map from p_2 within the area of P to the right of $SP_{p_2}(x)$ is known.

We use these results as follows. Assume that edges i and j have distance d. Assume also that we are given the shortest path maps from p_{j+1} and p_{i+1}. Move from p_j to p_i clockwise, using Hershberger's method to construct part of the shortest path map from p_j using the shortest path map from p_{j+1}. Then start scanning the boundary of P from p_i clockwise up to p_{j+1}. During this walk, we concurrently begin constructing the shortest path map from p_i using the one from p_{i+1} and finishing the one from p_j using the one from p_{j+1}. When we walk on edge k, we want to apply our method of the previous section on fan-shaped polygon f_{ijk} (see fig. 13). In order to do that, we need to know the elementary intervals for the continuous optimization problems, *i.e.* we need to know $S_{p_{j+1}}(k)$, $S_{p_j}(k)$, $S_{p_i}(k)$, $S_{p_{i+1}}(k)$. But the walk we described above will have produced these subdivisions. Specifically $S_{p_j}(k)$ can be derived from $S_{p_{j+1}}(k)$ and $S_{p_i}(k)$ from $S_{p_{i+1}}(k)$. Furthermore, we need to check the visibility of $y \in i$ and $z \in j$ using $S_{p_j}(i)$ and $S_{p_{i+1}}(j)$. But since we are on edge k which is "after" edge i and "before" edge j, both $S_{p_j}(i)$ and $S_{p_{i+1}}(j)$ are known. After a complete cycle on the boundary we have constructed the shortest path maps from p_i and p_j. Then we can proceed by advancing both edges i and j counterclockwise and repeating the same procedure. The above procedure suggests the following lemma:

Lemma 4.8 *The time spent in shortest path computations for fan-shaped polygons f_{ijk} when edges i and j have fixed distance is $O(n+m)$.*

Repeating the above algorithm for pairs i and j of all possible distances $1, ..., n$ yields:

Lemma 4.9 *The time spent for all shortest path computations is $O(n^2 + nm)$.*

The question which still remains is whether it is possible to reduce the total cost of the fan-shaped computations, i.e. the total number of continuous optimization problems we have to solve.

According to Lemma 4.4 we know that the time spent in a fan-shaped polygon f_{ijk} is $O(s_k^{p_i} + s_k^{p_{j+1}} + s_j^{p_{i+1}} + s_i^{p_j})$ and all these quantities sum up to $O(n^3)$ according to Theorem 4.5. But $|D^i| = \sum_{j=1}^{n} d_j^i$, and $\sum_{i=1}^{n} |D^i| = O(m)$.

Lemma 4.10 *The maximum inscribed triangle in a fan-shaped polygon can be found in $O(d_k^i + d_k^j + d_j^i + d_i^j)$.*

Lemma 4.11 *The total time spent in all fan-shaped polygons of a polygon P is $O(nm)$ where n, m is the number of vertices and the size of visibility graph of P respectively.*

Proof: According to *lemma* 4.10 the time spent in a fan-shaped polygon f_{ijk} is $O(d_k^i + d_k^j + d_j^i + d_i^j)$. Summing over all fanshaped polygons we get

$\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} (d_k^i + d_k^j + d_j^i + d_i^j)$

Since $\sum_{i=1}^{n} \sum_{j=1}^{n} d_j^i = O(m)$ then the above summation is $O(mn)$. □

Theorem 4.12 *The maximum triangle inscribed in a simple polygon P can be found in $O(n^2 + nm)$ arithmetic operations where n is the number of vertices of P and m is the size of the visibility graph. The space required is $O(n)$.* □

Unfortunately, the maximum area or perimeter triangle inscribed in a simple splinegon might not have two sides tangent to the chains of the fan-shaped splinegon, as was true in the polygon case. In fact, we present a construction where none of the sides of any maximum inscribed triangle is tangent to the fan-shaped polygon.

Construction: Consider two circles C_1 and C_2 with common center O and radii R_1 and R_2 respectively such that $R_1 < R_2$. Let xyz be an equilateral triangle inscribed in C_2. Let A, B, C, D, E, F be the intersection points of C_1 with the sides of $\triangle xyz$ as shown in fig.15. Let A', B' be points on C_1 counterclockwise and clockwise respectively from A, B. Similarly define points C', D', E', F', see fig.15. Define a convex curve segment with one endpoint A' the other endpoint B' so that lies entirely between C_1 and C_2 and intersects C_2 at a unique point x. Define similar curved segments with endpoints C', D' and E', F'. Define a concave curve segment between each pair of points (A', F'), (B', C'), (D', E') so that do not intersect $\triangle xyz$. The above six curved segments (three convex and three concave) define a splinegon S which lies entirely inside circle C_2 except for points x, y, z which lie on the boundary of C_2.

Claim: $\triangle xyz$ is the maximum area triangle inscribed into S.

Proof: By contradiction. Assume there exists a $\triangle abc$ such that $area(\triangle abc) > area(\triangle xyz)$. Then, at least one of the vertices a, b, c lies in the interior of C_2. But then there exists at least one triangle T inscribed in C_2 such that $area(\triangle abc) < area(\triangle T)$. It is well known, however, that a maximum area triangle inscribed in a circle is equilateral. Thus $area(\triangle T) \leq area(\triangle xyz)$ which implies that $area(\triangle abc) < area(\triangle xyz)$. □

In the polygon case, it was possible to reduce the number of triples of elementary segments considered within a single fan-shaped polygon to linear in the size of that polygon. In the splinegon case, all $O(n^3)$ triples of elementary segments must be considered.

Theorem 4.13 *The maximum area or perimeter triangle inscribed in a simple splinegon can be found in $O(n^4)$ time and $O(n)$ space.*

Proof: Move point x along the legal part of k. Each interval of this subdivision corresponds to specific anchors (although these anchors may be curved segments rather than points) of the shortest paths from p_i, p_{i+1}, p_j, p_{j+1} to x. The line through xy (xz) must have the anchors of x with respect to p_i and p_{i+1} (resp. p_j and p_{j+1}) in opposite sides in order to guarantee visibility of x, y (resp. x, z). To check the visibility of y and z, consider all pairs of segments on the subdivision of $p_i p_{i+1}$ and $p_j p_{j+1}$.

Each fan-shaped splinegon requires $O((s_k^{p_{i+1}} + s_k^{p_j} + s_k^{p_i} + s_k^{p_{j+1}}) s_i^{p_{j+1}} s_j^{p_i})$ time.

Since $s_k^{p_i} \leq n$, the total complexity is

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} ((s_k^{p_{i+1}} + s_k^{p_j} + s_k^{p_i} + s_k^{p_{j+1}}) s_i^{p_{j+1}} s_j^{p_i})$$

.

Since $s_k^{p_i} < n$, the above sum is $O(n^4)$. □

This result is significant in that it is the first instance in which there is an apparent asymptotic gap between polygon and splinegon solutions for the same problem.

We now move to the maximum constrained inscribed triangle problem.

Lemma 4.14 *The Maximum Inscribed Triangle with one of its sides having given length has at least one of the non-given length sides tangent to a fan-shaped polygon.*

Proof: Similar to that of Lemmas 4.1 and 4.3. □

Theorem 4.15 *For a simple polygon P of n vertices, the Maximum Inscribed Triangle with one of its sides having given length can be found in $O(n^3)$ time and $O(n)$ space.*

Proof: The algorithm for this problem uses techniques similar to those used to solve the unconstrained problems. As in the case of maximum area inscribed triangle, edge k is subdivided by the shortest path maps from the source points $p_{j+1}, p_j, p_{i+1}, p_i$ into elementary intervals such that for every point x in an elementary interval K, $anchor^{p_{j+1}}(x)$, $anchor^{p_j}(x)$, $anchor^{p_{i+1}}(x)$, $anchor^{p_i}(x)$ remain unchanged. As we did before instead of $anchor^{p_i}(x)$ we will refer to $anchor^{p_i}(K)$. Assume that triangle vertices x, y, z lie on edges k, i, j, respectively, and that yz is a non-fixed-length triangle side which remains tangent to $SP_{p_{j+1}}(p_i)$ (see fig. 14). The initial position of y coincides with the closest point to p_i which is visible from edge j. Rotate yz clockwise so that it remains always tangent to $SP_{p_{j+1}}(p_i)$. At any moment $anchor^{p_i}(z) = anchor^{p_{j+1}}(y)$. As in the case of edge k, both edges i and j are subdivided into elementary intervals of constant anchor. Thus let I (resp. J) be the elementary intervals where y (resp. z) belong. For every pair of elementary intervals I and J, consider all elementary intervals K of edge k. For each such triple (I, J, K), solve the following continuous constant size optimization problem in $O(1)$ time:

Problem: Given three line segments K, I, J and five fixed points A, B, C, D, E, compute the coordinates of points $x = (x_1, x_2)$ on K, $y = (y_1, y_2)$ on I, and $z = (z_1, z_2)$ on J such that: a) xy has constant length L, b) $\overrightarrow{xy}$ ($\overrightarrow{xz}$ resp.) keeps A and B (C and D resp.) on opposite halfplanes, c) $\overline{yz}$ contains E, and d) the area of xyz is maximized. (A, B, C, D represent $anchor^{p_i}(K), anchor^{p_{i+1}}(K), anchor^{p_j}(K), anchor^{p_{j+1}}(K)$ respectively, and E represents $anchor^{p_i}(J) = anchor^{p_{j+1}}(I)$.)

Solution: Since points x, y, z lie on given lines, x_2, y_2, z_2 are expressed in terms of x_1, y_1, z_1. Since yz passes through a given point E such that $E = anchor^{p_i}(J) = anchor^{p_{j+1}}(I)$ then z_1 is expressed in terms of y_1. Since xy has length L, y_1 can be expressed in terms of x_1. All of these substitutions produce a one variable optimization problem. Finally, we add the visibility constraints for the x, y, z described in part b) of the statement of the problem, generating four one-variable constraints.

Using the same notation as in the previous section, the number of elementary intervals K is $O((s_k^{p_{i+1}} + s_k^{p_i} + s_k^{p_j} + s_k^{p_{j+1}}))$ and the number of interval pairs (I, J) is $O(s_i^{p_{j+1}} + s_j^{p_i})$, making the complexity for the fan-shaped polygon $O((s_k^{p_{i+1}} + s_k^{p_i} + s_k^{p_j} + s_k^{p_{j+1}})(s_i^{p_{j+1}} + s_j^{p_i}))$. The complexity of the whole algorithm is:

$$\sum_{i=1}^{n}(O(n) + \sum_{j=1}^{n}(O(n) + \sum_{k=1}^{n}(O((s_k^{p_{i+1}} + s_k^{p_i} + s_k^{p_j} + s_k^{p_{j+1}})(s_i^{p_{j+1}} + s_j^{p_i})) = O(n^3)$$

□

Remark: For splinegons, as in the case of the maximum area inscribed triangle, we do not have the tangency property of at least one edge of the constrained inscribed triangle. It is not therefore hard to see that:

For a simple splinegon P of n vertices, the Maximum Inscribed Triangle with one of its sides having given length can be solved in $O(n^4)$ time and $O(n)$ space.

5 Minimum Area Concave Quadrilateral

In this section, we present algorithms to compute the minimum-area *non-degenerate* concave quadrilateral circumscribing a simple polygon P, if one exists. $ABCD$ will always represent a concave quadrilateral where C is the reflex vertex. We always seek *non-degenerate* quadrilaterals such that A, B, C, D are distinct points and no three of them are collinear.[2] $CH(P)$ represents the convex hull of a simple polygon P. Each simple polygonal or splinegonal region Q interior to $CH(P)$ but exterior to P is called a *pocket* of P. Each edge of $CH(P)$ which is not an edge of P is called *pseudoedge* of P. The following lemmas provide characterization of the minimum-area concave quadrilateral:

Lemma 5.1 *If $ABCD$ is a minimum-area concave quadrilateral containing a simple non-convex polygon or splinegon P then A, B, D are not in the interior of $CH(P)$, with each of AB and AD is tangent to $CH(P)$ at points k and l respectively. (AB (AD resp.) may contain a whole edge of $CH(P)$, not just a single point k, (l resp.)). (see fig. 16).*

Proof: By definition $CH(P)$ is the minimal convex set which contains P. Since $\triangle ABD$ is a convex object which contains P, implies that $CH(P) \subseteq \triangle ABD$. If either AB or AD is not tangent to P, then B can be moved, reducing the area of both ABD and $ABCD$. □

Lemma 5.2 *The reflex vertex C, of a minimum-area quadrilateral containing a simple non-convex polygon P, lies inside the visibility polygon VQ_i of some pocket Q_i of P with respect to pseudoedge v_iw_i.*

Proof: It suffices to prove that C lies in some pocket Q_i of P. If that is true then it is clear that C lies in the visibility polygon of Q_i with respect to pseudoedge v_iw_i, since both BC and DC intersect v_iw_i. Let $ABCD$ be a minimum non-degenerate quadrilateral which contains P and assume C is not inside $CH(P)$ (see fig.17). Then at least one of BC or CD contains no point of $CH(P)$. Rotate that edge around C by a small angle θ so that B or D, respectively, is closer to A. The new quadrilateral is smaller than the original, producing a contradiction. □

Lemma 5.3 *Sides BC, DC are tangent to the boundary of a pocket Q_i with pseudoedge v_iw_i at points a, b distinct from C where $a = anchor^{v_i}(C)$ and $b = anchor^{w_i}(C)$.*

Proof: Assume that C lies inside some pocket Q_i of P but that BC and DC are not tangent to the pocket boundary. C is visible within Q_i from v_iw_i, which implies that the shortest paths from v_i to C and from w_i to C inside Q_i are inward convex chains. Let a and b be the anchors of these two shortest paths. Assume that BC and DC do not pass through a and b respectively. Let B' (D' resp.) be the intersections of Ca and Cb with AB and AD respectively. Since B' (D' resp.) is between A and B (A and D resp.) then the area of $AB'CD'$ is less than the area of $ABCD$. □

Lemma 5.4 *If $ABCD$ is a minimum-area concave quadrilateral containing a simple polygon P, then the following hold:* [3]

a) The midpoint of AB (AD resp.) lies on $CH(P)$;

b) The midpoint of BC (DC resp.) either lies on Q_i or it lies between two distinct points of tangency on BC relative to Q_i.

[2] We exclude the collinear case since this reduces in finding the minimum area triangle containing a convex polygon, a problem solved in [37].

[3] The lemma does not necessarily hold for splinegons.

Proof:

a) According to *lemma* 5.1, AB is tangent to $CH(P)$. Let m (n resp.) be the common point of AB and $CH(P)$ closest to A (B resp.). (Note that m and n may be identical.) Assume that the midpoint of AB does not lie between m and n. Assume w.l.o.g. that $An < nB$. Then rotate AB counterclockwise around n by a very small angle θ. Let $A'B'$ be the new position of AB. By a continuity argument, $A'n < nB'$ and thus $area(\triangle AnA') < area(\triangle BnB')$, producing a contradiction to the fact that the area of $ABCD$ is minimum. (see fig. 18.)

b) According to *lemma* 5.3, BC is tangent to Q_i. Let m (n resp.) be the common point of BC and Q_i closest to B (C resp.). (Note that n and C may be identical, or that m and n may be identical.) Assume that the midpoint of BC does not lie between m and n. Assume w.l.o.g. that $Cm < mB$. By rotating BC clockwise around m by an infinitesimal angle θ to a new position $B'C'$ and using the same continuity argument as in a), we get that area of quadrilateral $AB'C'D$ is larger than the area of $ABCD$, producing a contradiction. (see fig. 19.)

□

Lemma 5.5 *A non-convex polygon P need not have a non-degenerate minimum-area concave quadrilateral.*

Proof: Consider the polygon P formed by taking a large equilateral triangle xyz of side length L, and cutting out a small equilateral notch from edge yz of side length ϵ to form a hexagon $uvwxyz$ with reflex angle at v (see fig. 20). Let $ABCD$ represent a minimum concave quadrilateral containing P with vertex C inside uvw, BC tangent to P at w and CD tangent to P at u. Clearly quadrilateral vertex A should lie above line yz. According to *lemma* 5.4, vertex B (D resp.) should lie in a half disk with center vertex u (w resp.) and radius ϵ. By choosing $\epsilon \ll L$, the tangents from B and D to the $CH(P)$ cannot intersect above line yz. Thus a minimum-area concave quadrilateral does nor exist. □

This characterization of the optimum concave quadrilateral leads to the following algorithm:

Compute $CH(P)$ and let $p =$ the number of pockets.
Triangulate P and all its pockets Q_i, for $i = 1...p$.
For all pairs k, l of vertices of $CH(P)$ do
  for $i = 1$ to p do begin
    Compute the visibility polygon VQ_i of Q_i from pseudo-edge v_iw_i and the
     shortest path maps inside VQ_i from both v_i and w_i [26].
    Merge those maps [27] and label each region
     (≤ 6 sides) with its anchors w.r.t. v_i and w_i.
    For every region R, call $pocket(a, b, k, l, i, R)$.
  end.
Report the optimum.

The procedure $pocket(a, b, k, l, i, R)$ must solve the following optimization problem:

> **Problem:** Construct the minimum area concave quadrilateral $ABCD$ such that a) AB and AD pass through given points k and l respectively, b) the slope of the line through AB (AD resp.) lies in the interval defined by the slopes of the lines which contain the edges of $CH(P)$ adjacent to k (l resp.) c) CB and CD pass through a and b respectively, and d) $C \in R$.

Solution: Let (a_1, a_2), (b_1, b_2), (k_1, k_2), (l_1, l_2) be the coordinates of points a, b, k, l (as defined previously) respectively. Let also (x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), be the coordinates of the vertices of the quadrilateral A, B, C, D respectively. Then the area of the quadrilateral is given by:

$$AREA = (x_1y_2 - x_2y_1) + (x_2y_3 - x_3y_2) + (x_3y_4 - x_4y_3) + (x_4y_1 - x_1y_4) \tag{11}$$

AB contains k:

$$x_1y_2 - x_2y_1 = k_1y_2 - k_1y_1 + k_2x_1 - k_2x_2 \tag{12}$$

BC contains a:

$$x_3y_2 - x_2y_3 = a_1y_2 - a_1y_3 + a_2x_3 - a_2x_2 \tag{13}$$

CD contains b:

$$x_4y_3 - x_3y_4 = b_1y_3 - b_1y_4 + b_2x_4 - b_2x_3 \tag{14}$$

AD contains l:

$$x_4y_1 - x_1y_4 = l_1y_1 - l_1y_4 + l_2x_4 - l_2x_1 \tag{15}$$

Substituting equations 12, 13, 14, 15 into equation 11 we get:

$$AREA = (l_1-k_1)y_1+(k_1+a_1)y_2+(b_1-a_1)y_3-(l_1+b_1)y_4+(k_2-l_2)x_1-(k_2+a_2)x_2+(a_2-b_2)x_3+(b_2+l_2)x_4 \tag{16}$$

which is linear in terms of the unkown coordinates of A, B, C, D.

Thus we have to solve a linear program in four variables which is subject to a constant number of linear constraints. That problem can be solved in $O(1)$ time. Thus, each pass through the loop above takes $O(n_i + k_i)$ time where n_i is the size of Q_i and k_i is the size of the subdivision created by merging the shortest path maps from v and w. Thus, if n_c is the number of vertices of $CH(P)$, n_p is the total number of vertices of all pockets of P, and k is the sum of the merged subdivisions over all pockets, then we have the following theorem:

Theorem 5.6 *The minimum concave quadrilateral that contains a simple polygon P can be found either in $O(n_c^2(n_p + k))$ time and $O(n + k)$ space or in $O(n_c^2 n_p^2)$ time and $O(n)$ space.*

Proof: Computing the visibility polygon and the shortest paths within a pocket can be done in $O(n)$ time [26]. Two convex plane subdivisions S_1 and S_2 with m and n vertices respectively, can be merged in $O(m + n + k)$ time and space where k is the size of the resultant subdivision. It should be clear that k can range from $O(m + n)$ up to $O(mn)$.

Instead of explicitly merging the two maps, however, we can take each pair (r_1, r_2) where r_1 (resp. r_2) is a region of the shortest path map from v_i (resp. w_i), and calculate the intersection region explicitly. For each such intersection region, call $pocket(a, b, k, l, i, R)$. Since every region of the shortest path map is a triangle, the intersection of two such regions has a constant number of sides. The space required is the space to keep the two shortest path maps, i.e. linear. □

The constraints of the linear program do not dictate that vertex C be reflex; the interior angle at C is constrained merely to be greater than or equal to 180 degrees. Thus our algorithm can return degenerate quadrilaterals as solutions. In computing the global minima, we keep the minimum valued quadrilateral of our local optima, degenerate or otherwise. In cases of a tie, we give precedence to a non-degenerate quadrilateral. If the total minimum is non-degenerate, it solves the global problem. If the total minimum is degenerate, then there is no minimum strictly concave circumscribed quadrilateral for P.

Remark: In case of splinegons a similar technique is applicable. The difference is that the elementary optimization problems are not linear programs any more since the sides of the quadrilateral do not pass through constant points and *lemma* 5.4 is not applicable any more. These elementary continuous optimization problems although are of constant number of constraints and constant number of variables are much more complicated.

6 Contained Triangles

The maximum area triangle $T = xyz$ contained in a simple polygon P may have $0, 1, 2$ or 3 vertices on the boundary of P. The case of 3 vertices of the triangle on the boundary of P corresponds to the maximum area inscribed triangle problem solved in a previous section. We focus here on what we call the 0-case, 1-case, and 2-case. To solve these three cases, we use the following lemma:

Lemma 6.1 *Let A and C (B and D resp.) be two points on $\overrightarrow{O}x$ ($\overrightarrow{O}y$ resp.) such that segments AB and CD intersect inside the wedge defined by Ox and Oy at point E. Let FG be a line segment through E with F (G resp.) between A and C (D and B resp.). The area of the triangle OFG is maximized when one of the following holds: $F = A$ and $G = B$; or $F = C$ and $G = D$ (see fig. 21a).*

Proof: Assume that the maximum occurs when F is between A and C. Assume that $FE < EG$. We can rotate FG by a small angle θ in position $F^{'}G^{'}$ such that $F^{'}E < EG^{'}$. That implies area of triangle $EGG^{'}$ is greater than the area of triangle $EFF^{'}$ which implies that area of OFG is less than the area of $OF^{'}G^{'}$ a contradiction. □

Corollary 6.2 *Let $\overrightarrow{O}x$ and $\overrightarrow{O}y$ be two rays with common origin O. Let also A and B be two points on $\overrightarrow{O}x$ and $\overrightarrow{O}y$ respectively and let C_{AB} be a convex chain as in fig. 21b. Let D (E resp.) lie on OA (OB resp.) such that O and C_{AB} do not lie on the same side of the line through D and E. Then the area of triangle ODE becomes maximum if DE contains an edge of the convex chain C_{AB}.*

Proof: Assume that DE does not contain any edge of C_{AB}. Then either a) the intersection of DE and C_{AB} is one vertex of C_{AB} or b) it is the empty set. In the case of a) we have an instance of Lemma 6.1. In the case of b) we can translate DE in a direction perpendicular to itself until it intersects C_{AB} and then apply Lemma 6.1. □

Lemma 6.3 *Let T be a maximum area triangle. Then each edge of T contains at least two points of the boundary of P. Specifically, the following conditions hold (see fig. 22): If T is of the 0-case, then each edge of T contains at least two reflex vertices of polygon P; if T is of the 1-case with x on the boundary of P, then yz touches at least two reflex vertices of P and xy and xz at least one; if T is of the 2-case with y and z on edges i and j, there exists at least one edge k of P such that i, j, k define a fan-shaped polygon $F_{i,j,k} \supseteq T$.*

Proof: Assume that there exists at least one side of the triangle xyz for which the above argument is not true. Without loss of generality, assume yz is that edge. Extend both xy and xz to the side of y and z respectively until they intersect the boundary of P. Let v and w be the two intersection points. Consider the convex hull of the part of P which is between v and w and inside the triangle xvw. Then applying the previous corollary, we can move yz so that the area of xyz increases. □

This characterization of the $0, 1, 2$-case maximum area triangle contained in a simple polygon suggests algorithms for finding these triangles.

0-Case Triangle Algorithm. One algorithm would consider all triples of pairs of reflex vertices i.e $O(n^6)$ objects, check whether the corresponding triangle is contained in P in $O(\log n)$ time using ray-shooting [13] or [26], and choose the largest one. This brute force approach requires $O(n^6 \log n)$ arithmetic operations. Another less naive algorithm would fix two sides of the candidate triangle by choosing a pair of pairs of reflex vertices (A, B) and (E, F), assuming that xy contains AB and xz contains EF, and spend linear time to find the optimum position of yz, for a total of $O(n^5)$ operations.

Using the linearity of shortest path trees inside simple polygons, we can reduce the complexity by an order of magnitude (see fig. 23). Fix a pair of reflex vertices C and D with the characteristic that $\overline{CD} \subset P$ and all edges incident to C and D lie on the same side of the line containing $\overline{CD}$. Assume that side yz contains these vertices. Determine in $O(\log n)$ time ([13] or [26]) the points G and H closest to C and D, respectively, where the line through segment CD intersects the boundary of P. Let $P^{'}$ represent the subpolygon of P which lies at the opposite side of GH from the edges incident to C and D. Since x must be visible from GH, the shortest paths from G to x and from H to x inside P must be inward convex chains containing segments AB and EF, respectively. That implies AB (EF resp.) are edges of the shortest path tree from G (H resp.) inside $P^{'}$. Since the sizes of the shortest path trees are linear in the size of $P^{'}$ and therefore in the size of P, we need consider only pairs of the $O(n)$ edges of the shortest path tree from G and the $O(n)$ edges of the shortest path tree from H, a total of $O(n^2)$ objects.

How can we test efficiently whether the chosen pair AB and EF of shortest path tree edges forms a legal triangle with GH? Choose only those pairs AB such that a) the shortest paths from G to A and G to B are inward convex chains, b) points y and z do not lie in segment CD, and c) segments Ax

and Bx lie inside P. We need the comparable conditions for EF. Conditions a) and b) clearly can be checked in $O(1)$ time. One way to test condition c) is to apply ray-shooting inside P in $O(\log n)$ time. Since AB and EF are shortest-path tree edges, however, constant time suffices. Define e_1 (resp. e_2) as the edge of the shortest path map from G which is adjacent to vertex A (resp. E) and collinear with AB (resp. EF). If e_1 and e_2 both exist and intersect, then the intersection point is a valid vertex x. Repeating the above procedure for every one of the $O(n^2)$ pairs of reflex vertices C and D yields the following lemma.

Lemma 6.4 *The 0-case maximum triangle can be found in $O(n^4)$ time and $O(n)$ space.*

1-Case triangle Algorithm. As in the 0-case algorithm, we fix a pair of reflex vertices C and D (see fig. 24). We find again points G and H as defined previously and then we have to walk on the shortest path maps of G and H along the boundary of P as we did in the inscribed triangle case of Section 4. Thus for a fixed pair of reflex vertices we spend, using similar arguments, $O(n)$ time and therefore a total $O(n^3)$ for the whole problem. Thus:

Lemma 6.5 *The 1-case maximum triangle can be found in $O(n^3)$ time and $O(n)$ space.*

2-Case Triangle Algorithm. According to Lemma 6.3, triangle xyz lies in a fan-shaped polygon where y (resp. z) lies on edge i (resp. j) (see fig. 25). Subdivide j (i resp.) according to the shortest path maps from both p_{i+1}, p_i and p_{k+1} (p_j, p_{j+1} and p_k resp.). For each interval on the subdivision of edge i and each interval of the subdivision of edge j, a) check whether points y and z are visible, using techniques developed in Section 4, b) let $a = anchor^{p_{k+1}}(z)$ and $b = anchor^{p_k}(y)$. Then, check whether $SP_{p_{k+1}}(a)$ and $SP_{p_k}(b)$ are inward convex. c) Check whether the intersection x of the lines through segments yb and za lies "below" the line through edge k. That guarantees that triangle xyz lies inside $F_{i,j,k}$. d) Solve the appropriate continuous optimization problem. It should be clear that steps a) through d) take $O(1)$ time.

According to the above discussion the time complexity per fan-shaped polygon is $O((s_j^{p_{i+1}}+s_j^{p_k})(s_i^{p_j}+s_i^{p_{k+1}}))$. Then summing over all fan-shaped polygons we get

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} O((s_j^{p_{i+1}}+s_j^{p_k})(s_i^{p_j}+s_i^{p_{k+1}})) = O(n^3)$$

Given that the total shortest path computation takes $O(n^4)$ time and $O(n)$ space we have the following lemma: □

Lemma 6.6 *The 2-case maximum triangle can be found in $O(n^4)$ time and $O(n)$ space.* □

Theorem 6.7 *The maximum area triangle contained in a simple polygon can be found in $O(n^4)$ time and $O(n)$ space.*

Unfortunately, solving this problem for splinegons seems complicated. All that we know is that each side of the maximum area triangle contained in a simple splinegon touches the splinegon boundary in at least one point. This constraint is not enough to produce anything other than a "brute force" algorithm.

7 Conclusions

We solved various geometric optimization problems using shortest paths within simple closed regions. There are several directions for further research. One is the obvious task of improving the current time bounds and extending our results in higher dimensions. Another question is whether shortest paths can be used in other inclusion, enclosure, or separator problems, or, more interestingly, whether they can be used in other classes of geometric optimization problems. A more exhaustive study of optimization over curved objects would also be appropriate.

Ackowledgements

The authors gratefully acknowledge the careful reading of the two anonymous referees and their many suggestions for improving the readability of the paper.

References

[1] A. Aggarwal, "Lecture notes in Computational Geometry", *MIT Research Seminar Series MIT/LCS/RSS 3*, August, 1988.

[2] A. Aggarwal, J. S. Chang and C. K. Yap, "Minimum Area Circumscribing Polygons", *Visual Computer*, 1 (1985), 112-117.

[3] A. Aggarwal, M. Klawe, S. Moran, P. Shor, R. Wilber, "Geometric Applications to a Matrix Searching Algorithm", *Algorithmica*, 2 (1987), 209-233.

[4] A. Aggarwal, J. Park, "Notes on Searching in Multidimensional Monotone Arrays", *Proc. 29th IEEE Symp. on Found. of Comp. Sci.* (1988), 497-512.

[5] J.E. Boyce, D.P. Dobkin, R.L. Drysdale, L.J. Guibas, "Finding Extremal Polygons", *SIAM J. of Computing*, 14 (1985), 134-147.

[6] R. D. Bourgin, S. E. Howe, "Algorithms for Shortest Curves in Planar Regions with Curved Boundary", *Manuscript*, (1989).

[7] R. D. Bourgin, M. S. Martin, P. L. Renz, "Shortest Curves in Jordan Regions Vary Continuously with the Boundary", to appear in *Advances in Mathematics.*

[8] R. D. Bourgin, P. L. Renz, "Shortest Paths in Simply Connected Regions in R^2", *Advances in Mathematics*, 76, No. 2, (1989), pp. 260-295.

[9] J. S. Chang, C. K. Yap, "A Polynomial Solution for Potato-Peeling and Other Polygon Inclusion and Enclosure Problems", *Discrete and Comp. Geom.*, 1 (1986), 155-182.

[10] J. S. Chang, "Polygon Optimization Problems", Ph.D. Thesis, New York University, 1986.

[11] B. Chazelle, "Triangulating a Simple Polygon in Linear Time", *Proc. 31st IEEE Symp. on Found. of Comp. Sci.*, (1990), 220-230.

[12] B. Chazelle, D. Dobkin, "Intersection of convex objects in two and three dimensions", *J. ACM*, 34 (1987), 1-27.

[13] B. Chazelle, L. Guibas, "Visibility and Intersection Problems in Plane Geometry", *Discrete and Comp. Geom.*, 4 (1989), 551-581.

[14] P. Chew, K. Kedem, "Placing the Largest Similar Copy of a Convex Polygon among Polygonal Obstacles", *Proc. of ACM Symp. Comp. Geom.* (1989), 167-74.

[15] N.A.A. DePano, "Polygon Approximation with Optimized Polygonal Enclosures: Applications and Algorithms", Ph.D. thesis, Dept of Computer Science, Johns Hopkins University, April 1988.

[16] N.A. DePano, Yan Ke, J. O'Rourke, "Finding Largest Inscribed Equilateral Triangles and Squares", *Proc. of the Allerton Conference*, 1987.

[17] D. Dobkin, L. Snyder, "On a General Method for Maximizing and Minimizing among Certain Geometric Problems", *Proc. 20th IEEE Symp. on Found. of Comp. Sci.* (1979), 9-17.

[18] D. Dobkin, D. Souvaine, "Computational Geometry in a Curved World", *Algorithmica*, 5 (1990), 421-457.

[19] D. Dobkin, D. Souvaine, C. Van Wyk, "Decomposition and intersection of splinegons", *Algorithmica,* 3 (1988), 473-485.

[20] H. Edelsbrunner, L. Guibas, G. Stolfi, "Optimal point location in monotone subdivisions", *SIAM J. Comput.*, 15 (1986), 317-340.

[21] R. Fleischer, K. Mehlhorn, G. Rote, E. Welzl, C. Yap, "On Simultaneous Inner and Outer Approximation of Shapes," *Proc. of the 6th ACM Symp. on Comp. Geometry*, June 1990, 216-224.

[22] S. Fortune, "A Fast Algorithm For Polygon Containment By Translation", *Proc. 13th ICALP* (1985), 189-198.

[23] A. Fournier and D. Y. Montuno, "Triangulating Simple Polygons and Equivalent Problems", *ACM Transactions on Graphics*, 3 (1984), 153-74.

[24] M. Garey, D. Johnson, F. Preparata, R. Tarjan, "Triangulation of a simple polygon", *Information Processing Letters*, 7 (1978), 175-179.

[25] L. Guibas, J. Hershberger, "Optimal shortest path queries in a simple polygon", Proc. ACM Symp. on Comp. Geometry, (1987).

[26] L. Guibas, J. Hershberger, D. Leven, M. Sharir, R. Tarjan, "Linear Time Algorithms for Visibility and Shortest Path Problems inside Triangulated Simple Polygons", *Algorithmica*, 2 (1987), 209-233.

[27] L. Guibas and R. Seidel, "Computing Convolutions via Reciprocal Search", *Discrete and Computational Geometry*, 2 (1988), 175-193.

[28] L. Guibas and J. Stolfi, "Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams, " *ACM Trans. Graphics*, 4 (1985), 74-123.

[29] J. Hershberger, "An Optimal Visibility Graph Algorithm for Triangulated Simple polygons", *Algorithmica*, 4 (1989), 141-155.

[30] V. Klee and M. Laskowski, "Finding the Smallest Triangles Containing a Given Convex Polygon", *J. of Algorithms*, 6 (1985), 359-375.

[31] D. T. Lee and F. P. Preparata, "Euclidean Shortest Paths in the Presence of rectilinear barriers", *Networks*, 14, No. 3 (1984), pp. 393-410.

[32] B. Lisper, TheoryNet posting and followup communication, July, 1988.

[33] E. A. Melissaratos, *Ph.D Thesis* in preparation, Rutgers University (1991).

[34] E. A. Melissaratos and D. L. Souvaine, "On Solving Geometric Optimization Problems Using Shortest Paths," *Proc. of the 6th ACM Symp. on Comp. Geometry*, June 1990, 350-359.

[35] E. A. Melissaratos and D. L. Souvaine, "Shortest Paths, Visibility, and Optimization Problems in Planar Curvilinear Objects," *Proc. of the 2nd Canadian Conf. on Comp. Geometry*, August 1990, pp. 337-342.

[36] J. D. Mittleman and D. L. Souvaine, "Shortest Area-Bisector of a Convex Polygon", Rutgers University Technical Report LCSR-TR-139, November 1989.

[37] J. O' Rourke, A. Aggarwal, S. Maddila, M. Baldwin, "An Optimal Algorithm for Finding Minimal Enclosing Triangles", *J. Algorithms*, 7 (1986), 258-269.

[38] M. H. Overmars and J. van Leeuwen, "Maintenance of Configurations in the Plane", *J. Comput. System Sci.*, 23 (1981), 166-204.

[39] O. Schwarzkopf, U. Fuchs, G. Rote, E. Welzl, "Approximation of Convex Figures by Pairs of Rectangles", *Proc. of Symp. on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science* 415 (1990), 240-249.

[40] A. Shaffer and C. J. Van Wyk, "Convex hulls of piecewise-smooth Jordan curves", *J. Algorithms*, 8 (1987), 66-94.

[41] D. L. Souvaine, "Computational Geometry in a Curved World." Ph.D. Thesis, Princeton University, October, 1986.

[42] R.E.Tarjan and C. Van Wyk, "Triangulation of a Simple Polygon", *SIAM Journal of Computing*, 17 (1988), 143-178.

Figure 1: a) Shortest-Path Tree; b) Shortest-Path Map

Figure 2: a) Type 1; b) Type 2; c) Type 3.

Figure 3: Refinement of degenerate trapezoids: a) Type 1; b) and c) Type 2; d) Type 3.

Figure 4: Funnel vertices between A and t_1 and vertices between t_2 and B will not be used again by the algorithm.

Figure 5: A trapezoidal component with many splinegon vertices interior to the parallel sides.

Figure 6: Several cases of funnel splitting.

Figure 7: Any arbitrary tree may be converted to a binary tree.

Figure 8: Shortest paths between endpoints are not inward convex chains.

Figure 9: A polygon P and an area-separator xy.

Figure 10: Hourglass between edges i and j and the inner common tangents ad and bc.

Figure 11: A fan-shaped polygon F inside a simple polygon P

Figure 12: The maximum triangle has two sides tangent to the inward convex chains.

Figure 13: Modified fan-shaped algorithm.

Figure 14: Triangle side xy is of constant length and yz remains tangent to the convex chain from p_{j+1} to p_i.

Figure 15: A maximum area or perimeter triangle inscribed in a simple splinegon might not have two sides tangent to the chains of the fan-shaped splinegon, as was true in the polygon case.

Figure 16: The minimum-area concave quadrilateral containing P.

Figure 17: Reflex vertex C should lie in the interior of $CH(P)$

Figure 18: The midpoints of AB and AD should lie on $CH(P)$

Figure 19: The midpoint of BC should lie between m and n

Figure 20: A case where an optimum non-degenerate quadrilateral does not exist

Figure 21: a) Figure for Lemma 6.1; b) Figure for Corollary 6.2.

Figure 22: a) 0-case; b) 1-case; c) 2-case; d) 3-case

Figure 23: 0-case

Figure 24: 1-case

Figure 25: 2-case