Discrete Comput Geom 14:445–462 (1995)

Discrete & Computational
Geometry

Efficient Piecewise-Linear Function Approximation Using the Uniform Metric*

M. T. Goodrich

Department of Computer Science, Johns Hopkins University,
Baltimore, MD 21218, USA
goodrich@cs.jhu.edu

Abstract. We give an $O(n \log n)$-time method for finding a best k-link piecewise-linear function approximating an n-point planar point set using the well-known uniform metric to measure the error, $\varepsilon \geq 0$, of the approximation. Our method is based upon new characterizations of such functions, which we exploit to design an efficient algorithm using a plane sweep in "ε space" followed by several applications of the parametric-searching technique. The previous best running time for this problem was $O(n^2)$.

1. Introduction

Given a set $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, the problem of approximating S by a function is classic in applied mathematics, and it finds applications in a number of computational problems. The general goals in this area of research are to find a function F belonging to a class of functions $\mathcal{F}$ such that each $F \in \mathcal{F}$ is simple to describe, represent, and compute and such that the chosen F approximates S well. For example, it may be desired that $\mathcal{F}$ be the class of linear or piecewise-linear functions, and, for any particular $F \in \mathcal{F}$, that the measure of the error be the well-known *uniform* metric

$$\|S - F\|_\infty = \max_{i \in \{1,2,\ldots,n\}} |y_i - F(x_i)|,$$

which is also known as the l_∞ or *Chebychev* measure of error [16], [18], [30]. The

* This research was announced in preliminary form at the 10th ACM Symposium on Computational Geometry. The author was partially supported by the NSF and DARPA under Grant CCR-8908092, and by the NSF under Grants IRI-9116843 and CCR-9300079.

goal, then, is to determine the value of

$$\varepsilon^* = \min_{F \in \mathcal{F}} \|S - F\|_\infty,$$

and find an $F \in \mathcal{F}$ achieving this error bound.

The version of this problem we address in this paper is to find a function $F \in \mathcal{F}$ that minimizes the uniform error term with respect to S where $\mathcal{F}$ is the class of k-link piecewise-linear functions, for some given $k \in \{1, 2, \ldots, n-1\}$. Using terminology from the approximation theory literature (e.g., see [9], [16], [18], and [19]), this is equivalent to the problem of finding a best $(k+1)$-knot degree-1 spline approximating S under the l_∞ norm. Of course, the case $k = n - 1$ is trivial, and there is a simple reduction of the case $k = 1$ to three-dimensional linear programming, which can be solved in $O(n)$ time [12], [20], [40], [42], [43], [53]. Thus, the interesting cases are for $1 < k < n - 1$. We show how to solve this problem for any such k in $O(n \log n)$ time.

The motivation for this problem is that one may have limited resources with which to describe the set S, but one wishes the best approximation possible within the given resource bounds. This can also be viewed as a data compression problem.

1.1. *Previous Work*

The problem we address is a special case of a whole class of problems in approximation theory where it is wished to fit a set of data using a spline function under some metric. Thus, the interested reader is referred to texts discussing approximation theory, such as those by Bellman and Roth [9], Conte and de Boor [16], Davis [18], and Dierckx [19], for a general treatment of such problems. Research in this literature is primarily interested in minimizing the number of knots in a spline under the least-squares metric, e.g., Jupp [36] gives a numerical approach to this problem. For the specific problem we address here, Bellman and Roth [8] describe a dynamic-programming approach based upon using a uniform grid to determine possible placements of link endpoints (which they call *knots*). Their method is not guaranteed to find a best k-kink approximation, however.

Hakimi and Schmeichel [30] show that such a best approximation can be found in $O(n^2 \log n)$ time, and this is the first method we know of that is guaranteed to find a best approximation. Their algorithm is based upon a clever lemma that shows that one can limit the number of "critical" ε values that are candidates for ε^* to be $O(n^2)$. They also show that one can test if any such ε value is equal to ε^* in $O(n)$ time, which implies that, once enumerated and sorted, one can perform a "binary search" among these critical values to find ε^*. Of course, enumerating these critical ε's requires $\Omega(n^2)$ time. Indeed, a straightforward application of the lemma by Hakimi and Schmeichel would require $O(n^3)$ time to enumerate them. They reduce the time to $O(n^2 \log n)$ using the powerful *plane-sweeping* technique (e.g., see [51]), which involves "sweeping" the plane with a line L while maintaining appropriate data structures for the points L encounters along the way. More recently, Wang

et al. [58] show how to use an even more clever plane-sweep procedure to find a best k-link approximation under the uniform metric in $O(n^2)$ time.

1.2. *Related Work*

With the exception of the papers by Hakimi and Schmeichel [30] and Wang *et al.* [58], related work in the computational geometry literature has been directed at what can be viewed as the "inverse problem," which is also addressed in the paper by Hakimi and Schmeichel [30]. In this problem one is given an error measure $\varepsilon \geq 0$ and asked to find a minimum-link polygonal path (which may or may not be required to be a function) that has distance at most ε from all the objects in S (which need not just be points), under some reasonable distance metric. As mentioned earlier, for the case when S is a set of points and the error measure is the uniform metric, then Hakimi and Schmeichel show that this problem can be solved in $O(n)$ time. Their method can be viewed as an extension of the linear-time method of Suri [54], which computes a minimum-link path inside a simple polygon, to the problem of finding a minimum-link monotone polygonal chain that "stabs" a given set of line segments. Hershberger and Snoeyink [32] show how to generalize this method further to find in $O(n)$ time a minimum-link path of a particular homotopy type in a nonsimple polygon, and Guibas *et al.* [29] show how to generalize this method even further to find in $O(n)$ time a minimum-link stabber for any given set of disjoint convex objects that must be stabbed in some given order (not necessarily just by increasing x-coordinates). Robert and Toussaint [52] study the problem of finding a line L that minimizes a weighted minmax error measure to a set of convex polygons in $O(n^2 \log n)$ time.

There has also been a considerable amount of work on finding a minimum-link approximation to a polygonal curve, subject to some error tolerance. The problem of fitting a minimum-link convex polygon nested between two given polygons was studied by Aggarwal *et al.* [5], who give an $O(n \log n)$-time solution to this problem. In addition, Imai and Iri [34], [35] give an $O(n)$-time method for finding the minimum-link function approximating a given monotone chain. Their method is very similar to an $O(n)$-time method independently discovered by Suri [54], [55] for solving the more-general problem of finding a minimum link path joining two points inside a simple polygon. There has also been some work on approximations that are required to use a subset of the endpoints of the given polygonal chain. For example, using an approach of Imai and Iri [33], [35], Toussaint [57] and Melkman and O'Rourke [44] give several $O(n^2 \log n)$-time methods under various metrics.

There has not been much work on a three-dimensional version of these approximation problems with guaranteed performance bounds, however, although the recent work by Mitchell and Suri [49] on a special case of the three-dimensional function approximation problem is a notable exception.

There is also a rich literature that studies minimum-link distance as a metric in its own right (e.g., see [6], [23], [24], [37], [39], [47], [48], [54], and [55]).

1.3. *Our Results*

As mentioned above, we give an $O(n \log n)$-time algorithm for finding a best k-link piecewise-linear function approximating a set S of n points in the plane under the uniform metric. Our method is based upon new geometric insights that allow us to apply a novel plane sweep in "ε space" to enumerate a set of $O(n)$ critical ε values, which we then search in a binary-search fashion. This allows us to restrict the range of ε values containing ε^* to be an interval $[\varepsilon_1, \varepsilon_2]$, but it does not necessarily give us $\varepsilon_1 = \varepsilon_2 = \varepsilon^*$. To achieve this latter result we give additional geometric characterizations of a best k-link approximation that allow us to follow this preprocessing step by several applications of pipelined versions of the well-known *parametric-searching* technique (e.g., see [2]–[4], [11], [13]–[15], and [41]).

Admittedly, the use of this technique typically makes an algorithm rather impractical to implement. However, we show that this is not true in our case, for we can design a relatively simple version of our algorithm that uses only the most simple versions of parametric searching (which can be made even more practically efficient via randomization).

In the section that follows we give some properties of a best k-link approximation and in Section 3 we show how to exploit these properties to restrict the range of candidate ε values. We then show how to complete the construction in Section 4 by relying on additional geometric properties of a best k-link approximation, which we show can be exploited in a series of applications of parametric searching. Finally, we show how to simplify our implementation in Section 5.

2. Some Properties of a Best k-Link Approximation

Let $S = (p_1, p_2, \ldots, p_n)$ be a left-to-right ordered listing of the points in S and let $\varepsilon \geq 0$ be given. So as to define our approximation problem formally and to articulate some of its important properties, we introduce some additional notation. For each point $p_i = (x_i, y_i)$ in S define $u_i = (x_i, y_i + \varepsilon)$ and $g_i = (x_i, y_i - \varepsilon)$, and let $S(\varepsilon)$ denote the ordered set of vertical segments $(\overline{u_1 g_1}, \overline{u_2 g_2}, \ldots, \overline{u_n g_n})$. Thus, if we view points as degenerate segments, then $S = S(0)$. For any ordered set of disjoint geometric objects A, a polygonal chain C is an *ordered stabber* if a traversal of C intersects the objects of A in the given order [29]. Finally, define $F(\varepsilon)$ to be a minimum-link ordered stabber of $S(\varepsilon)$.

The formal problem we address in this paper, then, is to find ε^*, the smallest $\varepsilon \geq 0$ such that $F(\varepsilon)$ has at most k links. Formulating the problem in this way allows us to deal with "degenerate" inputs, such as the one illustrated in Fig. 1, where $S(\varepsilon^*)$ may allow an ordered stabber with $k' < k$ links, but a minimum-link stabber of $S(\hat{\varepsilon})$ may require $\hat{k} > k$ links for any $\hat{\varepsilon} < \varepsilon^*$. Thus, a best k-link approximation $F = F(\varepsilon^*)$ may, in fact, have fewer than k links because of degeneracies. Of course, "dummy" vertices along F can always be introduced to force its link-count to be exactly k in such a case.

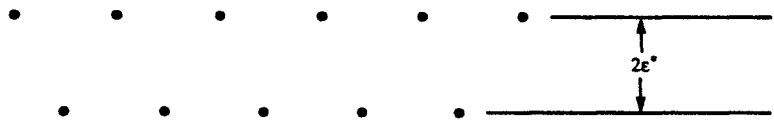

Fig. 1. An example set S such that $S(\varepsilon^*)$ has a one-link ordered stabber, but any ordered stabber of $S(\hat{\varepsilon})$ requires 10 links if $\hat{\varepsilon} < \varepsilon^*$.

2.1. *A Canonical Form for Best k-Link Approximations*

We connect consecutive g_i's and u_i's so as to form two "parallel" monotone chains $U(\varepsilon)$ and $G(\varepsilon)$, with $U(\varepsilon)$ being the upper chain, i.e., we create edges $\overline{g_i g_{i+1}}$ defining $G(\varepsilon)$ and $\overline{u_i u_{i+1}}$ defining $U(\varepsilon)$ for $i \in \{1, 2, \ldots, n-1\}$. One might be tempted to think that a best k-link function F approximating S can be constrained to lie between $U(\varepsilon^*)$ and $G(\varepsilon^*)$, but this is not the case[1] (as shown in Fig. 2). This is actually a good thing, for otherwise we would run into some robustness difficulties, for we would have to use a method for finding a minimum-link path in a simple polygon as a subroutine, and, as Snoeyink observers,[2] the bit complexity for finding such a path can be significantly larger than the bit complexity for representing the vertices of the input polygon. This is not a problem for our method, however, for we

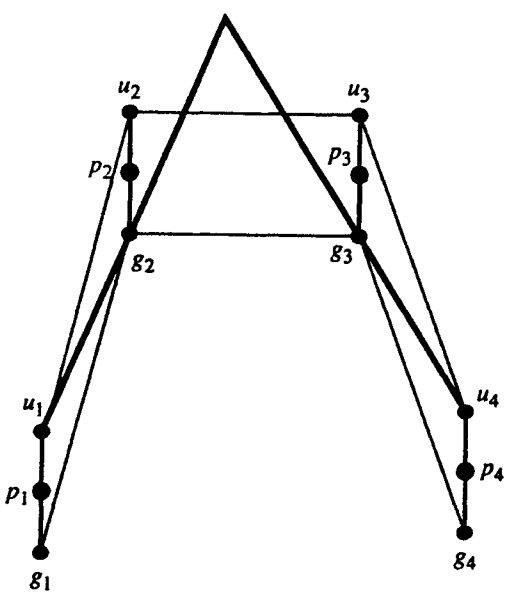

Fig. 2. An instance where a minimum-link stabber of $S(\varepsilon)$ is not confined to lie between $U(\varepsilon)$ and $G(\varepsilon)$.

[1] We are indebted to Jack Snoeyink (personal communication) for pointing this example out to us.

[2] Again, by a personal communication.

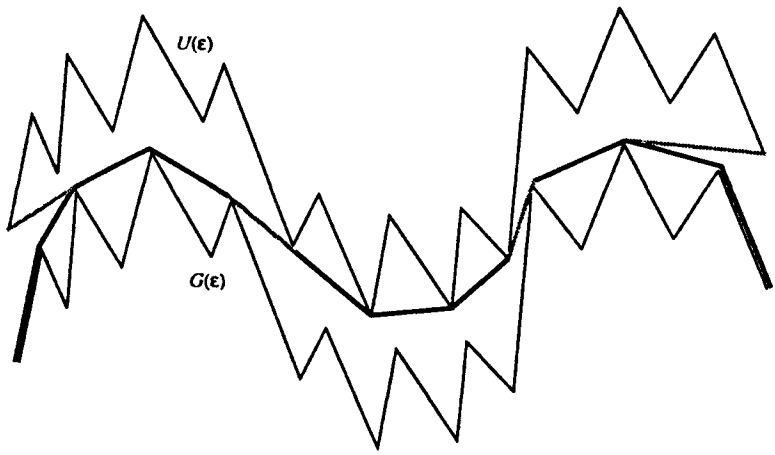

Fig. 3. An example hourglass. The inflection edges are shaded.

use methods for finding minimum-link stabbers as subroutines in our algorithm, and these methods do not suffer from this bit complexity blow-up difficulty.

To describe why we can use minimum-link stabber methods as subroutines, we must show how to restrict F to a certain canonical form. For a given ε, let $\pi_u(\varepsilon)$ be the shortest path from u_1 to u_n that does not go above $U(\varepsilon)$ and does not cross $G(\varepsilon)$. Similarly, let $\pi_g(\varepsilon)$ be the shortest path from g_1 to g_n that does not go below $G(\varepsilon)$ and does not cross $U(\varepsilon)$. Such paths were introduced by Lee and Preparata [38] and are often referred to as *geodesic* paths [1], [7], [10], [25], [28], [45], [46], [56]. The union of two such paths is often called an *hourglass* [22], [26]. We therefore use $H(\varepsilon)$ to denote this hourglass $\pi_u(\varepsilon) \cup \pi_g(\varepsilon)$. We say that an edge of $H(\varepsilon)$ is an *inflection* edge if one of its endpoints lies on $U(\varepsilon)$ while its other endpoint lies on $G(\varepsilon)$. Let $I(\varepsilon)$ denote the set of all such inflection edges. (See Fig. 3.)

We say that two consecutive links $\overline{pq}$ and $\overline{qr}$ in F have a *zig* turn type if r is above the ray $\overrightarrow{pq}$ (i.e., $\overline{pq}$ and $\overline{qr}$ form a "left turn"). Similarly, two consecutive links $\overline{pq}$ and $\overline{qr}$ in F have a *zag* turn type if r is below the ray $\overrightarrow{pq}$. This allows us to characterize each link in F, other than the first and last links, by the turn types they form with their predecessor and successor links. For example, a zig-zag link forms a left turn with its predecessor and a right turn with its successor. The next lemma establishes an important relationship between such links and inflection edges in $I(\varepsilon)$.

Lemma 2.1. *There is a best k-link function F approximating S such that*:

1. *Each $e \in I(\varepsilon^*)$ is contained by the first or last link of F or by a zig-zag or zag-zig link of F.*
2. *The first and last link of F, as well as each zig-zag and zag-zig link of F, contains an $e \in I(\varepsilon^*)$.*

Proof. 1. Suppose $e \in I(\varepsilon^*)$, i.e., e is an inflection edge of $H(\varepsilon^*)$. Also suppose, for the sake of contradiction, that e is contained in no link of the appropriate type in

any best k-link approximation F. We follow a proof technique of Ghosh [23], which involves performing local perturbations of a candidate stabber, to derive a contradiction. Since e is an inflection edge, it connects a u_i to a g_j; hence, F must intersect e along some link λ, for F cannot go above any u_i nor below any g_j. We assume for the time being that λ is neither the first nor last link of F, and we let λ^- denote λ's predecessor and λ^+ denote λ's successor. Also, let $L(e)$ denote the line containing e and let B denote the set of all points on some segment from a point p on λ^-, λ, or λ^+ to p's nearest neighbor on $L(e)$, i.e., the points "between" $L(e)$ and λ^-, λ, and λ^+. B can contain no points of $U(\varepsilon^*)$ nor $G(\varepsilon^*)$, for if this were not the case, then e would not be an inflection edge (e.g., see Fig. 4(a)). Thus, we can "move" the common endpoint of λ^- and λ to be on $L(e)$, and the common endpoint of λ and λ^+ to be on $L(e)$, keeping the rest of F fixed, and we keep F as a k-link approximation to S. To establish that λ must be a zig-zag or zag-zig link, note that the first place where the ray $\overrightarrow{u_i g_j}$ crosses $G(\varepsilon^*)$ cannot be before the first place it crosses $U(\varepsilon^*)$, and, likewise, the first place where the ray $\overrightarrow{g_j u_i}$ crosses $U(\varepsilon^*)$ cannot be before the first place it crosses $G(\varepsilon^*)$ (e.g., see Fig. 4(a)). If this were not so, $\overline{u_i g_j}$ would not be an inflection edge of $H(\varepsilon^*)$. Thus, either λ is the first or last link of F (which occurs if one of the rays $\overrightarrow{u_i g_j}$ or $\overrightarrow{g_j u_i}$ crosses neither $U(\varepsilon^*)$ nor $G(\varepsilon^*)$) or λ is a zig-zag or zag-zig link (since F is a minimum-link approximation to $S(\varepsilon^*)$). Similar (actually simpler) arguments hold for the cases when λ^- or λ^+ do not exist, and are left to the reader. Therefore, there is a best k-link approximation to S that contains each edge in S, and each such edge is contained in the first or last link of F or in a zig-zag or zag-zig link.

2. For the second part of the lemma, let m be the minimum number of zig-zag, zag-zig, first, and last links that do not contain any edge in $I(\varepsilon^*)$, taken over all best k-link approximations to S satisfying part one of the lemma (which we have just shown to be true). In addition, let $\lambda = \overline{pq}$ be one of these m links. Assume for the time being that λ is zig-zag link in F. Let $\lambda^- = \overline{rp}$ denote the predecessor of λ in F and let $\lambda^+ = \overline{qs}$ denote the successor of λ in F. Since F is a minimum-link path, the line $\overline{rs}$ must intersect both $U(\varepsilon^*)$ and $G(\varepsilon^*)$ (e.g., see Fig. 4(b)). However, this implies that F crosses an inflection edge, which is a contradiction; hence, we establish the second part of the lemma for this case. The proofs for the other cases are similar; hence, this establishes the lemma. □

Having established an important property of some of the links in a best k-link function approximation to S, we now turn to the problem of enumerating these edges, and in the process we also restrict the range of ε's that allow a k-link approximation.

3. Finding the Inflection Edges

We say that an ε is *geodesic-critical* if $H(\varepsilon)$ has l edges, but $H(\hat{\varepsilon})$ has fewer than l edges for $\hat{\varepsilon} > \varepsilon$. Our method for finding all the inflection edges is to determine an interval $[\varepsilon_1, \varepsilon_2]$ that contains ε^* and is such that $H(\varepsilon_1)$ is combinatorially equivalent to $H(\varepsilon_2)$. This allows us to determine all inflection edges that F must contain.

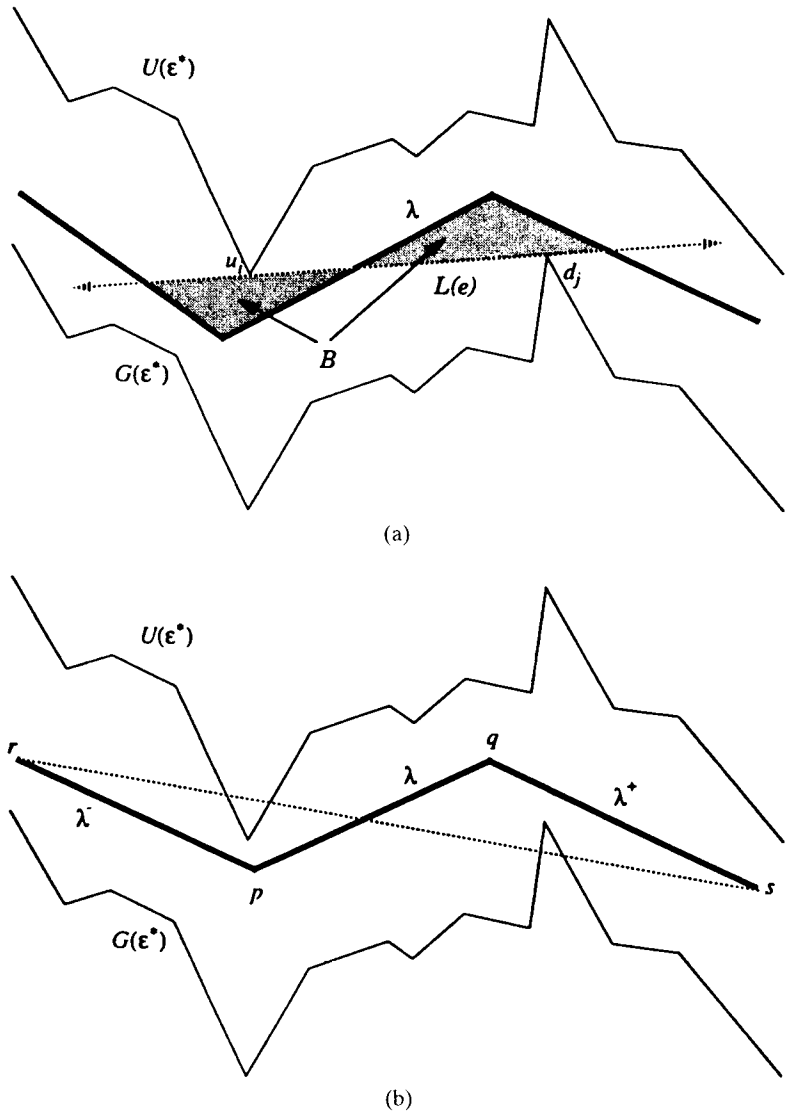

Fig. 4. Example zig-zag edges. In (a) we illustrate why an inflection edge is contained in a zig-zag link, and in (b) we illustrate why a zig-zag link contains an inflection edge.

Our procedure is conceptually quite simple. First we enumerate all $O(n)$ geodesic-critical ε values, and then we perform a binary search among these values to determine the interval $[\varepsilon_1, \varepsilon_2]$ containing ε^*.

3.1. *Enumerating all Geodesic-Critical ε Values*

Our method for enumerating all geodesic-critical ε values is based upon a sweep through "ε space." We maintain the hourglass $H(\varepsilon) = \pi_u(\varepsilon) \cup \pi_g(\varepsilon)$ while taking ε from 0 to $+\infty$, stopping at each geodesic-critical ε value along the way. To simplify the discussion, however, we concentrate on the problem of maintaining $\pi_u(\varepsilon)$, so that we restrict our notion of geodesic-critical ε's to those that change $\pi_u(\varepsilon)$; the method for maintaining $\pi_g(\varepsilon)$ is similar. Initially, for $\varepsilon = 0$, $\pi_u(\varepsilon)$ is the chain $U(\varepsilon) = G(\varepsilon)$; hence, it consists of $n - 1$ edges. If we then increase ε by an "infinitesimal" amount we find that some of the vertices of $\pi_u(\varepsilon)$ lie on $U(\varepsilon)$ while others lie on $G(\varepsilon)$. For any vertex p on $\pi_u(\varepsilon)$, if one of p's adjacent vertices on $\pi_u(\varepsilon)$ lies on the chain opposite from the chain that p lies on, then we say that p is a *pinch* vertex. For each pinch vertex p on $\pi_u(\varepsilon)$, let q and r denote p's adjacent vertices on $\pi_u(\varepsilon)$, and compute the $\hat{\varepsilon} > \varepsilon$ value at which p would cease to be a pinch vertex if we were to restrict $U(\hat{\varepsilon})$ and $G(\hat{\varepsilon})$ to that portion of the plane bounded by the lines $x = x(q)$ and $x = x(r)$ inclusive, where $x(t)$ denotes the x-coordinate of a point t. Call this $\hat{\varepsilon}$ value *locally critical* for p, and let E be the set of all $\hat{\varepsilon}$'s that are locally critical for pinch vertices on $\pi_u(\varepsilon)$.

Lemma 3.1. *The smallest $\hat{\varepsilon}$ in E is the smallest geodesically critical value bigger than ε.*

Proof. Let ε' be the smallest geodesically critical value bigger than ε. If we were to increase ε' "infinitesimally," then, by definition, $\pi_u(\varepsilon')$ would have at least one fewer edge. For this to occur, two consecutive edges of $\pi_u(\varepsilon')$ would have to be replaced by a single edge. Thus, the vertex incident upon the two removed edges is a pinch vertex; hence ε' is in E. Now, let ε'' be the smallest value in E. Since there is no value in E smaller than ε'' in E, $\pi_u(\hat{\varepsilon})$ does not change for $\hat{\varepsilon} \in (\varepsilon, \varepsilon'']$. Thus, there is a global change to $\pi_u(\hat{\varepsilon})$ for $\hat{\varepsilon} > \varepsilon''$, which implies that ε'' is geodesically critical. Therefore, $\varepsilon' = \varepsilon''$, which completes the proof. □

Our method for maintaining $\pi_u(\varepsilon)$, then, is as follows. We store the values belonging to E in a priority queue that supports the operations of insert, delete, and extract-min in $O(\log n)$ time (e.g., see [17]). While E is not empty, we extract the smallest $\hat{\varepsilon}$ in E, perform the modification of $\pi_u(\hat{\varepsilon})$ implied by this geodesically critical value, and then update E to reflect the new geodesic path. This update involves examining the two vertices of $\pi_u(\hat{\varepsilon})$ that now become adjacent and updating E accordingly. If either of them were previously pinch vertices, then we remove its corresponding locally critical value from E. Likewise, if either of them becomes a pinch vertex after performing the update for $\hat{\varepsilon}$, then we insert its new locally critical value into E. Since we reduce by one the number of edges of the geodesic path with

each event in E, the total number of events must be $O(n)$; hence, the total time to enumerate all the geodesically critical values is $O(n \log n)$.

Given these geodesically critical values it is then a simple manner to determine the consecutive pair that contains ε^* by using the method of Hershberger and Snoeyink [32], Guibas *et al.* [29], or Hakimi and Schmeichel [30] to drive a binary search among the set of geodesically critical values. Using one of these simple algorithms, all of which are based upon the "greedy method," to test if a given ε is smaller than ε^* takes $O(n)$ time. This implies that we can determine, in $O(n \log n)$ time, the combinatorial structure of $\pi_u(\varepsilon^*)$, and, as mentioned above, a similar procedure gives us the combinatorial structure of $\pi_g(\varepsilon^*)$. Therefore, we can identify all the inflection edges in $H(\varepsilon^*)$, each of which F must contain, by Lemma 2.1.

All that is left, then, is for us to determine all the links of F that do not contain inflection edges.

4. Completing the Construction

Whereas we used geodesic paths to find the zig-zag and zag-zig links, to complete the construction we use a related structure—the visibility graph. In particular, recall that the visibility graph of a set of line segments R has a vertex for each endpoint of a segment in R and an edge for each pair (p, q) such that the line segment $\overline{pq}$ does cross any segment in R, although we allow $\overline{pq}$ to intersect segment endpoints and even contain the segment $\overline{pq}$ if it is in R. It is well known, for example, that geodesic paths always follow visibility graph edges.[3] In our case we are interested in the visibility graph defined on the segments in $U(\varepsilon) \cup G(\varepsilon)$. We say that a line segment s is *U-anchored* (resp. *G-anchored*) if s contains an edge e in the visibility graph of $U(\varepsilon^*) \cup G(\varepsilon^*)$ such that both of e's vertices lie on $U(\varepsilon^*)$ (resp. $G(\varepsilon^*)$). The following lemma establishes an important relationship between a best k-link function approximation to S and these anchored links.

Lemma 4.1. *Any canonical best k-link approximation to S has a U-anchored link containing a vertex of $G(\varepsilon^*)$ or a G-anchored link-containing a vertex of $U(\varepsilon^*)$.*

Proof. The proof follows immediately from the characterization lemma of Hakimi and Schmeichel [30]. □

Our method for enumerating all such edges is based upon several applications of the parametric-searching technique, as optimized by Cole [13], [14]. One such optimization applies to any situation that involves a set $Z = \{z_1, z_2, \ldots, z_m\}$ of m independent binary searches among an ordered set $A = (a_1, a_2, \ldots, a_n)$ of n items, where each comparison $c(a_i, z_j)$ is parametrized by ε. The outcome of $c(a_i, z_j)$ depends upon which of a constant number of intervals, determined by a_i and z_j,

[3] For more information about visibility graphs and their properties see the excellent book by O'Rourke [50].

contain ε^*. If it takes T steps to determine if $\varepsilon^* < \varepsilon$, for a particular ε, then this parametric-searching technique allows us to perform all m binary searches in $O((T + m)(\log n + \log m))$ time (see [13] for details). It also gives us an interval $[\varepsilon_1, \varepsilon_2]$ containing ε^*, which is the intersection of all the intervals determined to contain ε^* during the m searches.

The second optimization applies when one wishes to sort a set $A = \{a_1, a_2, \ldots, a_n\}$, where, as in the previous case, each comparison $c(a_i, a_j)$ is parametrized by ε so that the outcome of $c(a_i, a_j)$ depends upon which of a constant number of intervals, determined by a_i and a_j, contain ε^*. If it takes T steps to determine if $\varepsilon^* < \varepsilon$, for a particular ε, then this parametric-searching technique allows us to sort the elements of A in $O((T + n)\log n)$ time (see [13] and [14] for details). Also, this method gives an interval $[\varepsilon_1, \varepsilon_2]$ containing ε^*, which is the intersection of all the intervals determined to contain ε^* during the sort.

The challenge, of course, in applying these techniques is to design the parametrized sorting and searching procedures so that the results are meaningful. We therefore turn to our application of these techniques.

4.1. *Finding* ε^*

Our method for completing the construction is to perform a parametric search for a U-anchored or G-anchored link satisfying Lemma 4.1. Observe that finding such a link effectively "clamps" $U(\varepsilon)$ and $G(\varepsilon)$ at $\varepsilon = \varepsilon^*$. Also note that we can use the linear-time method of Hershberger and Snoeyink [32], Guibas *et al.* [29], or Hakimi and Schmeichel [30] to resolve comparisons and to give us the final approximation F once we have narrowed the interval of candidate ε values to $[\varepsilon^*, \varepsilon^*]$.

To simplify the discussion we concentrate on the problem of determining a U-anchored link that contains a vertex of $G(\varepsilon^*)$; the method is similar for visibility edges anchored on $G(\varepsilon)$. Our algorithm actually "dovetails" the search for a U-anchored link containing a vertex of $G(\varepsilon^*)$ with the search for a G-anchored link containing a vertex of $U(\varepsilon^*)$.

Call a vertex p of $U(\varepsilon)$ a *left inflection* (resp. *right inflection*) vertex if p is the left (resp. right) endpoint of an inflection edge of $H(\varepsilon^*)$. Following an approach similar to that used by Ghosh [23], consider a portion of $U(\varepsilon)$ between a left inflection vertex p and the leftmost right inflection vertex q to the right of p. Denote this portion of $U(\varepsilon)$ as $U_{[p,q]}(\varepsilon)$. Note that the portion of $\pi_u(\varepsilon^*)$ between p and q is a convex chain of edges such that each consecutive pair forms a "right turn." Denote this portion of $\pi_u(\varepsilon^*)$ as $\pi[p, q]$ (but note that we have yet to determine ε^*—at this point we only know the combinatorial structure of $\pi_u(\varepsilon^*)$). Finally, observe that if $U_{[p,q]}(\varepsilon)$ contains the endpoints v and z of the U-anchored link that we seek, then $\overline{vz}$ must be equal to the common tangent of v and $\pi[p, q]$ as well as z and $\pi[p, q]$. (See Fig. 5.) Moreover, if the link satisfying Lemma 4.1 is a U-anchored link, then it must contain such a tangent edge for some $U_{[p,q]}(\varepsilon)$.

Our first parametric search therefore is to determine for each vertex v on $U_{[p,q]}(\varepsilon)$ its vertex of tangency, $t(v)$, with the convex chain $\pi[p, q]$ at $\varepsilon = \varepsilon^*$ (see Fig. 5). In this case we can use binary-search-based parametric searching [13] applied

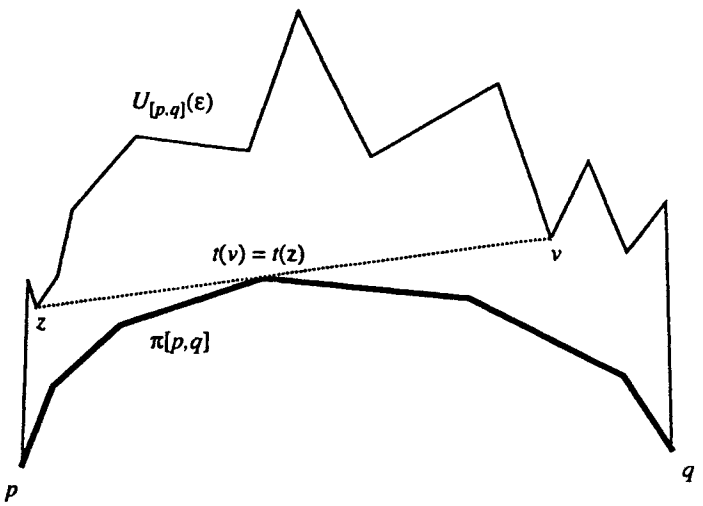

Fig. 5. A U-anchored edge determining ε^*.

to a well-known "binary-search" tangent-finding method (e.g., see [21] and [51]) to find all such tangents in $O(n \log n)$ time. This may still not restrict our interval of ε values to $[\varepsilon^*, \varepsilon^*]$, however, for a vertex v will have the same vertex of tangency, $t(v)$, over a range of ε values.

For each vertex w on $\pi[p, q]$, we collect each vertex v on $U_{[p,q]}(\varepsilon)$ such that $w = t(v)$ into two sets—$l(w)$, containing the vertices to the left of w, and $r(w)$, containing the vertices to the right of w. If w is the vertex of tangency for the U-anchored edge we seek, then that edge is determined by a $v \in l(w)$ and a $z \in r(w)$. In order for these two vertices to be able to "search for each other," however, we must first order the vertices in $l(w)$ and $r(w)$ radially around w. To accomplish this we make a second application of parametric searching, this time using the second version, based upon a parallel-sorting algorithm [13], [14], together with the linear-time method of Hershberger and Snoeyink [32], Guibas *et al.* [29], or Hakimi and Schmeichel [30] for resolving comparisons, to sort all $l(w)$ and $r(w)$ lists in $O(n \log n)$ time. This may still not restrict the range of ε values to $[\varepsilon^*, \varepsilon^*]$, however, for the vertices in an $l(w)$ and $r(w)$ may have the same ordering with respect to w over a range of ε values.

Therefore, to complete the process we must perform one more application of parametric searching where we perform a binary search in $l(w)$ for each $z \in r(w)$ to locate the vertex in $l(w)$ hit by the ray $\overrightarrow{zt(z)}$, if it exists. Given that each $l(w)$ is now sorted, we can perform all of these binary-search parametric searches in $O(n \log n)$ time. Since one of these searches (or a corresponding search for G-anchored edges) must succeed, we finally are guaranteed to have restricted the range of candidate ε values to the interval $[\varepsilon^*, \varepsilon^*]$. Moreover, seeing how the last comparison in this parametric search was resolved using a minimum-link ordered stabber algorithm, this completes the construction, giving us the following theorem.

Theorem 4.2. *Given a set S of n points in the plane, and an integer parameter $1 \leq k \leq n-1$, a best k-link approximation to S can be constructed under the uniform metric in $O(n \log n)$ time.*

5. Simplifying the Implementation

Although our method for finding a best k-link approximation to S is conceptually simple at a high level, implementing it in practice would probably not be so easy. The chief difficulty arises from the (up to) three calls to parametric searching that are made at the end of the algorithm; all the other steps would be relatively straightforward to implement. However, those calls to parametric searching, especially the version based upon a parametric sorting, would be a challenge to implement, and the constant factors they would imply in the running time would not be very small. Fortunately, as we show in this section, we can considerably simplify these calls to parametric searching. In particular, we show how to eliminate the (second) call based upon sorting all together, and we observe how all our calls to parametric searching can be simplified through a simple randomization technique. The resulting algorithm will still be guaranteed to find a best k-link approximation to S, however, for the randomization will only impact the running time of our method, not its correctness. Nevertheless, it will result in a method that runs in $O(n \log n)$ time with very high probability. Moreover, the (expected) constant factors will be reasonably small.

We begin by describing how we can eliminate the call to sorting-based parametric searching, which comprises the second call to parametric searching in our algorithm. Recall that after we have made our first call to (binary-search-based) parametric searching, we have determined for every vertex v on $U_{[p,q]}(\varepsilon)$ its tangent, $t(v)$ on $\pi[p,q]$. Previously, we then collected, for each w on $\pi[p,q]$, all the vertices on $U_{[p,q]}(\varepsilon)$ whose tangent is w to the "left set" $l(w)$ and the "right set" $r(w)$, and we sorted the vertices of $l(w)$ radially around w (using sorting-based parametric searching) so that we can then do a binary search in $l(w)$ for each vertex in $r(w)$ (using binary-search-based parametric searching). Observe, however, that we do not need all the vertices of $l(w)$ to perform this search; we need only those vertices that are visible from w on $U_{[p,q]}(\varepsilon^*)$. Thus, we use $l_{\text{vis}}(w)$ to denote the set of vertices in $l(w)$ that are visible from w on $U_{[p,q]}(\varepsilon^*)$. If we can determine these visible vertices, then, as the following lemma shows, we can sort them without resorting to parametric searching.

Lemma 5.1. *Sorting the vertices of $l_{\text{vis}}(w)$ radially around w as they appear on $U_{[p,q]}(\varepsilon^*)$ gives the same order as a listing of the vertices of $l_{\text{vis}}(w)$ by increasing x-coordinates.*

Proof. Suppose, for the sake of a contradiction, that there are two vertices u and z in $l_{\text{vis}}(w)$ such that u is before z in a radial listing around w as they appear on

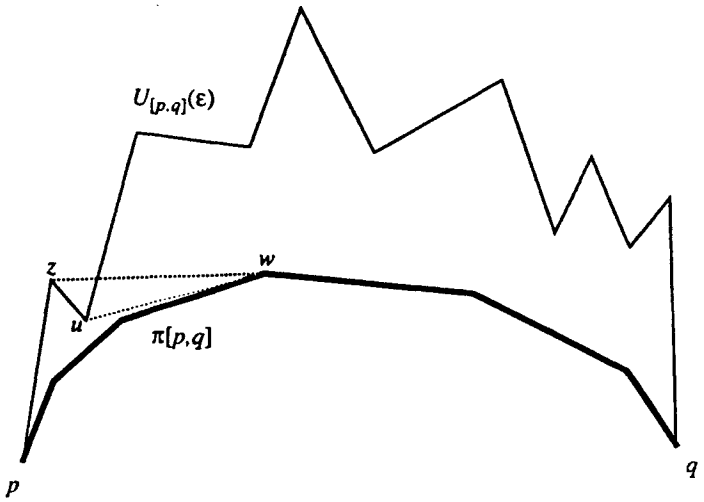

Fig. 6. An illustration of why $l_{\text{vis}}(w)$ is already sorted.

$U_{[p,q]}(\varepsilon^*)$, but $x(z) < x(u)$. (See Fig. 6.) Since w is below the curve $U_{[p,q]}(\varepsilon^*)$, this implies that $U_{[p,q]}(\varepsilon^*)$ intersects the segment $\overline{zw}$. However, this contradicts the fact that z is visible from w; hence, this establishes the lemma. □

Thus, it is sufficient for us to determine the members of $l_{\text{vis}}(w)$, for each w on $\pi[p,q]$, since they are already given by increasing x-coordinates (for that is how the vertices are ordered in S).

So, we turn to the problem of determining the members of $l_{\text{vis}}(w)$ for some given w on $\pi[p,q]$. Let w' denote the vertex on $U_{[p,q]}(\varepsilon)$ directly above w (i.e., if $w = g_i$, then $w' = u_i$). A simple consequence of the proof of Lemma 5.1 is that a vertex v in $l(w)$ is in $l_{\text{vis}}(w)$ if and only if the geodesic path from v to w' is above the line segment $\overline{vw}$. Indeed, it is sufficient that the first edge $e(v)$ in this path be above the segment $\overline{vw}$. Since each such geodesic path remains unchanged for $\varepsilon \in [\varepsilon_1, \varepsilon_2]$ (because it is a sequence of left turns that occur at vertices of $U_{[p,q]}(\varepsilon)$), we can determine $e(v)$ for each such v without knowing the value of ε^*. Specifically, we can use the data structure of Guibas and Hershberger [27], as simplified by Hershberger [31], to determine each $e(v)$ in $O(\log n)$ time. This data structure is relatively simple to construct and query, especially since each $U_{[p,q]}(\varepsilon)$ is a single monotone chain.

Note that each $e(v)$ determines a critical $\hat{\varepsilon}_v$ value such that v is visible if $\varepsilon \geq \hat{\varepsilon}$, and v is not visible otherwise. Thus, once we have determined all the $e(v)$'s, we can then determine which v's are visible from their respective $w = t(v)$ vertices simply by resolving all of the $\hat{e}$'s against ε^* using the method of Hershberger and Snoeyink [32], Guibas *et al.* [29], or Hakimi and Schmeichel [30] to drive a binary search. Since there are $O(n)$ such parametrized comparisons, and each resolution requires $O(n)$ time, the total time needed to resolve all of these comparisons is $O(n \log n)$. This implies that we can substitute the most simple form of parametric searching for the

more-complicated sorting-based method [13], [14]. The resulting algorithm would still run in $O(n \log n)$ time. The only slightly impractical steps in this algorithm are repeated computations of medians (e.g., of the remaining unresolved $\hat{\varepsilon}$'s in each iteration of the above binary search), for which the best deterministic algorithm has a relatively large constant (e.g., see [17]).

However, the computation of a median or weighted median (which is the least-efficient step in the binary-search-based parametric-searching method of Cole [13]) can be significantly improved in practice through the use of randomization (e.g., see [17]). The effect in our case is that the resulting algorithm will still be guaranteed to find a best k-link approximation, but it will not be guaranteed to run in $O(n \log n)$ time. Nevertheless, it will run in $O(n \log n)$ time with very high probability, and the (expected) constants factors in the running time will be very reasonable. Thus, we suggest such a use of randomization in any implementation of our method.

6. Concluson and Directions for Future Work

We have given an $O(n \log n)$-time method for finding a best k-link approximation to a set of n points in the plane under the uniform metric, and we have even given a method for efficiently implementing it in practice. This suggests a number of interesting directions for future work. Some possibilities include examining metrics other than the uniform metric, approximating with higher-degree spline functions, and approximating more general types of geometric objects (such as segments joined in a polygonal path). As with our results, the general goal should be to find a best k-piece approximation, since this is the version of the problem driven by bounds placed upon the computational resources needed to represent the approximation.

Acknowledgments

I would like to thank Jack Snoeyink for some stimulating discussions regarding minimum-link function approximations, for several helpful comments regarding earlier versions of this paper, and, in particular, for pointing out the counterexample illustrated in Figure 2. Finally, I would like to thank Esther Arkin, Mikhail Atallah, Paul Chew, Simon Kasif, S. Rao Kosaraju, Joe Mitchell, and David Mount for several helpful discussions concerning this and related problems.

References

1. P. K. Agarwal, A. Aggarwal, B. Aronov, S. R. Kosaraju, B. Schieber, and S. Suri, Computing external farthest neighbors for a simple polygon, *Discrete Appl. Math.*, **31**(2), 97–111, 1991.
2. P. K. Agarwal, B. Aronov, M. Sharir, and S. Suri, Selecting distances in the plane, *Proc. 6th Ann. ACM Symp. on Computational Geometry*, pp. 321–331, 1990.

3. P. K. Agarwal and J. Matoušek, Ray shooting and parametric search, *Proc. 24th Ann. ACM Symp. on Theory of Computing*, pp. 517–526, 1992.
4. P. K. Agarwal, M. Sharir, and S. Toledo, Applications of parametric searching in geometric optimization, *Proc. 3rd ACM–SIAM Symp. on Discrete Algorithms*, pp. 72–82, 1992.
5. A. Aggarwal, H. Booth, J. O'Rourke, S. Suri, and C. K. Yap, Finding minimal convex nested polygons, *Proc. 1st Ann. ACM Symp. on Computational Geometry*, pp. 296–304, 1985.
6. E. M. Arkin, J. S. B. Mitchell, and S. Suri, Optimal link path queries in a simple polygon, *Proc. 3rd ACM–SIAM Symp. on Discrete Algorithms*, pp. 269–279, 1992.
7. B. Aronov, On the geodesic Voronoi diagram of point sites in a simple polygon, *Algorithmica*, **4**, 109–140, 1989.
8. R. E. Bellman and R. S. Roth, Curve fitting by segmented straight lines, *Amer. Statist. Assoc. J.*, **64**, 1079–1084, 1969.
9. R. E. Bellman and R. S. Roth, *Methods in Approximation: Techniques for Mathematical Modelling*, Reidel, Boston, MA, 1986.
10. B. Chazelle, H. Edelsbrunner, M. Grigni, L. Guibas, J. Hershberger, M. Sharir, and J. Snoeyink, Ray shooting in polygons using geodesic triangulations, *Proc. 18th Internat. Colloq. on Automata Language Programming*. Lecture Notes in Computer Science, vol. 510, Springer-Verlag, Berlin, pp. 661–673, 1991.
11. B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir, Diameter, width, closest line pair, and parametric searching, *Proc. 8th Ann. ACM Symp. on Computational Geometry*, pp. 120–129, 1992.
12. K. L. Clarkson, Linear programming in $O(n3^{d^2})$ time, *Inform. Process. Lett.*, **22**, 21–24, 1986.
13. R. Cole, Slowing down sorting networks to obtain faster sorting algorithms, *J. Assoc. Comput. Mach.*, **34**, 200–208, 1987.
14. R. Cole, Parallel merge sort, *SIAM J. Comput.*, **17**(4), 770–785, 1988.
15. R. Cole, J. Salowe, W. Steiger, and E. Szemerédi, An optimal-time algorithm for slope selection, *SIAM J. Comput.*, **18**, 792–810, 1989.
16. S. D. Conte and C. de Boor, *Elementary Numerical Analysis: An Algorithmic Approach*, 3rd edn., McGraw-Hill, New York, 1980.
17. T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 1990.
18. P. J. Davis, *Interpolation and Approximation*, Blaisdell, New York, 1963.
19. P. Dierckx, *Curve and Surface Fitting with Splines*, Clarendon Press, New York, 1993.
20. M. E. Dyer, Linear time algorithms for two- and three-variable linear programs, *SIAM J. Comput.*, **13**, 31–45, 1984.
21. H. Edelsbrunner, *Agorithms in Combinatorial Geometry*, EATCS Monographs on Theoretical Computer Science, vol. 10, Springer-Verlag, Heidelberg, 1987.
22. H. ElGindy and M. T. Goodrich, Parallel algorithms for shortest path problems in polygons, *Visual Comput.*, **3**, 371–378, 1988.
23. S. Ghosh, Computing visibility polygon from a convex set and related problems, *J. Algorithms*, **12**, 75–95, 1991.
24. S. K. Ghosh and A. Maheshwari, Parallel algorithms for all minimum link paths and link center problems, *Proc. 3rd Scand. Workshop on Algorithm Theory*, Lecture Notes in Computer Science, vol. 621, Springer-Verlag, Berlin, pp. 106–117, 1992.
25. M. T. Goodrich and R. Tamassia, Dynamic ray shooting and shortest paths via balanced geodesic triangulations, *Proc. 9th Ann. ACM Symp. on Computational Geometry*, pp. 318–327, 1993.
26. L. J. Guibas and J. Hershberger, Optimal shortest path queries in a simple polygon, *Proc. 3rd Ann. ACM Symp. on Computational Geometry*, pp. 50–63, 1987.
27. L. J. Guibas and J. Hershberger, Optimal shortest path queries in a simple polygon, *J. Comput. System Sci.*, **39**, 126–152, 1989.
28. L. J. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan, Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons, *Algorithmica*, **2**, 209–233, 1987.

29. L. J. Guibas, J. E. Hershberger, J. S. B. Mitchell, and J. S. Snoeyink, Approximating polygons and subdivisions with minimum link paths, *Proc. 2nd Ann. SIGAL Internat. Symp. on Algorithms*, Lecture Notes in Computer Science, vol. 557, Springer-Verlag, Berlin, pp. 151–162, 1991.
30. S. L. Hakimi and E. F. Schmeichel, Fitting polygonal functions to a set of points in the plane, *CVGIP: Graph. Mod. Image Proc.*, **53**(2), 132–136, 1991.
31. J. Hershberger, A new data structure for shortest path queries in a simple polygon, *Inform. Process. Lett.*, **38**, 231–235, 1991.
32. J. Hershberger and J. Snoeyink, Computing minimum length paths of a given homotopy class, *Proc. 2nd Workshop on Algorithms and Data Structures*, Lecture Notes in Computer Science, vol. 519, Springer-Verlag, Berlin, pp. 331–342, 1991.
33. H. Imai and M. Iri, Computational-geometric methods for polygonal approximations of a curve, *Comput. Vision Graph. Image Process.*, **36**, 31–41, 1986.
34. H. Imai and M. Iri, An optimal algorithm for approximating a piecewise linear function, *J. Inform. Process.*, **9**(3), 159–162, 1986.
35. H. Imai and M. Iri, Polygonal approximations of a curve-formulations and algorithms, in *Computational Morphology*, G. T. Toussaint, ed., North-Holland, Amsterdam, pp. 71–86, 1988.
36. D. L. B. Jupp, Approximation to data by splines with free knots, *SIAM J. Numer. Anal.*, **15**(2), 328–343, 1978.
37. Y. Ke, An efficient algorithm for link-distance problems, *Proc. 5th Ann. ACM Symp. on Computational Geometry*, pp. 69–78, 1989.
38. D. T. Lee and F. P. Preparata, Euclidean shortest paths in the presence of rectilinear barriers, *Networks*, **14**, 393–410, 1984.
39. W. Lenhart, R. Pollack, J.-R. Sack, R. Seidel, M. Sharir, S. Suri, G. T. Toussaint, S. Whitesides, and C. K. Yap, Computing the link center of a simple polygon, *Discrete Comput. Geom.*, **3**, 281–293, 1988.
40. J. Matoušek, M. Sharir, and E. Welzl, A subexponential bound for linear programming, *Proc. 8th Ann. ACM Symp. on Computational Geometry*, pp. 1–8, 1992.
41. N. Megiddo, Applying parallel computation algorithms in the design of serial algorithms, *J. Assoc. Comput. Mach.*, **30**, 852–865, 1983.
42. N. Megiddo, Linear-time algorithms for linear programming in R^3 and related problems, *SIAM J. Comput.*, **12**, 759–776, 1983.
43. N. Megiddo, Linear programming in linear time when the dimension is fixed, *J. Assoc. Comput. Mach.*, **31**, 114–127, 1984.
44. A. Melkman and J. O'Rourke, On polygonal chain approximation, in *Computational Morphology*, G. T. Toussaint, ed., North-Holland, Amsterdam, pp. 87–95, 1988.
45. J. S. B. Mitchell, An algorithmic approach to some problems in terrain navigation, *Artificial Intelligence*, **37**, 171–201, 1988.
46. J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou, The discrete geodesic problem, *SIAM J. Comput.*, **16**, 647–668, 1987.
47. J. S. B. Mitchell, C. Piatko, and E. M. Arkin, Computing a shortest k-link path in a polygon, *Proc. 33rd Ann. IEEE Symp. on Foundations of Computer Science*, pp. 573–582, 1992.
48. J. S. B. Mitchell, G. Rote, and G. Woeginger, Minimum-link paths among obstacles in the plane, *Proc. 6th Ann. ACM Symp. on Computational Geometry*, pp. 63–72, 1990.
49. J. S. B. Mitchell and S. Suri, Separation and approximation of polyhedral surfaces, *Proc. 3rd ACM–SIAM Symp. on Discrete Algorithms*, pp. 296–306, 1992.
50. J. O'Rourke, *Art Gallery Theorems and Algorithms*, Oxford University Press, New York, 1987.
51. F. P. Preparata and M. I. Shamos, *Computational Geometry: an Introduction*, Springer-Verlag, New York, 1985.
52. J.-M. Robert and G. Toussaint, Linear approximation of simple objects, *Proc. 9th Symp. on Theoretical Aspects of Computing Science*, Lecture Notes in Computer Science, vol. 577, Springer-Verlag, Berlin, pp. 233–244, 1992.
53. R. Seidel, Small-dimensional linear programming and convex hulls made easy, *Discrete Comput. Geom.*, **6**, 423–434, 1991.

54. S. Suri, A linear time algorithm for minimum link paths inside a simple polygon, *Comput. Vision Graph. Image Process.*, **35**, 99–110, 1986.
55. S. Suri, Minimum link paths in polygons and related problems, Ph.D. thesis, Department of Computer Science, Johns Hopkins University, Baltimore, MD, 1987.
56. S. Suri, Computing geodesic furthest neighbors in simple polygons, *J. Comput. System Sci.*, **39**, 220–235, 1989.
57. G. T. Toussaint, On the complexity of approximating polygonal curves in the plane, *Proc. IASTED, Internat. Symp. on Robotics and Automation*, Lugano, 1985.
58. D. P. Wang, N. F. Huang, H. S. Chao, and R. C. T. Lee, Plane sweep algorithms for polygonal approximation problems with applications, *Proc. 4th Ann. Internat. Symp. on Algorithms and Computing* (*ISAAC 93*), Lecture Notes in Computer Science, vol. 762, Springer-Verlag, Berlin, pp. 515–522, 1993.

Received March 1994, and in revised form January 1995.