

# $k$ -Link Shortest Paths in Weighted Subdivisions

Ovidiu Daescu<sup>1\*</sup>, Joseph S.B. Mitchell<sup>2\*\*</sup>, Simeon Ntafos<sup>1</sup>,  
James D. Palmer<sup>1</sup>, and Chee K. Yap<sup>3\*\*\*</sup>

<sup>1</sup> Department of Computer Science, University of Texas at Dallas,  
Richardson, TX 75080, USA. {daescu,ntafos,jdp011100}@utdallas.edu

<sup>2</sup> Department of Applied Mathematics and Statistics,  
Stony Brook University, Stony Brook, NY 11794, USA. jsbm@ams.sunysb.edu

<sup>3</sup> Department of Computer Science,  
New York University, New York, NY 10012, USA. yap@cs.nyu.edu

**Abstract.** We study the shortest path problem in weighted polygonal subdivisions of the plane, with the additional constraint of an upper bound,  $k$ , on the number of links (segments) in the path. We prove structural properties of optimal paths and utilize these results to obtain approximation algorithms that yield a path having  $O(k)$  links and weighted length at most  $(1 + \epsilon)$  times the weighted length of an optimal  $k$ -link path, for any fixed  $\epsilon > 0$ . Some of our results make use of a new solution for the 1-link case, based on computing optimal solutions for a special sum-of-fractionals (SOF) problem. We have implemented a system, based on the CORE library, for computing optimal 1-link paths; we experimentally compare our new solution with a previous method for 1-link optimal paths based on a prune-and-search scheme.

## 1 Introduction

A weighted subdivision  $R$  in the plane is a decomposition of the plane into polygonal regions, each with an associated nonnegative weight. The weighted length of a line segment,  $\overline{ab}$ , joining two points  $a$  and  $b$  within the same region  $R_i \in R$  is defined as the product of the weight  $w_i$  of region  $R_i$  and the Euclidean length  $|\overline{ab}|$  of the line segment  $\overline{ab}$ . For a polygonal path  $p$ , the weighted length is given by a finite sum of subsegment (Euclidean) lengths, each multiplied by the weight of the region containing the subsegment.

We are motivated by applications that require paths that are optimal with respect to more than one criterion. For example, in emergency and medical interventions, in military route planning, and in air traffic applications, one may desire polygonal paths having only a few links (turns), while also having a small (weighted) length. A minimum-weight path may have unacceptably many turns.

---

\* Corresponding author. Daescu's work is supported by NSF grant CCF-0430366.

\*\* J. Mitchell is partially supported by the U.S.-Israel Binational Science Foundation (2000160), NASA (NAG2-1620), NSF (CCR-0098172, ACI-0328930, CCF-0431030), and Metron Aviation.

\*\*\* Yap's work is supported by NSF Grant CCF-0430836.

In this paper we study the *k-link shortest path problem* in weighted regions, in which we place an upper bound,  $k$ , on the number of links (edges) in the polygonal path, while minimizing the weighted length of the path. We compute paths from a given source region  $R_s$  to a target region  $R_t$  in a weighted subdivision  $R$ . An important special case, which arises as a subproblem in our approach, is that of computing a *1-link shortest path* from a source to a target.

**Related Work.** In the unweighted setting, approximation algorithms are known for  $k$ -link shortest paths inside simple polygons and polygons with holes [16]. In weighted subdivisions, turn-constrained optimal paths have been studied in the context of air traffic routing (using a grid-based dynamic programming algorithm) [10] and, very recently, in the context of mine avoidance routing (using a genetic algorithm) [13]; neither of these approaches gives approximation algorithm guarantees. The 1-link shortest path problem to compute an optimal “link” (or “penetration”) in weighted subdivisions has been studied in [2, 3, 6], where it is shown that the special structure of the optimal solution allows for efficient search for solutions.

Without a bound on the number of links, several results are known for computing shortest paths in weighted regions [1, 9, 11, 12, 14, 15, 19], beginning with the first polynomial-time results of Mitchell and Papadimitriou [15], who compute  $(1 + \epsilon)$ -approximate geodesic shortest paths on weighted terrains.

**Our Results.** We present the following results. (1) We prove there exists a  $(2k - 1)$ -link path  $p$  from source  $R_s$  to target  $R_t$  that turns only on the edges of  $R$ , such that the weighted length of  $p$  is at most that of an optimal  $k$ -link path  $p^*$  from  $R_s$  to  $R_t$ . (2) We give two approximation algorithms for computing  $k$ -link shortest paths in weighted regions. The first one requires the computation of 1-link shortest paths and produces a  $(5k - 2)$ -link path whose weight is within factor  $(1 + \epsilon)$  of optimal. The second algorithm relies only on computation of  $(1 + \epsilon/6)$ -approximate 1-link shortest paths and produces a  $14k$ -link path whose weight is within factor  $(1 + \epsilon)$  of optimal. (3) We give a new (in this context) algorithm for computing 1-link shortest paths, based on solving a variant of the sum-of-linear-fractionals (SOLF) problem [8]. (4) We have implemented a system, based on the CORE library [5], for computing 1-link shortest paths. We compare experimentally two algorithms for 1-link shortest paths: one based on our variant of the SOLF problem, and one based on a prune-and-search scheme [6].

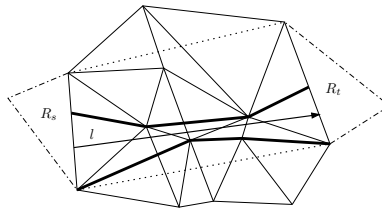
**Preliminaries.** Let  $R$  be a planar weighted subdivision with a total of  $n$  vertices and a set  $\mathcal{E}$  of  $O(n)$  edges. Without loss of generality, we assume  $R$  is triangulated, the source and target regions can be separated by a vertical line, and the vertices of  $R$  are in general position (no three collinear). For a path  $p$ , we let  $|p|$  denote the Euclidean length of  $p$  and  $\|p\|$  the weighted length of  $p$ . A polygonal path  $p$  whose turn points all lie on the edge set  $\mathcal{E}$  is said to be *edge-restricted*.

Consider a link (line segment)  $l$  between two edges  $e_s$  and  $e_t$  of  $R$ , with  $e_s$  and  $e_t$  not bounding the same (triangular) face (otherwise, the problem is trivial). The weighted length of the link  $l$  is  $d(l) = \sum_{i:l \cap R_i \neq \emptyset} w_i d_i(l)$ , where  $w_i$  is the weight of  $R_i$  and  $d_i(l)$  is the Euclidean length of  $l$  within the region  $R_i \in R$ .

Let the equation of the line supporting  $l$  be  $y = mx + b$ . Let  $R_i$  be a region intersected by  $l$ , with  $s_i^1$  and  $s_i^2$  the two sides of  $R_i$  that intersect  $l$ , at points  $v_i^1(x_i^1, y_i^1)$  and  $v_i^2(x_i^2, y_i^2)$ , respectively. From  $d_i(l) = \sqrt{1 + m^2}|x_i^2 - x_i^1|$ , we have that

$$d(l) = \sqrt{1 + m^2} \sum_{i: l \cap R_i \neq \emptyset} w_i |x_i^2 - x_i^1| = \sqrt{1 + m^2} \sum_i \sigma_i w_i \frac{b_i - b}{m - m_i} \quad (1)$$

where  $\sigma_i$  is +1 or -1,  $m_i$  and  $b_i$  are constants, and the number of terms in the summation is  $O(n)$ . If  $l$  is rotated and translated, while keeping its endpoints on  $e_s$  and  $e_t$ , the expression for  $d(l)$  does not change as long as no vertex of  $R$  is crossed by  $l$ . The corresponding set of pairs  $(m, b)$  defines a two-dimensional convex domain  $D$  whose edges correspond to  $l$  being tangent to a vertex of  $R$  and whose vertices correspond to  $l$  passing through two vertices of  $R$  [2]. The region swept by  $l$ , while maintaining its combinatorial type, is called an *hourglass* (Fig. 1). For a fixed slope  $m$ , we see from (1) that  $d(l)$  is linear in  $b$  as  $l$  varies within the hourglass; thus, there exists a segment  $l$  minimizing  $d(l)$ , over the hourglass, passing through a vertex  $v$  of  $R$  [6]. For a fixed choice of the vertex  $v$  of an hourglass, the expression for  $d(l)$  depends only on the slope  $m$  of the line through  $l$ :



**Fig. 1.** An hourglass for which the formula for  $d(l)$  does not change.

$$d(l) = \sqrt{1 + m^2} (d_0 + \sum_i \frac{a_i}{m + b_i}), \quad (2)$$

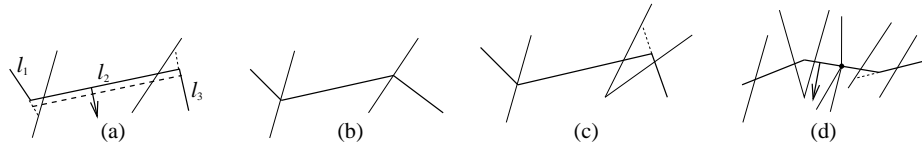
where  $d_0, a_i$  and  $b_i$  are constants. Note that  $d(l)$  is bounded and positive.

## 2 Approximating $k$ -Link Shortest Paths

**Lemma 1.** *Let  $p$  be a shortest  $k$ -link path between edges  $e_s$  and  $e_t$  of  $R$ . Then, each link of  $p$  has an endpoint on an edge of  $R$  or goes through a vertex of  $R$ .*

*Proof.* The proof is by contradiction. Let  $p$  be a  $k$ -link path between  $e_s$  and  $e_t$ , let  $l_1, l_2$  and  $l_3$  be three consecutive links of  $p$  and refer to Figure 2. Assume that  $p$  makes two consecutive region interior turns, one at the common endpoint of the links  $l_1$  and  $l_2$  and the other at the common endpoint of the links  $l_2$  and  $l_3$  (Figure 2 (a)). Assume also that the turn for  $l_1$  and  $l_2$  is inside the region  $R_1$ , the turn for  $l_2$  and  $l_3$  is inside the region  $R_2$ , and extend  $l_1$  and  $l_3$  to intersect the boundaries of  $R_1$  and  $R_2$ , respectively. If we slide  $l_2$  parallel to itself (i.e., the slope of  $l_2$  is fixed) then the length of  $p$  changes only locally, corresponding to the changes in length for  $l_1, l_2$ , and  $l_3$ . The change in length for  $l_2$  is a linear function of the intercept of the line supporting  $l_2$ . It can also be shown that the changes for  $l_1$  and  $l_3$  are linear functions of the same variable. Thus,  $p$  can be

improved locally by sliding  $l_2$  until either it hits a vertex of  $R$  or an intersection point of  $l_1 \cap R_1$  or  $l_3 \cap R_2$ . Figure 2 (b), (c) and (d) shows three possible cases for consecutive links on an optimal path  $p$ . The other cases can be obtained by symmetry. The rest follows by induction on the number of links on the path.  $\square$



**Fig. 2.** (a) Three consecutive links  $l_1, l_2$ , and  $l_3$ ; (b) the middle link ends on edges; (c) an inside region turn with a link ending on an edge and (d) an inside turn with a link stopped at a vertex of the subdivision.

**Theorem 1.** *There exists a path  $p$  between  $e_s$  and  $e_t$  with at most  $(2k-1)$ -links that turns only on the edges of  $R$  and such that the weighted length of  $p$  is at most that of a  $k$ -link shortest path  $p^*$  from  $e_s$  to  $e_t$ .*

We now show how to modify previous discretization schemes (e.g., [1, 19]) in order to approximate edge-restricted  $k$ -link shortest paths. We say a path  $p$  from source point  $s$  to destination point  $t$  is an  $\epsilon$ -good approximate shortest path if  $\|p\| \leq (1 + \epsilon)\|p_k(s, t)\|$ , where  $p_k(s, t)$  is an edge-restricted  $k$ -link shortest path from  $s$  to  $t$ .

Let  $\mathcal{E}$  be the set of boundary edges of  $R$ . Let  $\mathcal{E}(v)$  be the set of subdivision edges having endpoint  $v$ , and let  $d(v)$  be the minimum distance between  $v$  and the edges in  $\mathcal{E} \setminus \mathcal{E}(v)$ . (Note that if  $v$  is not a vertex of  $R$ , then  $\mathcal{E}(v) = \emptyset$ .) For each edge  $e \in \mathcal{E}$ , let  $d(e) = \sup_{x \in e} d(x)$ . Let  $v(e)$  be a point on  $e$  such that  $d(v(e)) = d(e)$ . For each vertex  $v \in R$ , let  $r(v) = \epsilon \frac{d(v)}{5}$ . We refer to  $r(v)$  as the vertex radius for  $v$ . The disk of radius  $r(v)$  centered at  $v$  defines the *vertex-neighborhood*  $S(v)$  of the vertex  $v$ .

We now describe how the Steiner points on an edge  $e = \overline{v_1 v_2}$  are chosen. Each vertex  $v_i$ , where  $i = 1, 2$ , has a vertex-neighborhood  $S(v_i)$  of radius  $r(v_i)$  and the Steiner points  $v_{i,1}, \dots, v_{i,j_i}$  are placed on  $e$  such that  $|v_i v_{i,1}| = r(v_i)$  and  $|v_{i,m} v_{i,m+1}| = \epsilon d(v_{i,m})$ ,  $m = 1, 2, \dots, j_i - 1$ . The value of  $j_i$  is such that  $v_{i,j_i} = v_i(e)$ . We call the line segment formed by two adjacent Steiner points  $v_{i,m}$  and  $v_{i,m+1}$  a *Steiner edge*. The pairing of any two Steiner edges forms a quadrilateral shape called a *Steiner strip*. The shape could be degenerate if the Steiner edges are on the same boundary edge. In [19], it has been shown such a discretization scheme can be used to guarantee a  $3\epsilon$ -good approximate shortest path. With  $\delta$  the maximum number of Steiner points placed on an edge, this discretization scheme gives  $\delta = O(\frac{1}{\epsilon} \log \frac{1}{\epsilon})$ . (It is important to note that  $\delta$  also depends on some geometric parameters of  $R$ , such as the longest edge; see [1].)

Let  $(e_i, e_j)$  be a pair of edges of the subdivision and assume a  $k$ -link edge-restricted shortest path has a turn on  $e_i$  and the next turn on  $e_j$ . Then, the shortest path link  $l^*$  between  $e_i$  and  $e_j$  is contained in an hourglass corresponding to one of the  $O(n^2)$  1-link shortest path subproblems defined by the pair  $(e_i, e_j)$  (see

Fig. 1). Each of the two endpoints of the shortest path link  $l^*$  lies between either two Steiner points or a vertex and a Steiner point. Assume each endpoint is between two Steiner points. Let  $l$  be one of the four line segments between  $e_i$  and  $e_j$  that are defined by the four Steiner points and further assume that  $l$  is fully contained in the same hourglass as  $l^*$ . Then,  $d(l)$  and  $d(l^*)$  have similar description and  $d(l) - d(l^*) = \sum_{i=1}^m w_i d_i(l) - \sum_{i=1}^m w_i d_i(l^*) = \sum_{i=1}^m w_i (d_i(l) - d_i(l^*))$  where  $m = O(n)$  and, without loss of generality, we assume that  $l$  and  $l^*$  intersect the regions  $R_1, R_2, \dots, R_m$  of subdivision  $R$ . Asking that  $\sum_{i=1}^m w_i (d_i(l) - d_i(l^*)) \leq \epsilon d(l^*)$  implies  $\sum_{i=1}^m w_i (d_i(l) - d_i(l^*)) \leq \epsilon \sum_{i=1}^m w_i d_i(l^*)$  and thus  $\sum_{i=1}^m w_i d_i(l) \leq \sum_{i=1}^m w_i ((1 + \epsilon) d_i(l^*))$ . Thus, the Steiner points on the edges  $e_i$  and  $e_j$  should be such that  $d_i(l) \leq (1 + \epsilon) d_i(l^*)$ ,  $i = 1, 2, \dots, m$ .

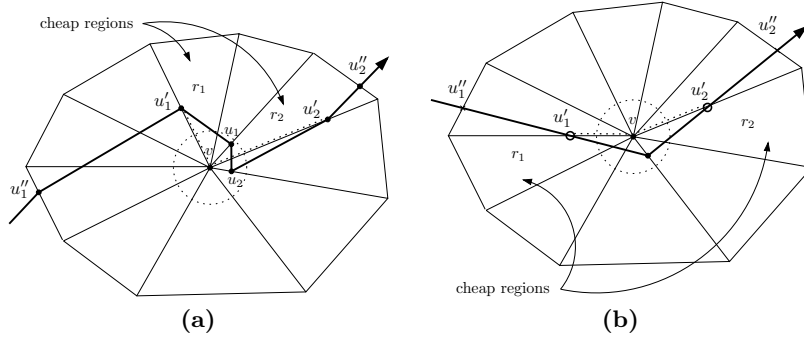
Clearly, if  $l^*$  passes very close to a vertex  $v$  of  $R$  and intersects two or more edges incident to  $v$ , a discretization scheme cannot guarantee that  $d_i(l) \leq (1 + \epsilon) d_i(l^*)$ , for the corresponding distances. More generally, a similar situation appears when the optimal path has multiple turns very close to  $v$ , e.g., with link endpoints between a vertex and a Steiner point (Fig. 3 (a)). Another potential problem is that it is possible that none of the four line segments joining the points that define the Steiner strip between  $e_i$  and  $e_j$  is fully contained in the same hourglass as  $l^*$ , implying that  $l$  and  $l^*$  intersect different subsets of regions of  $R$ . The challenge, then, is to formulate approximation schemes that either avoid or address these problems.

We address the first problem using normalization. A path  $p$  is said to be *normalized* if it does not turn on edges within a vertex-*vicinity*. In Lemma 1 of [19], Sun and Reif show that for any path  $p$  from  $s$  to  $t$ , there is a normalized path  $\hat{p}$  from  $s$  to  $t$  such that  $\|\hat{p}\| = (1 + \frac{\epsilon}{2})\|p\|$ . For  $k$ -link edge-restricted paths we have the following related lemma.

**Lemma 2.** *For any  $k$ -link edge-restricted path  $p_k$  from  $s$  to  $t$ , there is a normalized path  $\hat{p}$  from  $s$  to  $t$  such that  $\|\hat{p}\| = (1 + \epsilon/2)\|p_k\|$ .*

*Proof.* We observe that  $p$  need not be an optimal path for Sun and Reif's Lemma 1 to hold. Thus, the same proof holds for a  $k$ -link path,  $p_k$ . However, there is no guarantee that  $\hat{p}$  has only  $k$  links.  $\square$

We would like to bound the number of links that may be “added” to  $\hat{p}$  relative to  $p_k$ . Let  $u_1''$  be the boundary point where  $p_k$  first enters a region adjacent to  $v$  and let  $u_2''$  be the boundary point where  $p_k$  leaves a region adjacent to  $v$ . Let  $u_1' \in p_k$  be the boundary point on the cheapest region intersected by  $p_k$  before entering the vicinity of  $v$  and let  $u_2' \in p_k$  be the boundary point on the cheapest region intersected by the last link of  $p_k$  that has nonempty intersection with the vicinity of  $v$  (see Fig. 3 (a)). In constructing  $\hat{p}$ , we may remove zero or more links in  $p_k$  completely contained in the vertex-*vicinity* (link  $\overline{u_1 u_2}$  in Fig. 3 (a)) and then add up to two additional links that begin outside the vertex-*vicinity* and pass through  $v$  (links  $\overline{u_1' v}$  and  $\overline{v u_2'}$  in Fig. 3 (a)). Fig. 3 (a) represents a situation in which the number of links in the subpath  $p[u_1'', u_2'']$  in the original path  $p_k$  is one more than in the normalized path,  $\hat{p}$ . Fig. 3 (b) is representative of the worst case, where two links are added and none are removed.



**Fig. 3.** (a) The solid line path is part of an optimal  $k$ -link path. The dotted path represents a normalized path. (b) The solid line path represents a single turn in an optimal  $k$ -link path. The dotted path represents a normalized path.

**Lemma 3.** For any  $k$ -link path  $p_k$  from  $s$  to  $t$ , there is a normalized edge-restricted path  $\hat{p}$  such that  $\|\hat{p}\| = (1 + \epsilon/2)\|p_k\|$  and  $\hat{p}$  has at most  $3k - 2$  links.

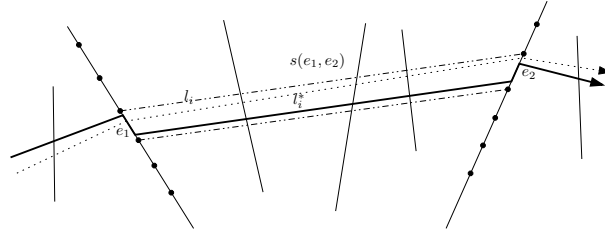
*Proof.* We observe that at most two links need to be added for each link of  $p_k$  when constructing a normalized path  $\hat{p}$  from a path  $p_k$ . Let  $k_1$  be the number of links that need normalization. Then, they are replaced by  $3k_1 - 2$  links. Let  $k_2$  be the remaining links (i.e.,  $k = k_1 + k_2$ ). These links may have an endpoint that is not on a vertex or edge of  $R$ . From Theorem 1, at most  $2k_2 - 1$  links are required to create an approximating path restricted to edges. Note that each of the at most  $k_2 - 1$  links that are within a triangle of  $R$  may also require normalization, which would add  $k_2 - 1$  extra links. Thus, in a normalized edge restricted path the  $k_2$  links are replaced by at most  $3k_2 - 2$  links.  $\square$

**An approximation using exact optimal links.** Recall that we refer to the line segment formed by two adjacent Steiner points  $v_{i,j}$  and  $v_{i,j+1}$  on an edge incident to vertex  $v_i \in R$  as a *Steiner edge*. The pairing of any two Steiner edges forms a *Steiner strip*. The pairing of a Steiner edge with a vertex of  $R$  forms a *Steiner-vertex strip*. The pairing of any two vertices of  $R$  forms a *vertex-pair*.

Consider a shortest normalized  $k$ -link path,  $p_k$ , that only turns on edges. Each link  $l_i$  in  $p_k$  is “captured” either by a Steiner strip, a Steiner-vertex strip, or a vertex-pair.

If  $l_i$  is captured by a vertex-pair  $(u, v)$  the weighted length of  $l_i$  can be easily computed as  $\|\overline{uv}\|$ . Then, the difficulty in approximating the weighted length of  $l_i$  is when  $l_i$  is captured by either a Steiner strip or a Steiner-vertex strip, where the Steiner strip is the more general case.

Let  $s(e_1, e_2)$  be a Steiner strip that captures  $l_i$ , where  $e_1$  and  $e_2$  are the Steiner edges at which  $l_i$  originates and terminates, respectively. Although  $l_i$  forms part of an optimal  $k$ -link path,  $l_i$  is not necessarily an optimal link between  $e_1$  and  $e_2$ . This suggests one could try to replace  $l_i$  with the optimal link  $l_i^*$  between  $e_1$  and  $e_2$ . We show in the next lemma (proof omitted here) that using  $k$  optimal links and  $k$  “small” connecting links we can construct an approximating path with  $2k$  links that provides an  $\epsilon$ -good approximation of  $p_k$ . We define an edge-crawling link as a link along an edge of  $R$  and contained within a Steiner edge (see Fig. 4).



**Fig. 4.** The dotted line path represents an optimal  $k$ -link path. The solid line path represents a  $2k$  approximation made up of optimal links and “small” connecting, edge-crawling links.

**Lemma 4.** *A normalized  $k$ -link path,  $p_k$ , that turns on edges can be approximated by an  $\epsilon$ -good  $2k$ -link path made up of  $k$  optimal links connected by  $k$  edge-crawling links.*

**Theorem 2.** *For sufficiently small positive  $\epsilon$ , a  $k$ -link shortest path can be approximated with a normalized  $\epsilon$ -good  $(5k - 2)$ -link edge restricted path.*

*Proof.* It follows from Lemma 3 that there is a normalized edge-restricted path with  $3k - 2$  links. When applying Lemma 4, only  $2k$  optimal links on this path need small edge crawling links. The approximation factor is  $(1 + \epsilon/2)(1 + \epsilon) = (1 + 3\epsilon/2 + \epsilon^2/2) \leq (1 + 2\epsilon)$ , assuming  $\epsilon \leq 1/2$ . Then, we can use  $\epsilon = \epsilon/2$  when generating the Steiner points to get the claimed result (we will no longer mention this in the remaining proofs).  $\square$

We now show how to use this discretization scheme to construct a weighted graph  $G_\epsilon(V, E)$  that captures approximate paths. Each node  $v \in V$  corresponds to a vertex of  $R$  or a Steiner edge in our discretization scheme. Each edge  $e \in E$  corresponds to either a Steiner strip, a Steiner-vertex strip or a vertex-pair. The weight of  $e$  is the weighted length of a shortest 1-link path through the respective Steiner strip, Steiner-vertex strip or vertex-pair. (Some other geometric informations are associated with  $G_\epsilon$ ; we defer this to the full paper.) This differs from the discretization graph in [1, 19], where  $V$  is formed of Steiner points and  $E$  is made up of links between Steiner points.

The number of vertices in  $V$  is  $O(\delta n)$  and the number of edges in  $E$  is  $O((\delta n)^2)$ . Computing a single edge in  $G_\epsilon$  corresponds to solving a 1-link shortest path problem for a specific hourglass. Let this time be  $T_h(n)$ . Thus, the time to compute  $G_\epsilon$  is  $O((\delta n)^2 T_h(n))$ . Once  $G_\epsilon$  is constructed, we can use dynamic programming to find a  $k$ -link shortest path in  $G_\epsilon$  in  $O(k(\delta n)^4)$  time.

**Theorem 3.** *There exists a path approximation graph  $G_\epsilon$  of size  $O((\delta n)^2)$ , that can be constructed in  $O((\delta n)^2 T_h(n))$  time, and can be used to report in  $O(k(\delta n)^4)$  time a  $(5k - 2)$ -link path that  $(1 + \epsilon)$ -approximates a  $k$ -link shortest path.*

**An approximation using approximate optimal links.** Finding optimal 1-link paths can be computationally expensive (e.g., see Section 3). We now describe a technique for computing approximate optimal 1-link paths for the subproblems that arise in building the path approximation graph.

Observe that a link  $l$  between two Steiner edges  $e_1$  and  $e_2$  may intersect several Steiner edges placed on the edges of each region  $l$  crosses. Each region that  $l$  intersects captures part of  $l$  in a Steiner strip with one exception. If  $l$  passes within the vertex-vicinity of a vertex in  $R$  then part of  $l$  is not captured. Furthermore,  $l$  could intersect  $O(n)$  vertex-neighborhoods.

We would like to find an approximating path that has no vertex-neighborhood intersections. However, as we have seen earlier, avoiding a vertex neighborhood could add two extra links on the approximating path. To reduce the increase in the number of links on the approximating path we then need to reduce the number of vertex-neighborhoods that can be intersected by a single link.

We accomplish this by changing the discretization scheme slightly. Let  $\mathcal{T}$  be the set of all possible vertex triplets formed by the vertices in  $R$ .  $O(n^3)$  such triplets exist that correspond to  $O(n^3)$  triangles. For each vertex in each triangle we can compute the minimum distance to the opposite edge. Let  $\gamma_i$  be the minimum such distance obtained from a triplet containing the vertex  $v_i$ , and let  $\gamma = \min\{\gamma_i/2 \mid i = 1, 2, \dots, n\}$ . By a recent result in [7],  $\gamma_i$  can be found in  $O(n \log n)$  time and thus  $\gamma$  can be computed in  $O(n^2 \log n)$  time.

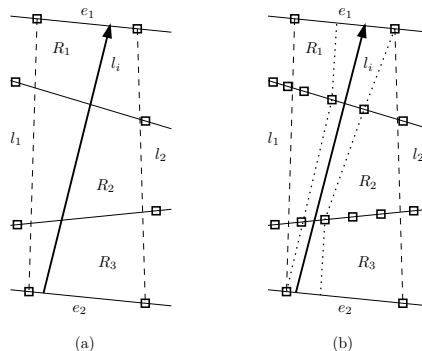
Let the new vertex-neighborhood radius,  $r'(v_i)$ , be  $\min(r(v_i), \gamma)$ . The first Steiner point after a vertex  $v_i$  is placed such that  $|v_i v_{i,1}| = r'(v_i)$ . A path  $p$  is said to be  $\gamma$ -normalized if each link in  $p$  does not turn within a vertex-neighborhood or pass through a vertex-neighborhood without also passing through the vertex.

The proofs of Lemmas 5-8 are deferred to the full paper.

**Lemma 5.** *For any  $k$ -link path  $p_k$  from  $s$  to  $t$ , there is a  $\gamma$ -normalized edge-restricted path  $\hat{p}$  from  $s$  to  $t$  such that (1)  $\|\hat{p}\| = (1 + \frac{\epsilon}{2})\|p_k\|$  and (2)  $\hat{p}$  has no more than eight times as many links as  $p_k$ .*

Next, we consider the computation of a single link. Fig. 5 (a) illustrates a situation in which a Steiner strip intersects one or more edges with sparsely placed Steiner points. Fig. 5 (b) illustrates a situation where a Steiner strip intersects one or more edges with densely placed Steiner points. In Fig. 5 (a), the line segments  $l_1$  and  $l_2$  provide an  $\epsilon$ -good approximation for all line segments captured between  $l_1$  and  $l_2$ . In Fig. 5 (b),  $l_1$  and  $l_2$  only provide  $\epsilon$ -good approximations for the lines that pass between the same Steiner points as  $l_1$  and  $l_2$ .

The number of possible approximating 1-link paths for a pair of edges is  $O((\delta n)^2)$ . The complexity to store both every possible approximation and the ranges over which those approximations are valid for the entire subdivision is then  $O((\delta n)^4)$ .



**Fig. 5.** A Steiner strip formed by lines  $l_1$  and  $l_2$  may intersect edges that are more coarsely (a) or more finely (b) sampled.

**Lemma 6.** *Let  $\hat{l}$  be an optimal  $\gamma$ -normalized link between two edges  $e_1$  and  $e_2$ . A single  $(1 + \epsilon)$ -factor approximating link can be computed in  $O(n(\delta n)^2)$  time.*

**Lemma 7.** *For sufficiently small positive  $\epsilon$ , a  $k$ -link shortest path can be approximated with a  $\gamma$ -normalized  $\epsilon$ -good  $(14k)$ -link edge restricted path.*

**Lemma 8.** *The path approximation graph  $G_\epsilon$  has size  $O((\delta n)^2)$  and can be constructed in  $O(n(\delta n)^4)$  time.*

Using a dynamic programming algorithm for computing  $k$ -link shortest paths in weighted graphs, one can find an approximate solution for the  $k$ -link shortest path using  $G_\epsilon$  in  $O(k(\delta n)^4)$  time.

**Theorem 4.** *There exists a path approximation graph  $G_\epsilon$  of size  $O((\delta n)^2)$ , that can be constructed in  $O(n(\delta n)^4)$  time, and can be used to report in  $O(k(\delta n)^4)$  time a  $14k$ -link path that  $(1 + \epsilon)$ -approximates a  $k$ -link shortest path.*

### 3 Optimal 1-Links: A Sum of Fractionals Approach

To compute 1-link shortest paths, we adapt an algorithm for minimizing a sum of linear fractional functions (SOLF) [8], to the sum of fractional functions (SOF) problem that describes an optimal 1-link path. In order to find a 1-link shortest path one needs to solve a number of optimization problems of the form  $\min_{x \in S} \{\sum_{i=1}^m \sqrt{1+x^2} \frac{a_i}{b_i x + c_i}\} = \min_{x \in S} \{\sum_{i=1}^m r_i(x)\}$ , where  $b_1 = 0$ ,  $b_i = 1$ ,  $i = 2, 3, \dots, m$ ,  $a_i, c_i$  are constants and  $b_i x + c_i > 0$  over  $S$ ,  $i = 1, 2, \dots, m$ . Thus, the functions we try to optimize are 1-dimensional SOFs with generic term  $r_i(x) = \sqrt{1+x^2}(a_i/(b_i x + c_i))$  rather than 1-dimensional SOLFs, where the generic term would have the form  $r_i(x) = a_i/(x + c_i)$ . While in general one may not be able to apply the  $d$ -dimensional SOLF algorithm in [8] for a SOF problem, we will show below that this is possible for our objective function. Our choice of method is based on the results in [4], which show that the one-dimensional SOLF algorithm is very fast in practice. Since our function is a one-dimensional SOF, adapting the SOLF method for SOF functions may lead to similar results.

The only place in the SOLF algorithm where the expression of the ratio  $r_i(x)$  is important is in the optimization subproblems that require to minimize (or maximize)  $r_i(x)$  over a convex domain (an interval in our case). For a 1-dimensional SOLF, this reduces to minimizing a linear function over an interval and thus takes constant time.

**Lemma 9.** *The function  $r(x) = \sqrt{1+x^2} \frac{a_i}{b_i x + c_i}$  is unimodal if  $b_i x + c_i > 0$  (or  $b_i x + c_i < 0$ ), with extremal value obtained at  $x^* = 1/c_i$ .*

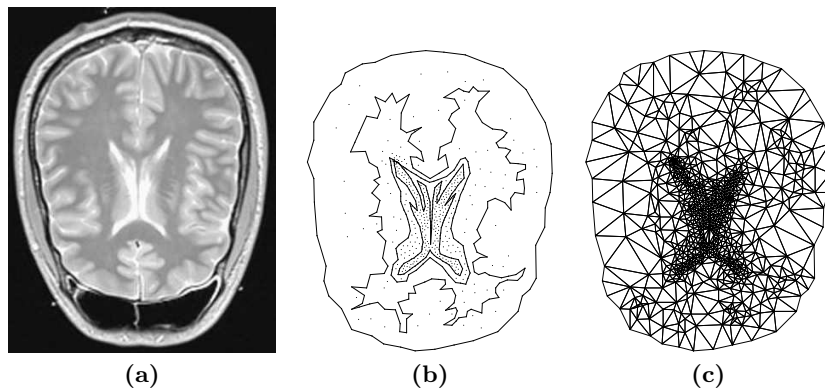
From Lemma 9 it follows that for the SOF problems associated with the 1-link shortest path, the optimization subproblems for  $r_i(x)$  can be solved exactly in constant time each and thus we can apply the SOLF algorithm for these SOFs. As shown in [4], an iteration of the algorithm can be implemented to run

in  $O(m)$  time for the 1-dimensional case, while some special steps (executed in case of a stall) require altogether  $O(m^2)$  time.

One way to speed up the computation is to process each of the SOF problems in turn, temporarily suspending the processing of the current SOF before  $k$  has been incremented (i.e. before the execution of Step 5). Each time an upper or lower bound is updated, we can use the new bound to remove or cull SOF problems from the problem space. Experimental results suggest this culling process very quickly removes many subproblems that will not lead to an optimal solution. A hybrid implementation where the subdivision algorithm in [6] is applied to stalled regions would also fit into this framework.

## 4 Implementation and Experiments

We have implemented two algorithms for solving the weighted region 1-link shortest path problem. The first one is based on the prune-and-search scheme in [6] and the second one is based on the SOF algorithm in Section 3. Our results show that both algorithms are fast on random generated subdivisions. To ensure robustness for the special cases when a source-to-target link is close to an edge or vertex of  $R$  our implementation uses the CORE library [5].

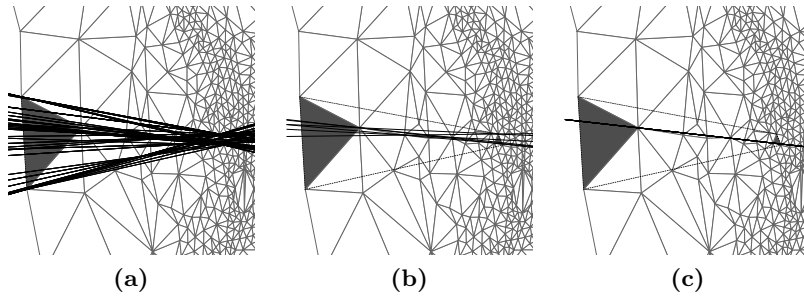


**Fig. 6.** (a) The original transverse CT scan, (b) a trace of structural elements in the scan and (c) the triangulation of that structure.

We exemplify our software package on a problem inspired by biomedicine. Consider the CT scan in Fig. 6 (a) which was taken from the Visible Human Project [17]. The source image is made up of pixels which are samples from the CT process. Fig. 6 (b) shows one possible “trace” of this data which emphasizes certain structures. Using Shewchuk’s triangulator each region can be tessellated into triangles using different constraints [18]. This is illustrated in Fig. 6 (c) where we have chosen to make the area constraints on the interior regions greater than those of the exterior region.

Fig. 7 shows the source and target regions we chose for this example. There are over 1500 triangles in this mesh, but far fewer are of interest after the bound-

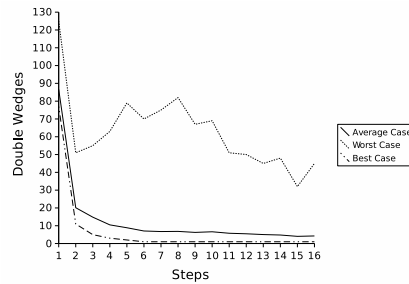
ing box is applied. The prune-and-search technique begins by enumerating possible optimization subproblems, with each subproblem corresponding to a double wedge through a vertex of  $R$ . The initial double wedges are represented as dotted lines in Fig. 7 (a). 37 wedges are found initially for the example in Fig. 7 but after only one round of culling based on the upper and lower bounds for each subproblem, there are only four wedges at step two (see Fig. 7 (b)). The number of wedges continues to grow and shrink in next steps, and for our example the algorithm terminates in twenty one steps (see Fig. 7 (c)).



**Fig. 7.** Prune-and-search progress after (a) zero, (b) one, and (c) twenty one steps.

While in general this algorithm performs well, the difficulty in assessing it is that it has very good best-case behavior and very bad worst-case behavior. In the worst case, the algorithm may continue subdividing a double wedge region over which the value of the objective function changes extremely slowly, such that subdivided double wedges cannot be culled. Fig. 8 shows an example of the best, worst and average number of subproblems competing in each step.

In applying the SOF technique to the same problem we find similar variabilities in performance. In some instances the SOF technique approaches a solution faster than the subdivision approach and in other instances the SOF technique stalls and fails to progress. If the SOF algorithm stalls, it is often necessary to run the SOF algorithm recursively on a subdivided problem. Unfortunately it is possible the SOF algorithm will then continue to stall. When the SOF algorithm was applied directly to each of the subproblems in the series, we found that 61.71% of the subproblems stalled. An average of 23.9 iteration steps and 7.64 recursive calls were required to solve each subproblem. If we consider only the subproblems which did not stall, we find that on average only 2.4 iterations were required. One improvement that has shown



**Fig. 8.** The wedge count at each step represents the number of active problems.

some success is the adoption of a hybrid solution where a SOF approach is used until a stall is detected and then in that case we revert to the subdivision technique in order to avoid a recursive stall. However, more experiments are needed to decide which of the two algorithms performs better in practice.

## References

1. L. Aleksandrov, A. Maheshwari, and J.-R. Sack. Determining approximate shortest paths on weighted polyhedral surfaces. *Journal of the ACM*, 52(1):25–53, 2005.
2. D. Z. Chen, O. Daescu, X. Hu, X. Wu, and J. Xu. Determining an optimal penetration among weighted regions in two and three dimensions. *Journal of Combinatorial Optimization*, 5(1):59–79, 2001.
3. D. Z. Chen, X. Hu, and J. Xu. Optimal beam penetration in two and three dimensions. *Journal of Combinatorial Optimization*, 7(2):111–136, 2003.
4. D. Z. Chen, J. Xu, N. Katoh, O. Daescu, X. Wu, and Y. Dai. Efficient algorithms and implementations for optimizing the sum of linear fractional functions, with applications. In *Journal of Combinatorial Optimization*, 9(1):69–90, 2005.
5. CORE library project. <http://www.cs.nyu.edu/exact/core>.
6. O. Daescu. Improved optimal weighted links algorithms. In *Proc. 2nd Internat. Workshop on Computational Geometry and Applications*, pages 65–74, 2002.
7. O. Daescu and J. Luo. Proximity problems on line segments spanned by points. In *14th Annual Fall Workshop on Computational Geometry*, Nov 2004.
8. J. E. Falk and S. W. Palocsay. Optimizing the sum of linear fractional functions. In *Recent advances in global optimization*, pages 221–258, 1992.
9. L. Gewali, A. Meng, J. S. B. Mitchell, and S. Ntafos. Path planning in  $0/1/\infty$  weighted regions with applications. *ORSA J. on Computing*, 2(3):253–272, 1990.
10. J. Krozel, C. Lee, and J. S. B. Mitchell. Estimating time of arrival in heavy weather conditions. In *Proc. AIAA Guidance, Navig., and Control*, pages 1481–1495, 1999.
11. M. Lanthier, A. Maheshwari, and J.-R. Sack. Approximating shortest paths on weighted polyhedral surfaces. *Algorithmica*, 30(4):527–562, 2001.
12. C. Mata and J. Mitchell. A new algorithm for computing shortest paths in weighted planar subdivisions. *Proc. 13th Ann. Symp. Comput. Geom.*, pages 264–273, 1997.
13. C. B. McCubbin, C. D. Piatko, A. V. Peterson, and C. R. Donnald. Cooperative organic mine avoidance path planning. In *Proc. of the SPIE*, vol. 5794, 2005.
14. J. S. B. Mitchell. Geometric shortest paths and network optimization. In J.-R. Sack and J. Urrutia, eds, *Handbook of Computational Geometry*. Elsevier, 2000.
15. J. S. B. Mitchell and C. H. Papadimitriou. The weighted region problem: Finding shortest paths through a weighted planar subdivision. *Journal of the ACM*, 38(1):18–73, Jan. 1991.
16. J. S. B. Mitchell, C. Piatko, and E. Arkin. Computing a shortest  $k$ -link path in a polygon. In *Proc. 33rd IEEE Sympos. Found. Comput. Sci.*, pages 573–582, 1992.
17. National Library of Medicine. The visible human project. <http://www.nlm.nih.gov/research/visible>.
18. J. R. Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In M. C. Lin and D. Manocha, eds., *Appl. Computat. Geom.: Towards Geometric Engineering*, LNCS vol. 1148, pages 203–222. Springer-Verlag, 1996.
19. Z. Sun and J. H. Reif. Adaptive and compact discretization for weighted region optimal path finding. In *Proc. 14th Sympos. on Fundamentals of Computation Theory*, LNCS, Springer-Verlag, 2003.