On Finding Approximate Optimal Paths in Weighted Regions *

Zheng Sun [a,*], John H. Reif [b]

[a] *Department of Computer Science, Hong Kong Baptist University, Kowloon, Hong Kong*

[b] *Department of Computer Science, Duke University, Box 90129, Durham, NC 27708-0129, USA*

Abstract

The main result of this paper is an approximation algorithm for the weighted region optimal path problem. In this problem, a point robot moves in a planar space composed of n triangular regions, each of which is associated with a positive unit weight. The objective is to find, for given source and destination points s and t, a path from s to t with the minimum weighted length. Our algorithm, BUSHWHACK, adopts a traditional approach (see ([1–3]) that converts the original continuous geometric search space into a discrete graph $\mathcal{G}$ by placing representative points on boundary edges. However, by exploiting geometric structures that we call intervals, BUSHWHACK computes an approximate optimal path more efficiently as it accesses only a sparse subgraph of $\mathcal{G}$. Combined with the logarithmic discretization scheme introduced by Aleksandrov et al.[3], BUSHWHACK can compute an ϵ-approximation in $O(\frac{n}{\epsilon}(\log\frac{1}{\epsilon}+\log n)\log\frac{1}{\epsilon})$ time. By reducing complexity dependency on ϵ, this result improves on all previous results with the same discretization approach. We also provide an improvement over the discretization scheme of [3] so that the size of $\mathcal{G}$ is no longer dependent on *unit weight ratio*, the ratio between the maximum and minimum unit weights. This leads to the first ϵ-approximation algorithm whose time complexity does not depend on unit weight ratio.

Key words:
robotics, computational geometry, approximation algorithms, optimal paths

* Parts of this work were published in *Proceedings of the 4th Workshop on Foundations of Algorithmic Robotics* [4] and *Proceedings of the 14th International Symposium on Fundamentals of Computation Theory* [5].

* Corresponding author.
Email addresses: sunz@comp.hkbu.edu.hk (Zheng Sun), reif@cs.duke.edu (John H. Reif).

Preprint of paper appearing in Journal of Algorithms, 2005.

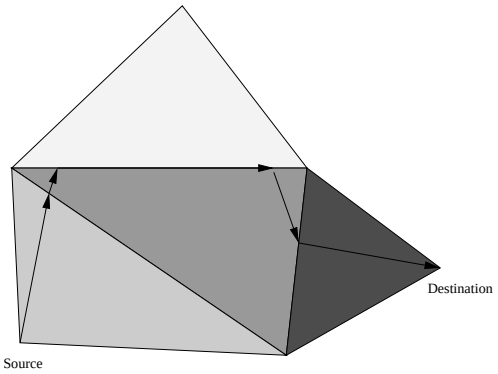


Figure 1: Optimal path in weighted regions

this problem, it is generally considered to be very hard. As a result, most of the known research works have focused on various approximation algorithms.

For any s and t, let $p_{opt}(s,t)$ be an optimal path from s to t and let $p'_{opt}(s,t)$ be an approximate optimal path from s to t computed by a given approximation algorithm. We decompose the error of the approximation into two components: the *absolute error* (or *additive error*), which is bounded by a constant disregarding the value of $\|p_{opt}(s,t)\|$; and the *relative error* (or *multiplicative error*), which is linear in $\|p_{opt}(s,t)\|$. We say that the error of the approximation is ϵ-relative and δ-absolute if $\|p'_{opt}(s,t)\| - \|p_{opt}(s,t)\| = \epsilon \cdot \|p_{opt}(s,t)\| + \delta$. In this paper we mainly study *ϵ-approximation algorithms* that guarantee to produce *ϵ-good approximate optimal paths*, that is, approximate optimal paths with no absolute error but just relative error.

In the following we will discuss our work and other related work in the context of planar weighted region optimal path problem. It is important to note that most of the results on weighted region optimal path problem, including the ones we will present in the following sections, can also be applied to weighted polyhedral surfaces.

1.2 Previous Works

We first define some parameters that will be used in the complexity analysis of various approximation algorithms. We let n be the number of triangular regions in the plane, and let L be the length of the longest boundary edge of all regions. For our convenience, we assume that the coordinates of the vertices are all non-negative integers, and let N be the maximum coordinate. We use w_{max} (w_{min}, respectively) to denote the maximum (minimum, respectively) unit weight among all regions, and define the *unit weight ratio* μ to be $\frac{w_{max}}{w_{min}}$.

Table 1: Performance Parameters

ϵ	the user-defined relative error allowed
n	number of regions
L	length of the longest edge
N	maximum coordinate of the vertices
w_{max} (w_{min})	the maximum (minimum) unit weight
μ	the unit weight ratio
θ_{min}	minimum angle between two adjacent boundary edges of any region

One of the earliest ϵ-approximation algorithms on this problem was provided by Mitchell and Papadimitriou [11]. Their algorithm uses "Snell's Law of Refraction" and the continuous Dijkstra method to give an optimal-path map for a given source point s. The time complexity of their algorithm is $O(n^8 \log \frac{nN\mu}{\epsilon})$. In practice, however, the time complexity is expected to be much lower.

Later Mata and Mitchell [9] presented an approximation algorithm based on constructing a "pathnet graph" of size $O(nk)$ by tracing k evenly-spaced "refraction rays" (rays that obey Snell's Law) from each vertex. This graph is guaranteed to contain an ϵ-approximation of an optimal path, where $\epsilon = O(\frac{\mu}{k\theta_{min}})$. The time complexity, in terms of ϵ, n and other geometric parameters, is $O(\frac{n^3\mu}{\theta_{min}\epsilon})$. The advantage of this algorithm, as compared to Mitchell and Papadimitriou's algorithm [11], is that the data structure allows for efficient computation of approximate optimal paths between any pair of source and destination points.

In the same work Mata and Mitchell also provided an alternative algorithm using edge subdivision. This algorithm discretizes the original continuous space by placing with even spacing m representative points (which we call *Steiner points*) along each boundary edge. Then it constructs a discrete graph $\mathcal{G}$ that includes these Steiner points as well as the vertices of the triangular subdivision. For any two points (vertices or Steiner points) v_1 and v_2 in $\mathcal{G}$ that are on the border of the same region, an edge (v_1, v_2) is added into $\mathcal{G}$ with an assigned weight of the weighted length of $|\overline{v_1v_2}|$. Each path in $\mathcal{G}$, which we call *discrete path*, corresponds to a path with the same cost in the original space. An *optimal discrete path* (the minimum-cost path among all discrete paths in $\mathcal{G}$) from s to t is then computed in $O(nm^2 + nm\log nm)$ time using Dijkstra's algorithm. Depending on the quality of the discretization, this optimal discrete path usually gives a good approximation to a *real* optimal path from s to t. Lanthier *et al.* [7] independently provided a $O(n^5)$ algorithm using a uniform discretization that guarantees an absolute error of $O(Lw_{max})$ by choosing $m = n^2$. This uniform discretization method was also used earlier by Smith *et al.* [17].

Aleksandrov *et al.* [1, 2] later provided two logarithmic discretization methods. For a given ϵ, either of the two discretization methods constructs, by placing $m = O(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$ Steiner points on each boundary edge, a graph $\mathcal{G}_\epsilon$ that can guarantee to contain an ϵ-good approximate optimal path from s to t. Therefore, it takes $O(\frac{n}{\epsilon}(\frac{1}{\epsilon}\log\frac{1}{\epsilon} + \log n)\log\frac{1}{\epsilon})$ time to find an ϵ-good approximate optimal path using any of the two discretization methods. Similar to the other algorithms, the complexity also involves a number of geometric parameters such as $w_{min}, w_{max}, \theta_{min}$, etc.

The time complexities of the above edge-subdivision-based algorithms are rather high, as $\mathcal{G}$ is a dense graph with $O(nm^2)$ edges. To reduce the cost of processing the edges of $\mathcal{G}$, alternative algorithms that only access a subgraph of $\mathcal{G}$ were investigated. For example, with the uniform discretization, Lanthier *et al.* [7] proposed an algorithm that only computes an optimal discrete path in an ϵ-spanner of $\mathcal{G}$. The time complexity is reduced to $O(\frac{n^3}{\epsilon} + n^3\log n)$, as each point is selectively connected to $O(\frac{1}{\epsilon})$ other points. In addition to the absolute error of $O(Lw_{max})$ introduced by the discretization, the approximate optimal path found by this algorithm also has a relative error of ϵ introduced by using the spanner.

The best result is provided by Aleksandrov *et al.* [2]. Their pruned Dijkstra's algorithm uses two stages to trim the edges of $\mathcal{G}$. First, by exploiting the Snell's Law they define a *geodesic cone* for each Steiner point v, claiming that any optimal discrete path passing through v can only leave v via an edge inside this cone. The second stage is to construct an ϵ-spanner for the geodesic cone of each Steiner point. The number of incident edges of each Steiner point is therefore reduced to $O(\frac{1}{\sqrt{\epsilon}})$, resulting in an $O(\frac{n}{\epsilon}(\frac{1}{\sqrt{\epsilon}} + \log n)\log\frac{1}{\epsilon})$ algorithm. Although the ϵ-spanner causes an additional term of error, the approximate optimal path computed is still $O(\epsilon)$-good.

In Table 2, we list the time complexity and error bound of each of the above-mentioned algorithms for a comparison. For our convenience, we omit all geometric parameters and state the complexity of each algorithm in terms of n and ϵ.

Table 2: Complexity Comparison

Algorithm	Complexity	Error
Mitchell and Papadimitriou [11]	$O(n^8\log\frac{n}{\epsilon})$	ϵ-relative
Lanthier et al. [7]	$O(n^5$	$O(Lw_{max})$-absolute
Lanthier et al. [7]	$O(\frac{n^3}{\epsilon} + n^3\log n)$	ϵ-relative and $O(Lw_{max})$-absolute
Mata and Mitchell [9]	$O(\frac{n^3}{\epsilon})$	ϵ-relative
Aleksandrov et al. [1]	$O(\frac{n}{\epsilon}(\frac{1}{\epsilon} + \log n)\log\frac{1}{\epsilon})$	ϵ-relative
Aleksandrov et al. [2]	$O(\frac{n}{\epsilon}(\frac{1}{\sqrt{\epsilon}} + \log n)\log\frac{1}{\epsilon})$	ϵ-relative

For other related work, see [3, 4, 5, 6, 12, 13, 15, 16], or see survey [10].

2 Preliminaries

2.1 Notations

Let S be a planar space consisting of n triangular regions. We use V and E to denote the set of vertices and the set of boundary edges, respectively, of all regions. Therefore, $|V| = O(n)$, and $|E| = O(n)$. For any region r, we use w_r to denote the unit weight of r. For each boundary edge $e \in E$, the unit weight w_e is defined to be $\min\{w_r, w_{r'}\}$, where r and r' are the two regions incident to e. For any v_1 and v_2 in a region r, we define $d_r(v_1, v_2) = w_r \cdot |\overline{v_1 v_2}|$ to be the *region distance* between v_1 and v_2. Similarly, we say that $d_e(v_1, v_2) = w_e \cdot |\overline{v_1 v_2}|$ is the *edge distance* between v_1 and v_2 for any v_1 and v_2 on boundary edge e. Let s and t be the designated source point and destination point, respectively. We assume that $s, t \in V$. If otherwise, we can add a constant number of boundary edges to construct from S a new triangular decomposition S' that has s and t as vertices.

It has been proved (see [11]) that a weighted region optimal path is a piecewise linear path consisting of $O(n^2)$ segments. The two endpoints of each segment are on the boundary of the same region. A segment of a path is said to be "edge-crawling" if it lies on a boundary edge. Similarly, a segment is "face-crossing" if it cuts through a region. We call a path a "face-crossing" ("edge-crawling") path if the last segment of the path is "face-crossing" ("edge-crawling", respectively).

For any two points v_1, v_2 on a path p, let $p[v_1, v_2]$ denote the part of p between v_1 and v_2. For two paths p_1 and p_2, we let $p_1 + p_2$ denote the concatenation of p_1 and p_2. If $p = p_1 + p_2$, we say that p is an *extension* of p_1. In particular, if $p = p_1 + \overline{v_1 v_2}$, we call p a *one-segment extension* of p_1. For any two points x and y, we use $p(x, y)$ to denote a path from x to y and use $p_{opt}(x, y)$ to denote an optimal path from x to y. We define the "distance" from s to t, $d_{opt}(s, t)$, to be the cost of $p_{opt}(s, t)$.

Following the same approach used by [7, 1, 2], we discretize the $2\mathcal{D}$ space by introducing Steiner points on boundary edges. For each boundary edge $e \in E$, we add m Steiner points on e for some positive integer m. Let V_s be the set of Steiner points and let $V' = V \bigcup V_s$. A weighted graph $\mathcal{G}(V', E')$ is constructed by interconnecting points in V' that are on the boundary of the same region. Each edge (x, y) in $\mathcal{G}$ is assigned a weight $\omega(x, y)$, which is defined to be $d_e(x, y)$ if $\overline{xy}$ is an edge-crawling segment on boundary edge e; or $d_r(x, y)$ if $\overline{xy}$ is a face-crossing segment in region r.

By constructing $\mathcal{G}$, the original path planning problem in a continuous space is transformed to the problem of finding an optimal discrete path connecting s and t in $\mathcal{G}$. This optimal discrete path is then used to approximate a real optimal path in the original continuous space. As the goal of our algorithm is to find an optimal discrete path, in the following discussion, wherever we say a discrete path is "optimal," we mean that it is optimal among all discrete paths. The phrases "optimal discrete path" and "approximate optimal path" are used interchangeably, and are both denoted by $p'_{opt}(s, t)$. Let $d'_{opt}(s, t)$ be the cost of $p'_{opt}(s, t)$. At any time during the search of an optimal discrete path from s to t, a point v is said to be *discovered* if and only if $d'_{opt}(s, v)$ is determined. For each boundary edge e we use PLIST_e to denote the list that includes all discovered points on e.

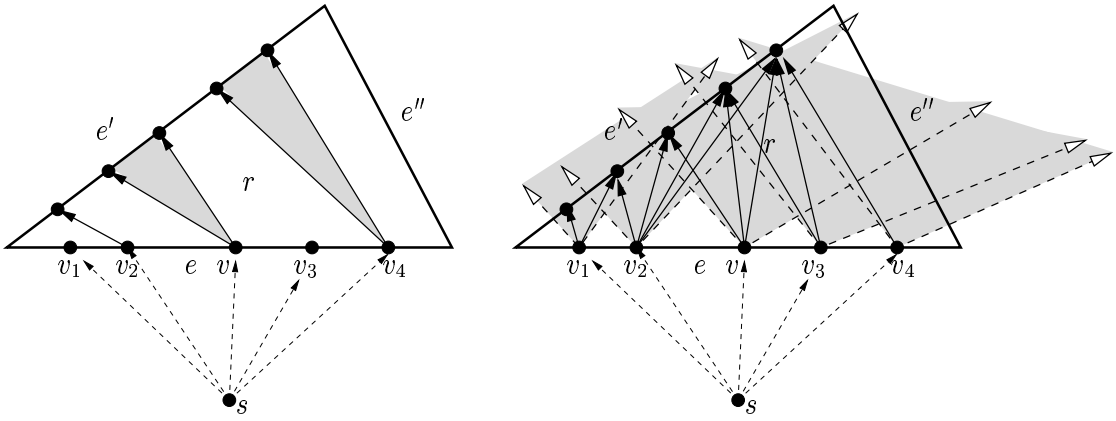
2.2 Our Approach

2.2.1 BUSHWHACK Algorithm

Instead of applying Dijkstra's algorithm to the resulting graph $\mathcal{G}$, our approximation algorithm uses a new discrete search algorithm called BUSHWHACK. The BUSHWHACK algorithm can compute optimal discrete paths on both uniform and logarithmic discretizations more efficiently than the ones proposed in [7] and [1, 2]. For each Steiner point v, BUSHWHACK dynamically maintains a small set of incident edges of v that may contribute to an optimal discrete path from s to t. If m Steiner points are placed on each boundary edge, during the entire computation the number of edges accessed in each region is $O(m \log m)$, and therefore the complexity of the algorithm is reduced to $O(nm \log nm)$, as compared to $O(nm^2 + nm \log nm)$ of Dijkstra's algorithm.

Let r be a weighted region and let e, e' be two boundary edges of e. For each point $v \in e$, we define an *interval* $I_{v,e,e'}$ to be a set of contiguous Steiner points such that, for any point $v' \in e'$, edge $\overline{vv'}$ may be "used" by an optimal discrete path originated from s only if v' is inside the interval. Each point $v' \in I_{v,e,e'}$ is associated with an *almost-optimal path* from s to v' that passes through v. This path is an concatenation of an optimal discrete path from s to v and segment $\overline{vv'}$. Moreover, this path is the least costly path among all paths from s to v' that enter region r through Steiner points on e.

The concept of intervals is similar to the geodesic cones used by Aleksandrov *et al.* [2]. The main difference is that, for any two points v and v_1 on edge e, the intervals associated with v and v_1 are mutually exclusive, as it will be clear later when we give the formal definition for intervals. Therefore, on average the number of Steiner points in each interval is $O(1)$, assuming that e and e' contain roughly the same number of Steiner points. The geodesic cones, however, may overlap with each other as shown in Figure 2.b. In fact, using Aleksandrov *et al.*'s discretization scheme [2], which places $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$ Steiner points on each boundary edge, the number of Steiner points of even an ϵ-spanner of each geodesic cone can only be bounded by $O(\frac{1}{\sqrt{\epsilon}})$.



(a) On average each interval contains $O(1)$ Steiner points.

(b) Geodesic cones may overlap and each contains $O(\frac{1}{\sqrt{\epsilon}})$ Steiner points.

Figure 2: Comparison between interval and geodesic cone

Our definition of intervals is actually more related to that of the intervals of optimality used by Mitchell and Papadimitriou [11]. For the clarity of the following discussion, we call their intervals of optimality *continuous intervals* and our intervals *discrete intervals*. Each interval (continuous or discrete) consists of points to which the optimal paths have the same discrete structure. Each continuous interval describes a segment of a boundary edge so that the optimal paths to the points in the segment go through the same sequence of boundary edges. Each discrete interval describes the set of Steiner points on a boundary edge so that the almost-optimal paths to these points go through the same sequence of bending points. The difference is that, while continuous intervals are treated in the continuum, discrete intervals are sets of discrete points.

Because of the difference in the definitions, our data structures and procedures for processing the discrete intervals are totally different from those found in Mitchell and Papadimitriou's algorithm, although in some cases we follow the terminologies used in [11]. Because almost-optimal paths to the points in each discrete interval go through the same sequence of bending Steiner points (not just boundary edges), we do not need to compute reverse ray tracing, which is done by the `Find-Point` procedure in Mitchell and Papadimitriou's algorithm. Reverse ray-tracing is a very costly operation that takes $O(n^4 \log\frac{1}{\epsilon})$ time.

The number of continuous intervals is $\Omega(n^4)$ in the worst case, while the number of discrete intervals is proportional to the number of Steiner points, which is $O(\frac{n}{\epsilon}\log\frac{1}{\epsilon})$ if the logarithmic discretization of [2] is used. As a result, the complexity of our algorithm is more dependent on ϵ, while that of Mitchell and Papadimitriou's algorithm is more dependent on n. It is worth mentioning that, although in the worst case the dependency of Mitchell and Papadimitriou's algorithm on n is very high ($O(n^8)$), their algorithm appears

to be the only one that has logarithmic dependency on ϵ and other geometric parameters.

One key difference between our algorithm and the approximation algorithms of [2] and [11] is that, while these previous algorithms exploit the fact that an optimal path obeys Snell's Law when crossing boundary edges, our algorithm only uses the simple property that two optimal paths originated from the same source point cannot intersect in the interior of any region. Therefore, our algorithm is more flexible, allowing for many other heuristic cost criteria (e.g., charging a varying cost on crossing boundary edges) to be added easily. Sun and Reif [18] showed that with minor modifications this algorithm can be applied to the time-optimum movement planning problem in the presence of flows [14], as well as the shortest anisotropic path problem [8, 19].

Maintaining and updating interval information for each point v involves an extra cost of accessing $O(\log m)$ incident edges of v. Therefore, on average, our algorithm uses $O(\log m)$ edges for each point in $\mathcal{G}$. When the uniform discretization of [7] is used, our algorithm improves the time complexity to $O(n^3 \log n)$ (as opposed to $O(\frac{n^3}{\epsilon} + n^3 \log n)$ of [7]), while eliminating the ϵ-relative factor in the approximation error. With the logarithmic discretization of [2], our algorithm can find an ϵ-good approximate optimal path in $O(\frac{n}{\epsilon}(\log \frac{1}{\epsilon} + \log n) \log \frac{1}{\epsilon})$ time. This result improves on the $O(\frac{n}{\epsilon}(\frac{1}{\epsilon} + \log n) \log \frac{1}{\epsilon})$ algorithm of [1] or the $O(\frac{n}{\epsilon}(\frac{1}{\sqrt{\epsilon}} + \log n) \log \frac{1}{\epsilon})$ algorithm of [2] by at least a factor of $O(\frac{1}{\sqrt{\epsilon}} / \log \frac{1}{\epsilon})$. Our algorithm, however, does not improve the complexity with respect to n.

2.2.2 Compact Discretization Scheme

Table 2 lists for each algorithm the time complexity with respect to n and ϵ. To our best knowledge, the time complexities of all previous ϵ-approximation algorithms also depend on the unit weight ratio μ, either linearly or logarithmically. This dependency is caused by the corresponding discretization scheme used. In particular, the discretization scheme of Aleksandrov *et al.* [2] constructs for a given ϵ a graph $\mathcal{G}_\epsilon$ with $O(\frac{1}{\epsilon} \log \frac{1}{\epsilon} \log \mu)$ Steiner points placed on each boundary edge. Here again we omit the other geometric parameters.

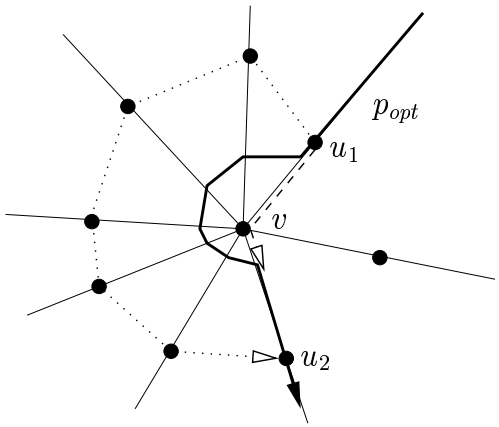


Figure 3: "Bad" path

The main obstacle for removing the dependency on μ from the size of $\mathcal{G}_\epsilon$ is that otherwise it is difficult to prove that for each optimal path p_{opt} there exists in $\mathcal{G}_\epsilon$ a discrete path that is an ϵ-approximation of p_{opt}. One traditional proof technique used in proving the existence of such a discrete path is to decompose p_{opt} into k subpaths $p_1, p_2, \cdots, p_k$ and then construct a discrete path $p' = p'_1 + p'_2 + \cdots + p'_k$ such that $\|p'_i\| \leq (1+\epsilon)\|p_i\|$ for each i. Ideally, we could choose each p'_i such that p_i and p'_i lie in the same region, and therefore the discretization just needs to make sure that $|p'_i| \leq (1+\epsilon)|p_i|$.

However, due to the discrete nature of $\mathcal{G}_\epsilon$, it is not always possible to find such p'_i for each p_i. For example, as shown in Figure 3, p_{opt} could cross a series of boundary edges near a vertex v. The point where it crosses each boundary edge e is between v and the closest Steiner point from v on e. To approximate p_{opt} by a discrete path, one way is to replace $p_{opt}[u_1, u_2]$, the subpath of p_{opt} between u_1 and u_2, by the dotted path from u_1 to u_2. However, each p'_i could be much longer than the corresponding p_i, thus making it impossible

to bound $\|p'_{opt}\|$ the way we described above. Instead, in the error bound proof of Aleksandrov *et al.* [1, 2] subpath $p_{opt}[u_1, u_2]$ is replaced by the dashed path that goes through v. In that case, p'_i could travel in regions different from where p_i lies in, and therefore to bound $\|p'_i\|$ with respect to $\|p_i\|$, the discretization scheme has to take into consideration variance of unit weights.

By modifying the above proof technique, we provide in Section 6 an improvement on the discretization scheme of Aleksandrov *et al.* [2]. The number of Steiner points inserted by this new discretization scheme is $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$, with the dependency on other geometric parameters unchanged. Combining BUSHWHACK with this discretization scheme, we can have the first ϵ-approximation algorithm whose time complexity is not dependent on μ.

3 Intervals

3.1 Disadvantage of Dijkstra's Algorithm

The BUSHWHACK algorithm works similarly to Dijkstra's algorithm. It keeps a sorted list QLIST of *candidate optimal paths*. At each step, the candidate optimal path p_{min} with the minimum cost is extracted from QLIST. Consequently, a number of candidate optimal paths, which are one-segment extensions of p_{min}, are inserted into QLIST. We call this process "path propagation." The iteration continues until the destination point is reached.

Dijkstra's algorithm can be used to compute an optimal discrete path in an arbitrary weighted graph, while the aforementioned graph $\mathcal{G}$ is derived from a $2\mathcal{D}$ space consisting of weighted regions. Therefore, directly applying Dijkstra's algorithm to $\mathcal{G}$ does not fully utilize the underlying geometric properties.

For example, in Figure 4 there are two paths originated from the same source point s. Dijkstra's algorithm will consider both paths as potential optimal discrete paths for their respective destination points. However, it is not possible that two optimal discrete paths intersect inside a weighted region, as stated by the following lemma.

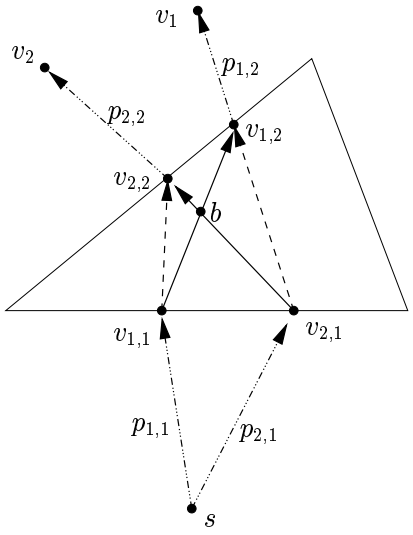


Figure 4: Optimal discrete paths cannot intersect inside region.

Lemma 1 *In a $2\mathcal{D}$ space consisting of weighted regions, any two optimal discrete paths with the same source point cannot intersect in the interior of any region.*

Proof Assume that two optimal discrete paths $p'_{opt}(s, v_1)$ and $p'_{opt}(s, v_2)$ intersect at b inside a triangular region r, as shown in Figure 4. For each $i = 1, 2$, let $\overline{v_{i,1}v_{i,2}}$ be the segment of $p'_{opt}(s, v_i)$ that contains b. Thus, each $v_{i,j}$ is a point on the boundary of r. Let $p_{i,1} = p'_{opt}(s, v_i)[s, v_{i,1}]$ and let $p_{i,2} = p'_{opt}(s, v_i)[v_{1,2}, v_i]$.

Therefore, $p_{2,1}+\overline{v_{2,1}v_{1,2}}+p_{1,2}$ is a path from s to v_1 and $p_{1,1}+\overline{v_{1,1}v_{2,2}}+p_{2,2}$ is a path from s to v_2. However, the total cost of the above two paths is less than that of $p'_{opt}(s,v_1)$ and $p'_{opt}(s,v_2)$, as $d_r(v_{1,1},v_{1,2})+d_r(v_{2,1},v_{2,2}) > d_r(v_{1,1},v_{2,2}) + d_r(v_{2,1},v_{1,2})$. Therefore, one of $p'_{opt}(s,v_1)$ and $p'_{opt}(s,v_2)$ must not be optimal. This is a contradiction. ∎

The fact that line segments $\overline{v_{1,1}v_{1,2}}$ and $\overline{v_{2,1}v_{2,2}}$ intersect inside region r indicates that these two line segments cannot both contribute to optimal discrete paths originated from s. If we can identify *useful* line segments (*i.e.*, those that may contribute to optimal discrete paths) from *useless* line segments, we will be able to avoid an explicit construction of the entire graph $\mathcal{G}$.

3.2 Intervals

To keep track of useful line segments, we introduce a data structure that we call *interval*.

Definition 1 *Let r be a region of S and let e, e' be two boundary edges of r. For any discovered point $v \in e$ that is not incident to e', the interval $I_{v,e,e'}$ is defined to be*

$$I_{v,e,e'} = \left\{v^* \in e' \mid d'_{opt}(s,v) + d_r(v,v^*) \leq d'_{opt}(s,v') + d_r(v',v^*) \forall\ v' \in PLIST_e\right\}.$$

That is, for any point v^* on e', $v^* \in I_{v,e,e'}$ if and only if path $p'_{opt}(s,v) + \overline{vv^*}$ is the least costly path among all paths from s to v^* that are one-segment extensions of optimal discrete paths from s to discovered points on e. In case a point v^* on e' is included in two or more intervals according to this definition, we will choose only one of these intervals to contain v^*. This is to keep intervals $I_{v,e,e'}$ for all $v \in \text{PLIST}_e$ mutually exclusive subsets of e'. The way e' is partitioned into intervals is analogous to the way a space is partitioned into regions in a Voronoi diagram.

It is important to note that each interval $I_{v,e,e'}$ is a dynamic set of points on e'. It is first created when v is discovered. When more points on e are discovered, PLIST_e will contain more points and thus $I_{v,e,e'}$ may also be changed, according to the definition.

For any edge e and e' that share a region r, we use $\text{ILIST}_{e,e'}$ to denote the list of intervals $I_{v,e,e'}$ for all $v \in \text{PLIST}_e$, sorted by the Euclidean distance between v and the shared endpoint of e and e'. Lemma 1 implies that each interval is composed of consecutive points on e' (which leads us to name this data structure "interval"). Further, an interval $I_{v,e,e'}$ is located to the left (right, respectively) of another interval $I_{v',e,e'}$ on e' if and only if v is located to the left (right, respectively) of v' on e. Therefore, the order of discovered points on e is exactly the same as the order of their corresponding intervals in $\text{ILIST}_{e,e'}$, as shown in Figure 5.

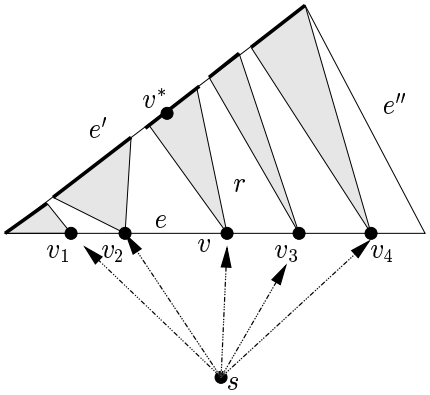


Figure 5: Intervals

When a Steiner point v on boundary edge e is newly discovered, for each boundary edge e' that shares a region r with e, an interval $I_{v,e,e'}$ needs to be created and inserted into $\text{ILIST}_{e,e'}$. In case the newly discovered point v is a vertex of a triangular region, an interval needs to be created for every boundary edge

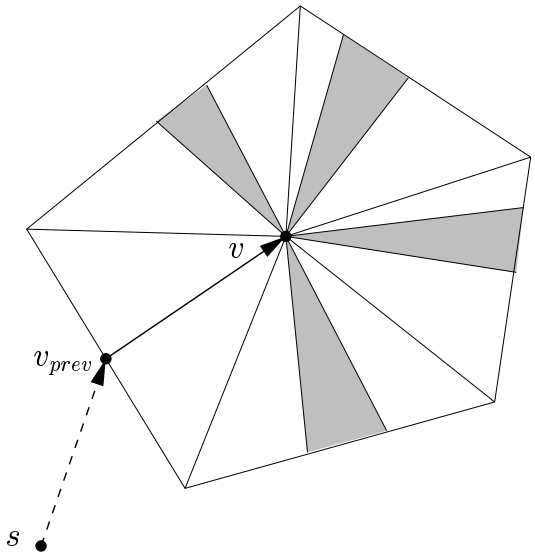


Figure 6: Creating intervals for a vertex

facing v, as shown in Figure 6. The range of the new interval $I_{v,e,e'}$, *i.e.*, the two endpoints of the interval, can be computed efficiently in $O(\log m)$ time.

If the new interval $I_{v,e,e'}$ is not empty, the ranges of the neighboring intervals need to be readjusted. Figure 7 illustrates how a new interval could *totally deplete* or *partially deplete* existing intervals. As shown in the figure, the left and right endpoints are determined for the new interval associated with v_{new}, the most recently discovered point. The new interval covers entirely $I_{v_1,e,e'}$, the interval associated with v_1, and thus $I_{v_1,e,e'}$ is totally depleted by the new interval. The interval associated with v_2 is partially depleted by the new interval as it loses part of Steiner points in it. Totally depleted intervals will be removed from $\text{ILIST}_{e,e'}$, while the range of each partially depleted interval will be readjusted.

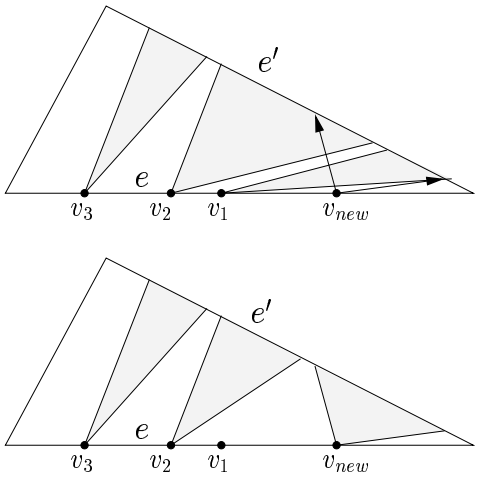


Figure 7: Inserting an interval

From Definition 1 we can conclude that, for any discovered point v on e and any point v^* on e', $\overline{vv^*}$ cannot be part of any optimal discrete path originated from s that enters region r through point v if $v^* \notin I_{v,e,e'}$. Therefore, by maintaining $\text{ILIST}_{e,e'}$ for each e and e' that share a region, the BUSHWHACK algorithm is able to avoid accessing most of the useless edges in $\mathcal{G}$.

3.3 Monotonic Intervals

For each $v^* \in I_{v,e,e'}$, line segment $\overline{vv^*}$ *may* be a useful segment and therefore we need to check the optimality of the face-crossing path $p'_{opt}(s,v) + \overline{vv^*}$. We call such a path a *direct interval path* associated with $I_{v,e,e'}$. One strategy is to insert all these paths into QLIST simultaneously when v is discovered (and $I_{v,e,e'}$ is created). However, this may not be efficient as both $\text{ILIST}_{e,e'}$ and intervals in $\text{ILIST}_{e,e'}$ are dynamic data structures. Whenever a new point on e is discovered, a new interval (although possibly empty) will be created and inserted into $\text{ILIST}_{e,e'}$. If the new interval is not empty, the ranges of the two neighboring intervals will be adjusted.

Therefore, even though a point v^* originally is in $I_{v,e,e'}$, after a new point $v_{new} \in e$ is discovered, v^* may fall into the range of the new interval $I_{v_{new},e,e'}$. If this is the case, path $p'_{opt}(s,v) + \overline{vv^*}$ no longer needs to be considered as an optimal discrete path from s to v^* as it is more costly than $p'_{opt}(s,v_{new}) + \overline{v_{new}v^*}$, according to Definition 1.

A more efficient strategy is to insert direct interval paths in the following *lazy* and *best-first* manner: interval paths associated with $\text{ILIST}_{e,e'}$ are sorted in the increasing order of path cost, and a path $p'_{opt}(s,v) + \overline{vv^*}$ is inserted into QLIST only when the previous path is extracted from the list, and only if v^* is still in $I_{v,e,e'}$. This strategy will avoid inserting a path $p'_{opt}(s,v) + \overline{vv^*}$ into QLIST if v^* is later "switched" to another interval.

To achieve this, we need to sort efficiently the direct interval paths by path cost. Since these paths are all one-segment extensions of $p'_{opt}(s,v)$, we only need to sort $d_r(v,v^*)$ for all $v^* \in I_{v,e,e'}$. As each weighted region is a Euclidean space, each interval $I_{v,e,e'}$ can be split into at most two parts by the perpendicular point of v on e', so that the region distance from v to points in each part is monotonically increasing or decreasing.

One way to get a sorted list of direct interval paths associated with $I_{v,e,e'}$ is to do a merge-sort using these two parts. A simpler and more efficient way is to create an interval for each monotonic part of $I_{v,e,e'}$, as illustrated in Figure 8. The original $I_{v,e,e'}$ is thus replaced by these intervals in $\text{ILIST}_{e,e'}$. Points in each such interval are readily sorted by region distance to v. We call these intervals *monotonic intervals*.

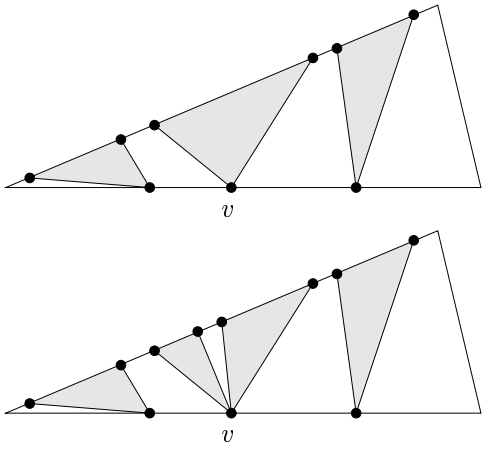


Figure 8: Splitting an interval

Although as a result of splitting the number of intervals in $\text{ILIST}_{e,e'}$ is increased, each interval is easier to manipulate. The implementation of our algorithm follows this scheme. In the following discussion, we always assume that each interval is monotonic, and points in each interval $I_{v,e,e'}$ are sorted by region distance to v.

3.4 Propagating Interval Paths

Following the notations used previously, we let e and e' be two boundary edges of region r. Suppose interval $I_{v,e,e'}$ contains points $v_1^*, v_2^*, \cdots, v_d^*$ when it is initially created (*i.e.*, when v is discovered), as shown in Figure 9. Here $v_1^*, v_2^*, \cdots, v_d^*$ are consecutive points on e' and v_1^* and v_d^* are the two endpoints of the interval.

W.L.O.G, we assume $d_r(v, v_1^*) \leq d_r(v, v_d^*)$. As interval $I_{v,e,e'}$ is monotonic, we have $d_r(v, v_1^*) \leq d_r(v, v_2^*) \leq \cdots \leq d_r(v, v_d^*)$.

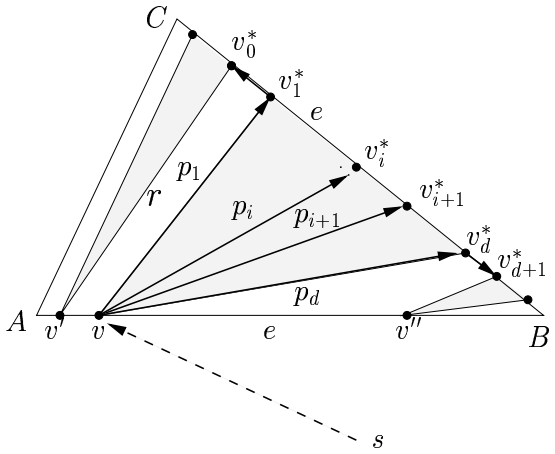


Figure 9: Propagating interval paths

Let $p_1, p_2, \cdots, p_d$ be direct interval paths associated with $I_{v,e,e'}$, where $p_i = p'_{opt}(s, v) + \overline{vv_i^*}$ for $1 \leq i \leq d$. For each v_i^*, let $P_i = \{p_{j,i} \mid 1 \leq j \leq d\}$, where $p_{j,i} = p_j + \overline{v_j^* v_i^*}$. Observe that $p_i \in P_i$ as $p_i = p_i + \overline{v_i^* v_i^*}$. All paths in P_i except p_i are *extended interval paths*; each of them extends by one segment a direct interval path associated with $I_{v,e,e'}$. We call both direct interval paths and extended interval paths *interval paths*. P_i is the set of all interval paths associated with $I_{v,e,e'}$ that connect s and v_i^*.

For each v_i^*, BUSHWHACK inserts into QLIST only one interval path p_i^* that connects s and v_i^*. This path is a *locally optimal path*. An interval path $p' \in P_i$ is said to be locally optimal if $\|p'\| = \min\{\|p\| \mid p \in P_i\}$. In the following we describe how BUSHWHACK determines these locally optimal interval paths $p_1^*, p_2^*, \cdots, p_d^*$ for $I_{v,e,e'}$. In Section 5 we will prove that these paths are indeed locally optimal.

If $w_{e'} = w_r$, for each v_i^* it is always less costly to travel directly from v to v_i^* than to first travel from v to some other point on e' and then (along edge e') to v_i^*, and therefore $p_i^* = p_i$.

Now we consider the more interesting case where $w_{e'} < w_r$. We let v_h be the perpendicular point of v on edge e' and let $h = |\overline{vv_h}|$. Since $I_{v,e,e'}$ is a monotonic interval, v_1^* is between v_h and v_d^* on e', as v_1^* precedes v_d^* in $I_{v,e,e'}$. Let u be the point on ray $\overrightarrow{v_h v_d^*}$ such that $\frac{|\overline{v_h u}| \cdot w_r}{\sqrt{|\overline{v_h u}|^2 + h^2}} = w_{e'}$. It is easy to see that, for any point u' between v_h and u, $\frac{|\overline{v_h u'}| \cdot w_r}{\sqrt{|\overline{v_h u'}|^2 + h^2}} < w_{e'}$; for any point u'' beyond u, $\frac{|\overline{v_h u''}| \cdot w_r}{\sqrt{|\overline{v_h u''}|^2 + h^2}} > w_{e'}$.

If u is between v_h and v_1^*, each p_i^* is chosen to be $p_{1,i}$ for each $i, 1 \leq i \leq d$. That is, each interval path p_i^* except p_1^* is an edge-crawling path that extends p_1 by one segment. If v_d^* is between v_h and u, each p_i^* is chosen to be p_i. In this case, every interval edge p_i^* is a face-crossing path that extends $p'_{opt}(s, v)$ by one segment.

Otherwise, u lies between v_k^* and v_{k+1}^* on edge e' for some k, $1 \leq k < d$. Then p_i^* can be determined as follows:

Definition 2 *Suppose v_k^* is the Steiner point inside interval $I_{v,e,e'}$ such that u, as defined above, is between v_k^* and v_{k+1}^*, then:*

(a) if $1 \leq i \leq k$, then p_i^* is defined to be p_i;

(b) if $i = k + 1$, then p_i^* is defined to be p_{k+1} or $p_{k,k+1}$, whichever is less costly;

(c) if $k + 1 < i \leq d$, then p_i^* is defined to be $p_{k+1}^* + \overline{v_{k+1}^* v_i^*}$.

The propagation process terminates when all points in $I_{v,e,e'}$ are reached by such interval paths. Observe that this process may be terminated before p_d^* is generated and inserted into QLIST. This would occur when another interval $I_{v',e,e'}$ is created that re-adjusts the range of $I_{v,e,e'}$.

By the BUSHWHACK algorithm, each interval $I_{v,e,e'}$ will generate no more than $O(d')$ locally optimal interval paths, where d' is the number of points inside $I_{v,e,e'}$ after all points on e are discovered. Thus, for each interval list $\text{ILIST}_{e,e'}$, only $O(m)$ interval paths are inserted into QLIST. The total number of interval paths, therefore, is $O(mn)$.

From the propagation scheme of interval paths it is clear that, once an interval $I_{v,e,e'}$ is created, at any time in path list QLIST there is exactly one interval path p associated with $I_{v,e,e'}$, until the last interval path associated with $I_{v,e,e'}$ is extracted from QLIST. We call path p the *current interval path* of $I_{v,e,e'}$ and denote it by $p_{v,e,e'}$. The Steiner point in $I_{v,e,e'}$ to which path $p_{v,e,e'}$ connects is correspondingly referred as the *current Steiner point* of $I_{v,e,e'}$ and is denoted by $v^*_{v,e,e'}$. For any point v^* preceding $v^*_{v,e,e'}$ in $I_{v,e,e'}$, a locally optimal interval path that connects s and v^* has already been extracted from QLIST. We call such a point *processed interval point* of $I_{v,e,e'}$. Any point v^* following $v^*_{v,e,e'}$ in $I_{v,e,e'}$ is an *unprocessed interval point*. According to the definitions, we can establish the following lemma.

Lemma 2 *For any interval $I_{v,e,e'}$, there is always a current Steiner point for $I_{v,e,e'}$ in QLIST until all points in $I_{v,e,e'}$ are processed.*

Recall that in Subsection 3.2 we showed that an existing interval $I_{v',e,e'}$ could be partially or totally depleted by a newly created interval $I_{v,e,e'}$. If $I_{v',e,e'}$ is totally depleted, the current interval path $p_{v',e,e'}$ no longer needs to be considered as a potential optimal discrete path. We call it an *invalid interval path*. Unlike other (valid) candidate optimal paths, an invalid interval path will not be propagated when it is extracted from QLIST.

For a partially depleted interval $I_{v',e,e'}$, if the current Steiner point $v_{v',e,e'}$ still belongs $I_{v',e,e'}$ after the new interval $I_{v,e,e'}$ is created, the current interval path $p_{v',e,e'}$ of $I_{v',e,e'}$ is still locally optimal and we will still propagate it when it is being extracted from QLIST. However, if $v_{v',e,e'}$ is now included in $I_{v,e,e'}$, $p_{v',e,e'}$ becomes an invalid interval path. Just like the case of totally depleted intervals, we will extract $p_{v',e,e'}$ from QLIST without propagation when it becomes the least costly path in QLIST.

This will, however, stop the chain of propagation of interval paths associated with $I_{v',e,e'}$, as after $p_{v',e,e'}$ is extracted from QLIST there will be no interval path in QLIST that is associated with $I_{v',e,e'}$. This causes a problem in the case that there are still unprocessed points inside the adjusted $I_{v',e,e'}$. To fix this problem, we will treat the adjusted $I_{v',e,e'}$ as a newly created interval and insert into QLIST the interval path for the first unprocessed point in $I_{v',e,e'}$. This will re-initiate the process of path propagation for the interval.

4 The BUSHWHACK Algorithm

The BUSHWHACK algorithm maintains three types of dynamic lists:

QLIST the list of candidate optimal paths sorted by path cost

PLIST$_e$ the list of discovered points on edge e

ILIST$_{e,e'}$ the list of intervals for edge e and e' that are on the boundary of the same region

As we mentioned previously, all paths can be divided into two categories, interval paths and non-interval paths. All face-crossing paths along with some edge-crawling paths are interval paths. We have shown how two non-interval paths are generated for each interval. This section will explain how other non-interval paths are created by extending these paths.

The main body of the BUSHWHACK algorithm is a loop. Each iteration, the candidate optimal path p in QLIST with the minimum cost is extracted from the list. Let v be the ending point of path p. If v is not a discovered point (*i.e.*, the distance from s to v is not yet decided), we claim that path p is an optimal discrete path from s to v, and $\|p\|$ is the distance from s to v.

1. insert path $p'_{opt}(s,s)$ into QLIST
2. **while** t is not reached
3. extract the least costly path $p(s,v)$ from QLIST
4. **if** v is a not a discovered point **then**
5. $p'_{opt}(s,v) \leftarrow p(s,v)$; $d'_{opt}(s,v) \leftarrow \|p(s,v)\|$
6. HandleNewDiscovery(v)
7. Propagate(v,p)

Algorithm 1: FindPath(s,t)

Here $p'_{opt}(s,s)$ is the zero-cost path from s to s.

Function HandleNewDiscovery(v) creates new intervals for the newly discovered point v, and then inserts into QLIST an interval path associated with each of these intervals.

1. **if** v is a Steiner point on an edge e **then**
2. **for** each region r incident to e
3. **for** each edge e' of r that is not e
4. create interval $I_{v,e,e'}$
5. **else** (v is a vertex in S)
6. **for** each edge e incident to v
7. $v_{next} \leftarrow$ the neighboring Steiner point of v on e
8. insert path $p_{new} = p + \overline{vv_{next}}$ into QLIST
9. **for** each region r incident to e
10. **for** each edge e' of r that is not e
11. create interval $I_{v,e,e'}$
12. **for** each newly created interval $I_{v,e,e'}$
13. split $I_{v,e,e'}$ into monotonic intervals $I^1_{v,e,e'}$ and $I^2_{v,e,e'}$
14. **for** each monotonic interval $I^j_{v,e,e'}$, $1 \leq j \leq 2$
15. insert into QLIST the first non-interval path for $I^j_{v,e,e'}$ if necessary
16. insert into QLIST the first interval path for $I^j_{v,e,e'}$

Function 1: HandleNewDiscovery(v)

Whether or not v is a newly discovered point, Function Propagate(v,p) creates candidate optimal paths by propagating p in a constant number of ways and inserts these paths into QLIST.

1. **if** p is an interval path associated with $I_{v',e',e}$ **then**
2. **if** p is still valid **then**
3. **if** v is the last point in $I_{v',e',e'}$ **then**
4. insert into QLIST the second non-interval path for $I_{v',e',e'}$ if necessary
5. **else**
6. insert into QLIST the next interval path for $I_{v',e',e}$
7. **else** (p is an edge-crawling path whose last segment is on edge e)
8. **if** v is not an endpoint of e **then**
9. $v_{prev} \leftarrow$ the previous point of v on path p
10. $v_{next} \leftarrow$ the neighboring point of v that is not between v_{prev} and v
11. **if** there has not already been a non-interval edge-crawling path that extends to v_{next} from v **then**
12. insert path $p_{new} = p + \overline{vv_{next}}$ into QLIST

Function 2: Propagate(v,p)

We have explained previously how paths are propagated inside intervals. Observe that the task of handling interval paths is accomplished by the combination of the procedures Propagate and HandleNewDiscovery. For example, intervals are created in the procedure HandleNewDiscovery when a point v is discovered. At the same time, the first interval path associated with each new interval is inserted into QLIST (line 16 of the procedure HandleNewDiscovery). The propagation of interval paths for each interval is accomplished in the procedure Propagate (line 6).

Each interval also generates two non-interval paths, one when the interval is created (line 15 of the procedure HandleNewDiscovery) and the other when the last interval path of that interval is extracted from QLIST (line 4 of the procedure Propagate). Another situation that generates non-interval paths is when the newly discovered point v is a vertex. In this case, for each incident boundary edge e of v a non-interval path is inserted into QLIST; this path extends $p'_{opt}(s,v)$ to the neighboring Steiner point of v along e, as indicated by line 8 of HandleNewDiscovery.

All the non-interval paths are edge-crawling paths. According to the procedure Propagate (line 8 to line 12), when an non-interval path p from s to v is extracted from QLIST, we may insert an extension of this path into QLIST. Suppose $\overline{v_{prev}v}$ is the last segment of path p. Since p is edge-crawling, v_{prev} is on the same edge e as v. Let v_{next} be the adjacent Steiner point of v on e that is on the other side of v_{prev}. We insert path $p + \overline{vv_{next}}$ into QLIST, if there has not been another path $p' + \overline{vv_{next}}$ inserted into QLIST.

It is easy to see that, for each Steiner point $v \in V_s$, there will be at most two non-interval paths from s to v inserted into QLIST, one approaching v from left and one approaching v from right. Similarly, for any vertex $v \in V$, there will be at most $D(v)$ non-interval paths that connect s and v, one from each edge incident to v. Here $D(v)$ is the number of incident boundary edges of v in the original triangular decomposition. Thus, the total number of non-interval paths is bounded by $\sum_{v \in V} D(v) + \sum_{e \in E} 2m = 2|E| + 2m|E| = O(mn)$. The total number of all paths inserted into QLIST, therefore, is bounded by $O(mn)$.

5 Complexity and Correctness

5.1 Complexity

The complexity of the BUSHWHACK algorithm depends on three factors:

Cost of maintaining QLIST: There are $O(mn)$ candidate optimal paths inserted into QLIST. Each insertion or deletion operation takes $O(\log(mn))$ time. The total cost of maintaining QLIST is $O(nm \log nm)$.

Cost of maintaining PLIST$_e$: There are $O(m)$ points on each edge. Each insertion or deletion operation takes $O(\log m)$ time. As there are $O(n)$ boundary edges, the total cost of maintaining PLIST$_e$ for all e is $O(nm \log m)$.

Cost of maintaining ILIST$_{e,e'}$: When a point v on e is discovered, we first need to decide the range of $I_{v,e,e'}$ for each e' that share a region with e. Deciding each of the two endpoints of $I_{v,e,e'}$ can be done by a binary search on intervals in ILIST$_{e,e'}$. As the maximum number of intervals in ILIST$_{e,e'}$ is m, a binary search takes $O(\log m)$ time. Also, the time to insert an interval to the interval list ILIST$_{e,e'}$ is $O(\log m)$. As there are $O(n)$ interval lists, the total cost of maintaining these interval lists is $O(nm \log m)$.

The complexity of the algorithm, therefore, is $O(nm \log nm)$.

In Section 2 we claimed that, in average, for each Steiner point v BUSHWHACK needs to evaluate the costs of only $O(\log m)$ adjacent edges of v. Even though inside each region only $O(m)$ edges are ever used by candidate optimal paths inserted into QLIST, that is, $O(1)$ edges per Steiner point in the region, BUSHWHACK has to evaluate the costs of additional edges in order to maintain the intervals. As mentioned above, deciding the bounding points of a new interval $I_{v,e,e'}$ takes a binary search of $O(\log m)$ steps. At each step, BUSHWHACK needs to compare the cost of $\overline{vv^*}$ for some $v^* \in e'$ with the cost of $\overline{v'v^*}$, where v' is one of the two neighboring discovered points of v on e. As a result, $O(\log m)$ edges are evaluated for each Steiner point.

5.2 Correctness

To show that the BUSHWHACK algorithm is correct, it suffices to prove the following theorem:

Theorem 1 *When path $p(s,v)$ is extracted from QLIST, if v is not yet discovered, $p(s,v)$ is an optimal discrete path.*

Before we give the proof of Theorem 1, we first establish the following lemma:

Lemma 3 *Each p_i^* as defined in Definition 2 is locally optimal.*

Proof In the following we assume that u lies between v_k^* and v_{k+1}^* for some $k, 1 \le k < d$. The other two cases can be proved similarly. We need to show that for each i, p_i^* as defined in Definition 2 is locally optimal.

We first discuss the case in which $i \le k$ and therefore, according to Definition 2.a, $p_i^* = p_i$. For any j, if $j \ge i$, $\|p_{j,i}\| = \|p_j\| + d_e(v_j^*, v_i^*) \ge \|p_j\| \ge \|p_i\| = \|p_i^*\|$. If $j < i$, we have the following inequality:

$$\begin{aligned}
\|p_i^*\| &= \|p_i\| \\
&= \|p_j\| + w_r \cdot (\sqrt{|\overline{v_h v_i^*}|^2 + h^2} - \sqrt{|\overline{v_h v_j^*}|^2 + h^2}) \\
&= \|p_j\| + \int_{|\overline{v_h v_j^*}|}^{|\overline{v_h v_i^*}|} w_r d(\sqrt{x^2 + h^2}) \\
&= \|p_j\| + \int_{|\overline{v_h v_j^*}|}^{|\overline{v_h v_i^*}|} \frac{w_r \cdot x}{\sqrt{x^2 + h^2}} dx \\
&< \|p_j\| + \int_{|\overline{v_h v_j^*}|}^{|\overline{v_h v_i^*}|} w_{e'} dx \\
&= \|p_j\| + w_{e'} \cdot |\overline{v_j^* v_i^*}| \\
&= \|p_{j,i}\|
\end{aligned}$$

Now we consider the case in which $i = k+1$ and therefore $\|p_i^*\| = \min\{\|p_{k,k+1}\|, \|p_{k+1}\|\}$. For any $j > i$, $\|p_{j,i}\| \ge \|p_i\| = \|p_{i,i}\| \ge \|p_i^*\|$. For any $j < i$, $\|p_{j,i}\| = \|p_{j,k}\| + w_{e'} \cdot |\overline{v_k^* v_{k+1}^*}| \ge \|p_k^*\| + w_{e'} \cdot |\overline{v_k^* v_{k+1}^*}| = \|p_{k,k+1}\| \ge p_i^*$.

Finally, we suppose $k+1 < i \le d$. For any $j \le k+1$, $\|p_{j,i}\| = \|p_{j,k+1}\| + w_{e'} \cdot |\overline{v_{k+1}^* v_i^*}| \ge \|p_{k+1}^*\| + w_{e'} \cdot |\overline{v_{k+1}^* v_i^*}| = \|p_i^*\|$. For any $j \ge i$, $\|p_{j,i}\| = \|p_j\| + d_e(v_j^*, v_i^*) \ge \|p_j\| \ge \|p_i\| \ge \|p_i^*\|$. For any j, $k+1 < j \le i$, we have the following inequality:

$$\begin{aligned}
\|p_i^*\| &= \|p_{k+1}^*\| + w_{e'} \cdot |\overline{v_{k+1}^* v_j^*}| + w_{e'} \cdot |\overline{v_j^* v_i^*}| \\
&= \|p_{k+1}^*\| + \int_{|\overline{v_h v_{k+1}^*}|}^{|\overline{v_h v_j^*}|} w_{e'} dx + w_{e'} \cdot |\overline{v_j^* v_i^*}| \\
&< \|p_{k+1}^*\| + \int_{|\overline{v_h v_{k+1}^*}|}^{|\overline{v_h v_j^*}|} \frac{w_r \cdot x}{\sqrt{x^2 + h^2}} dx + w_{e'} \cdot |\overline{v_j^* v_i^*}| \\
&\le \|p_{k+1}\| + w_r \cdot \int_{|\overline{v_h v_{k+1}^*}|}^{|\overline{v_h v_j^*}|} d(\sqrt{x^2 + h^2}) + w_{e'} \cdot |\overline{v_j^* v_i^*}| \\
&= \|p_{k+1}\| + w_r \cdot (\sqrt{|\overline{v_h v_j^*}|^2 + h^2} - \sqrt{|\overline{v_h v_{k+1}^*}|^2 + h^2}) + w_{e'} \cdot |\overline{v_j^* v_i^*}| \\
&= \|p_j\| + w_{e'} \cdot |\overline{v_j^* v_i^*}| \\
&= \|p_{j,i}\|
\end{aligned}$$

This finishes the proof. ∎

Now we are ready to prove the main theorem.

Proof (of Theorem 1) For the sake of contradiction we assume that such a path p is not optimal. Let p^* be an optimal discrete path from s to v such that $\|p^*\| < \|p\|$, as shown in Figure 10. Let u_{next} be the first undiscovered point on p^* and let u be the predecessor of u_{next} on p^*. As p^* is an optimal discrete path from s to v, $p^*[s, u_{next}]$ must be an optimal discrete path from s to u_{next}, and $p^*[s, u]$ must be an optimal discrete path from s to u.

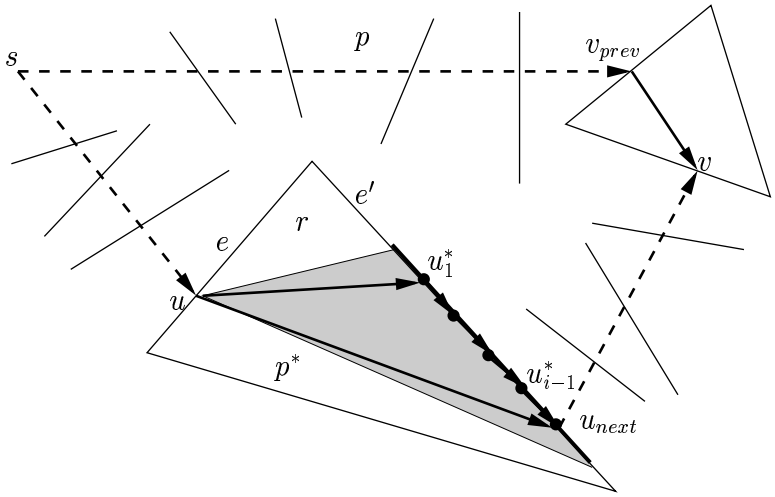


Figure 10: Correctness proof

There are two cases. We first consider the case when segment $\overline{uu_{next}}$ is face-crossing. Let r be the region in which $\overline{uu_{next}}$ lies. Let e and e' be the boundary edges of r such that $u \in e$ and $u_{next} \in e'$. Since u is a discovered point, $I_{u,e,e'}$ has already been created and inserted into $\text{ILIST}_{e,e'}$. According to Definition 1, u_{next} must be inside interval $I_{u,e,e'}$. Otherwise there should be anther discovered point u' on e such that $d'_{opt}(s,u') + d_r(u', u_{next}) < d'_{opt}(s,u) + d_r(u, u_{next}) = \|p^*[s, u_{next}]\|$. This is a contradiction to the fact that $p^*[s, u_{next}]$ is an optimal discrete path from s to u_{next}. As u_{next} is not discovered yet, u_{next} must have not been processed. Therefore, according to Lemma 2 there is a current Steiner point $v^*_{u,e,e'}$ for $I_{u,e,e'}$, and $v^*_{u,e,e'}$ is ahead of u_{next} in interval $I_{u,e.e'}$. Let $u_1^*, u_2^*, \cdots, u_{j-1}^*, u_j^* = v^*_{u,e,e'}, u_{j+1}^*, \cdots, u_{i-1}^*, u_i^* = u_{next}, u_{i+1}^*, \cdots, u_d$ be the d Steiner points in interval $I_{u,e,e'}$. For each k, $1 \le k \le d$, let p_k^* be the locally optimal path for u_k^*. In particular, p_j^* is the current interval path of $I_{u,e,e'}$ as it connects s and u_j^*, the current Steiner point. As $\|p_j^*\| \le \|p_i^*\| = \|p^*[s, u_{next}]\| \le \|p^*\| < \|p\|$, p_j^* must be extracted from QLIST before p is extracted from QLIST. For the same reason, any $p_k^*, j \le k \le i$, must be extracted from QLIST before p. This is a contradiction to the assumption that u_{next} is undiscovered at the time p is extracted from QLIST.

Now we consider the case when $\overline{uu_{next}}$ is edge-crawling. Let e be the edge that contains both u and u_{next}. We first assume that $u \in V$. Let $u_1 = u, u_2, \cdots, u_{i-1}, u_i = u_{next}$ be the Steiner points from u to u_{next} on e. According to line 8 of procedure HandleNewDiscovery, path $p'_{opt}(s,u) + \overline{uu_2}$ is inserted into QLIST when u is discovered. As $\|p'_{opt}(s,u) + \overline{uu_2}\| = \|p'_{opt}(s,u)\| + d_e(u,u_2) < \|p'_{opt}(s,u)\| + d_e(u,u_i) = \|p^*[s,u_{next}]\| \le \|p^*\| < \|p\|$, path $p'_{opt}(s,u) + \overline{uu_2}$ should be extracted from QLIST before p is. Similarly, we can show that for each k, $1 < k \le i$, path $p'_{opt}(s,u) + \overline{uu_k}$ is extracted from QLIST before p is. This is a contradiction to the assumption that u_{next} is not discovered when p is extracted from QLIST.

Next we assume that $u \in V_s$. Let u_{prev} be the predecessor of u on path p^*. Then u_{prev} must not be on e and thus segment $\overline{u_{prev}u}$ is face-crossing. Let e_{prev} be an edge that contains u_{prev}. As u_{prev} is a discovered point, interval $I_{u_{prev},e_{prev},e}$ must have been created and inserted into $\text{ILIST}_{e_{prev},e}$. u must be inside $I_{u_{prev},e_{prev},e}$ as otherwise $p^*[s,u]$ is not an optimal discrete path.

If u_{next} is also inside $I_{u_{prev},e_{prev},e}$, similar to the case where $\overline{uu_{next}}$ is face-crossing, we can prove that a locally optimal path associated with $I_{u_{prev},e_{prev},e}$ that connects s and u_{next} must have been extracted from $I_{u_{prev},e_{prev},e}$ before p is. Again a contradiction.

If u_{next} is not in the interval, let $u_1^*, u_2^*, \cdots, u_{i-1}^*, u_i^* = u, u_{i+1}^*, \cdots, u_d^*$ be the points inside $I_{u_{prev},e_{prev},e}$. As $I_{u_{prev},e_{prev},e}$ is monotonic, without loss of generality, we assume that $d_r(u_{prev}, u_1^*) \leq d_r(u_{prev}, u_2^*) \leq \cdots \leq d_r(u_{prev}, u_d^*)$. Let u_0^* be the Steiner point adjacent to u_0^* outside $I_{u_{prev},e_{prev},e}$ and u_{d+1}^* the Steiner point adjacent to u_d^* outside $I_{u_{prev},e_{prev},e}$. If u_d is between u and u_{next}, as $\|p_d^*\| \leq \|p^*[s,u] + \overline{uu_d^*}\| = \|p^*[s,u]\| + d_e(u, u_d^*) < \|p^*[s,u]\| + d_e(u, u_{next}) = \|p^*[s,u_{next}]\| \leq \|p^*\| < \|p\|$, all locally optimal paths associated with $I_{u_{prev},e_{prev},e}$ must be extracted from QLIST before p is extracted. Therefore, according to line 4 of procedure Propagate, non-interval path $p_{d+1}^* = p_d^* + \overline{u_d^* u_{d+1}^*}$ must be inserted into QLIST before p is extracted from QLIST. This non-interval path, also less costly than p, is eventually going to be extended along e to u_{next} before p is extracted. A contradiction to the fact that u_{next} is undiscovered. The case when u_1^* is between u and u_{next} can be handled similarly.

By the above analysis, we show that in any case u_{next} cannot be an undiscovered point when p is extracted from QLIST. Therefore, such a p^* does not exist and hence p is an optimal discrete path from s to v. ∎

The above correctness proof may help to explain why we need to propagate the paths the way described by the pseudo code.

6 Discretization with Reduced Size

In this section we describe an improvement on the discretization scheme proposed by Aleksandrov *et al.* [2]; this improvement removes the dependency of the discretization size (and hence the time complexity of the approximation algorithm) on the unit weight ratio μ.

We first introduce some notations. For any point v, we let $E(v)$ be the set of edges incident to v and let D_v be the minimum distance between v and edges in $E \backslash E(v)$. For each edge $e \in E$, we let $D_e = \sup\{D_v \mid v \in e\}$ and let v_e be the point on e so that $D_{v_e} = D_e$. For each vertex v of the triangular subdivision, the *radius* $r'(v)$ of v is defined to be $\frac{D_v}{5}$, and the *weighted radius* $r(v)$ of v is defined to be $\frac{w_{min}(v)}{w_{max}(v)} \cdot r'(v)$, where $w_{min}(v)$ and $w_{max}(v)$ are the minimum and maximum unit weights among all regions incident to v, respectively.

According to the discretization scheme of Aleksandrov *et al.* [2], for each boundary edge $e = \overline{v_1 v_2}$, the Steiner points on e are chosen as the following. Each vertex v_i has a "vertex-vicinity" $S(v_i)$ of radius $r_\epsilon(v_i) = \epsilon r(v_i)$ and the Steiner points $v_{i,1}, v_{i,2}, \cdots, v_{i,k_i}$ are placed on the segment of e outside the vertex-vicinities so that $|\overline{v_i v_{i,1}}| = r_\epsilon(v_i)$, $|\overline{v_{i,j} v_{i,j+1}}| = \epsilon D_{v_{i,j}}$ and $\overline{v_{i,k_i} v_i} + \epsilon D_{v_{i,k_i}} \geq |\overline{v_i v_e}|$. The number of Steiner points placed on e can be bounded by $C(e) \cdot \frac{1}{\epsilon} \log \frac{1}{\epsilon}$, where

$$C(e) = O\left(\frac{|e|}{D_e} \log \frac{|e|}{\sqrt{r(v_1) r(v_2)}}\right) = O\left(\frac{|e|}{D_e}\left(\log \frac{|e|}{\sqrt{r'(v_1) r'(v_2)}} + \log \mu\right)\right).$$

This discretization can guarantee a 3ϵ-good approximate optimal path.

Observe that, for this discretization scheme, on each boundary edge e Steiner points are placed more densely in the portion of e closer to the two endpoints, with the exception that no Steiner point is placed inside the vertex-vicinities. Therefore, the larger the vertex vicinities are, the less Steiner points the discretization needs to use. In the following we show that the radius $r_\epsilon(v)$ of the vertex-vicinity of v can be increased to $\epsilon r'(v)$ while still guaranteeing the same error bound. Here we assume that $\epsilon \leq \frac{1}{2}$.

A piecewise linear path p is said to be a *normalized path* if it does not cross region boundaries inside vertex vicinities other than at the vertices. That is, for each bending point u of p, if u is located on boundary edge $e = \overline{v_1 v_2}$, then either u is one of the endpoints of e, or $|\overline{v_i u}| \geq r_\epsilon(v_i)$ for $i = 1, 2$. For example, the path shown in Figure 11 is not a normalized path, as it passes through u_1 and u_2, both of which are inside the vertex vicinity of v. We first prove the following lemma:

Lemma 4 *For any path p from s to t, there is a normalized path $\hat{p}$ from s to t so that $\|\hat{p}\| = (1 + \frac{\epsilon}{2}) \cdot \|p\|$.*

Proof Suppose path p passes through the vertex vicinity $S(v)$ of v, as shown in Figure 11. We use u_1 (u_2, respectively) to denote the first (last, respectively) bending point of p inside $S(v)$, and use u_1'' (u_2'') to denote the first (last, respectively) bending point of p on the boundary of the union of all regions incident

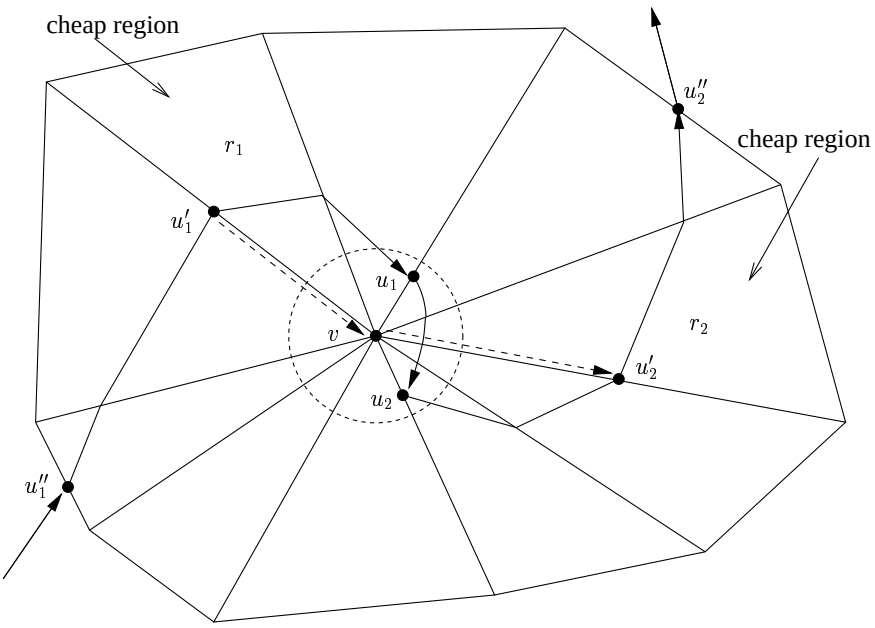


Figure 11: Path passing through vicinity of a vertex

to v. By the definition of D_v, we have $|p[u_1'', u_1]| + |\overline{u_1 v}| \geq D_v$ and $|p[u_2, u_2'']| + |\overline{v u_2}| \geq D_v$. Therefore, $|\overline{u_1 v}|/|p[u_1'', u_1]| \leq \frac{\epsilon \cdot D_v/5}{D_v - \epsilon \cdot D_v/5} = \frac{\epsilon}{5-\epsilon} \leq \frac{\epsilon}{4}$, as $|\overline{u_1 v}| \leq \frac{\epsilon D_v}{5}$. Similarly, we can prove that $|\overline{v u_2}|/|p[u_2, u_2'']| \leq \frac{\epsilon}{4}$.

We let r_1 be the region with the minimum unit weight among all regions crossed by subpath $p[u_1'', u_1]$, and u_1' be the point where $p[u_1'', u_1]$ enters region r_1 for the first time. Similarly, we let r_2 be the region with the minimum unit weight among all regions crossed by subpath $p[u_2, u_2'']$, and let u_2' be the point where $p[u_2, u_2'']$ leaves region r_1 for the last time.

Consider replacing subpath $p[u_1'', u_2'']$ by normalized subpath $\hat{p}[u_1'', u_2''] = p[u_1'', u_1'] + \overline{u_1' v} + \overline{v u_2'} + p[u_2', u_2'']$. We have the following inequality:

$$
\begin{aligned}
& \|\hat{p}[u_1'', u_2'']\| - \|p[u_1'', u_2'']\| \\
= \;& w_{r_1} \cdot |\overline{u_1' v}| + w_{r_2} \cdot |\overline{v u_2'}| - \|p[u_1', u_1]\| - \|p[u_1, u_2]\| - \|p[u_2, u_2']\| \\
\leq \;& (w_{r_1} \cdot |\overline{u_1' v}| - \|p[u_1', u_1]\|) + (w_{r_2} \cdot |\overline{v u_2'}| - \|p[u_2, u_2']\|) \\
\leq \;& w_{r_1}(|\overline{u_1' v}| - |p[u_1', u_1]|) + w_{r_2}(|\overline{v u_2'}| - |p[u_2, u_2']|) \\
\leq \;& w_{r_1} \cdot |\overline{u_1 v}| + w_{r_2} \cdot |\overline{v u_2}| \\
\leq \;& w_{r_1} \cdot \frac{\epsilon \cdot |p[u_1'', u_1]|}{4} + w_{r_2} \cdot \frac{\epsilon \cdot |p[u_2, u_2'']|}{4} \\
\leq \;& \frac{\epsilon}{4} \cdot (\|p[u_1'', u_1]\| + \|p[u_2, u_2'']\|) \\
\leq \;& \frac{\epsilon}{4} \cdot \|p[u_1'', u_2'']\|
\end{aligned}
$$

Therefore, $\|\hat{p}[u_1'', u_2'']\| \leq (1+\frac{\epsilon}{4})\|p[u_1'', u_2'']\|$. Suppose p passes through k vertex vicinities, $S(v_1)$, $S(v_2)$, $\cdots$, $S(v_k)$. For each v_i, we replace the subpath p_i of p that passes through $S(v_i)$ by a normalized subpath $\hat{p}_i$ as we described above. Let $\hat{p}$ be the resulting normalized path. Note that the sum of the weighted lengths of $p_1, p_2, \cdots, p_k$ is less than twice of the weighted length of p, we have $\|\hat{p}\| \leq \|p\| + \frac{\epsilon}{4} \sum_{i=1}^{k} \|p_i\| \leq (1 + \frac{\epsilon}{2})\|p\|$. ∎

With Lemma 4, it is easy to prove the claimed error bound for this modified discretization:

Theorem 2 *The discretization constructed with $r_\epsilon(v) = \epsilon r'(v)$ contains a 3ϵ-good approximation for an optimal path p_{opt} from s to t, for any vertices s and t.*

Proof We first construct a normalized path $\hat{p}$ such that $\|\hat{p}\| \leq (1 + \frac{\epsilon}{2})\|p_{opt}\|$.

We call a segment of a boundary edge bounded by two adjacent discrete points (Steiner points or vertices) a *Steiner segment*. In particular, if both of the two end points of a Steiner segment are Steiner points, we say that it is a *true Steiner segment*. Otherwise, if one of the end point is a vertex, we say that the Steiner segment is a *half Steiner segment*.

For each segment $\overline{v_1v_2}$ of $\hat{p}$, let $\overline{u_{1,1}u_{1,2}}$ ($\overline{u_{2,1}u_{2,2}}$, respectively) be the Steiner segment that contains v_1 (v_2, respectively). Since $\hat{p}$ is normalized, there are only three cases regarding these two Steiner segments:
i) both $\overline{u_{1,1}u_{1,2}}$ and $\overline{u_{2,1}u_{2,2}}$ are pure Steiner segments;
ii) one of $\overline{u_{1,1}u_{1,2}}$ and $\overline{u_{2,1}u_{2,2}}$ is a pure Steiner segment, and the other is a half Steiner segment;
iii) both $\overline{u_{1,1}u_{1,2}}$ and $\overline{u_{2,1}u_{2,2}}$ are half Steiner segments; however, they do not share the same vertex.

For each of the three cases, it is easy to show that $|\overline{u_{1,i}u_{2,j}}| \leq (1+2\epsilon)\cdot|\overline{v_1v_2}|$ for any $i = 1, 2$ and $j = 1, 2$. So we can construct a discrete path p' such that $\|p'\| \leq (1+2\epsilon)\|\hat{p}\|$. Therefore, $\|p'\| \leq (1+2\epsilon)(1+\frac{\epsilon}{2})\|p_{opt}\| = (1 + \frac{5}{2}\epsilon + \epsilon^2)\|p_{opt}\| \leq (1 + 3\epsilon)\|p_{opt}\|$, assuming $\epsilon \leq \frac{1}{2}$. ∎

With the modification on the radius for each vertex vicinity, for each edge e the number of Steiner points placed on e is reduced to $C'(e) \cdot \frac{1}{\epsilon} \cdot \log\frac{1}{\epsilon}$, where $C'(e) = O(\frac{|e|}{D_e}\log\frac{|e|}{\sqrt{r'(v_1)r'(v_2)}})$. $C'(e)$ is independent of the unit weight ratio, and therefore so is the time complexity of any approximation algorithm that uses this discretization. It appears that each of the previous ϵ-approximation algorithms has a time complexity dependent on the unit weight ratio.

7 Experimental Results

In order to provide a performance comparison, we implemented using Java the following three algorithms: 1) the BUSHWHACK algorithm, the Algorithm 1 we presented in Section 4; 2) *pure Dijkstra's algorithm*, which searches every incident edge of a Steiner point in $\mathcal{G}_\epsilon$; 3) *cone-based Dijkstra's algorithm* (by Aleksandrov *et al.* [2]), which searches an incident edge of a Steiner point only if the edge is inside the geodesic cone of the point. All the timed results were acquired from a Sun Blade-1000 workstation with 4GB memory.

We also have implemented the *pruned Dijkstra's algorithm* (also by Aleksandrov *et al.* [2]), which only uses an ϵ-spanner of each geodesic cone. Asymptotically, this algorithm is better than the cone-based Dijkstra's algorithm. However, it occurs to us that, for the ϵ values we have chosen for our experiments, the pruned Dijkstra's algorithm is actually slower than the cone-based Dijkstra's algorithm. First of all, using the discretization $\mathcal{G}_\epsilon$ that guarantees to contain an ϵ-good approximate optimal path, the pruned Dijkstra's algorithm can only guarantee to find an 5ϵ-good approximate optimal path, due to the extra error introduced by using the spanner. Therefore, to guarantee to find an ϵ-good approximate optimal path, the pruned Dijkstra's algorithm has to use the discretization $\mathcal{G}_{\epsilon/5}$, which contains much more Steiner points than $\mathcal{G}_\epsilon$. Secondly, the pruned Dijkstra's algorithm needs to partition each geodesic cone into roughly $\frac{1}{\epsilon}$ equal-sized angular ranges, so that one Steiner point can be picked inside each range. In our current implementation, this step will require $\frac{1}{\epsilon}$ binary searches, which are very time-consuming. Before we come up with a more efficient implementation of the pruned Dijkstra's algorithm, we will not be able to correctly measure its performance.

One of the concerns in conducting these experiments is the choice of triangulations. We want to avoid randomly generated triangulations for two reasons. First, randomly generated data may affect the performances of various algorithms in an unexpected way. The second and main reason is that randomly generated triangulations may contain many "skewed" triangular faces (e.g., faces that are very skinny). Recall that the size of a discretization generated by either the discretization method of [2] or the one we presented in Section 6 is dependent on θ_{min}. If a certain experiment uses ten problem instances and for one problem instance θ_{min} is very small, and much smaller than those of other problem instances, the experiment will take too much time to finish, and the result of this problem instance will dominate the result of the entire experiment.

For our experiments we chose triangulations converted from terrain maps in grid data format. More specifically, we used the DEM (Digital Elevation Model) file of Kaweah River basin. It is a 1424x1163 grid

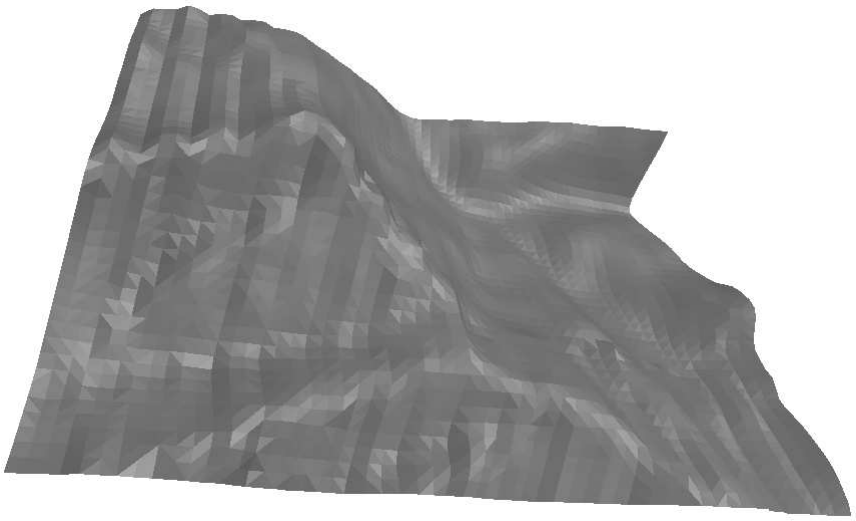


Figure 12: A 60x45 TIN

with 30m between two neighboring grid points. We randomly took twenty 60x45 patches and converted them to TINs by connecting two grid points diagonally for each grid cell. Figure 12 shows one of the TINs we used.

Each triangular face of the resulting TINs will not be too skewed; its projection to the horizontal plane is an isosceles right triangle. We assign to each face r a unit weight w_r that is equal to $1 + 10 \sin \alpha_r$, where α_r is the angle between r and the horizontal plane.

For each TIN, we ran the four algorithms five times, each time choosing randomly generated source and destination points. For each algorithm, we took the average of the running times of all experiments. We repeated the experiments with $\frac{1}{\epsilon} = 3, 5, 7$ and 9. We list the results in Table 3.

Table 3: Statistics of running time (in seconds) and number of visited edges per region

Algorithm	BUSHWHACK	pure Dijkstra	cone-based Dijkstra
$\frac{1}{\epsilon} = 3$	156.9 / 2371	243.0 / 16558	272.8 / 13680
$\frac{1}{\epsilon} = 5$	290.7 / 4603	711.0 / 55797	655.1 / 38623
$\frac{1}{\epsilon} = 7$	440.6 / 7098	1506.0 / 124086	1166.9 / 74045
$\frac{1}{\epsilon} = 9$	631.9 / 9795	2672.5 / 224987	1824.7 / 119832

From Table 3, it is easy to see that, when $\frac{1}{\epsilon}$ grows, the running time of the BUSHWHACK algorithm is growing much slower than those of both the pure Dijkstra's algorithm and cone-based Dijkstra's algorithm. This observation is consistent with the asymptotic analysis, as the time complexity of the BUSHWHACK algorithm is less dependent on $\frac{1}{\epsilon}$. We also list the average number of visited edges per region for each algorithm and each ϵ value. It occurs to us that, the number of visited edges per region is closely related to the running time.

From the experimental results we learned that, although for the discretization we used the number of Steiner points inserted on each boundary edge is $O(\frac{1}{\epsilon} \log \frac{1}{\epsilon})$, the hidden constant factor, which is determined by some geometric parameters, is very large. For example, for $\epsilon = \frac{1}{9}$, the number of Steiner points on each edge is in the order of a few hundreds. The reason is that, for discretization based on edge-subdivision, the error of an approximate optimal path is bounded "locally." The approximate optimal path and the corresponding optimal path are divided into many subpaths, and the weighted length of each subpath of the approximate optimal path has to be no more than $(1 + \epsilon)$ times of that of the corresponding subpath of the

optimal path. All the subpaths of the approximate optimal path usually cannot contribute an ϵ-relative error simultaneously, (and some of the subpaths may even be less costly than their counterparts in the optimal path,) the overall relative error of the approximate optimal path is far less than ϵ.

That being said, we believe that this discretization scheme based on edge subdivision is still very useful in practice. It is very easy to implement, and can accommodate different cost metrics. For example, Lanthier *et al.* [8] showed that with minor modifications a discretization can be constructed for the shortest anisotropic path problem, while Reif and Sun [14] presented a similar discretization for the optimal path planning problem in regions with flows.

8 Conclusion

In this paper we present a new approximation algorithm to solve the weighted shortest path problem. Compared to some of the previous work, our algorithm provides a more effective way of finding optimal paths in the discretized space (resulted from either uniform or non-uniform discretization). We also provide an improved discretization method that removes the dependency of the size of the discretized space (and hence the complexity of any approximation algorithm that uses the discretization method) on the unit weight ratio.

Acknowledgments

This work was supported in part by Grants NSF/DARPA CCR-9725021, CCR-96-33567, NSF IRI- 9619647, ARO contract DAAH-04-96-1-0448, and ONR contract N00014-99-1-0406.

References

[1] L. Aleksandrov, M. Lanthier, A. Maheshwari, and J.-R. Sack. An ϵ-approximation algorithm for weighted shortest paths on polyhedral surfaces. In *Proceedings of the 6th Scandinavian Workshop on Algorithm Theory*, volume 1432 of *Lecture Notes in Computer Science*, pages 11–22, 1998.

[2] L. Aleksandrov, A. Maheshwari, and J.-R. Sack. Approximation algorithms for geometric shortest path problems. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pages 286–295, 2000.

[3] R. Alexander. Construction of optimal-path maps for homogeneous-cost-region path-planning problems. Ph.D. Thesis, Computer Science, U.S. Naval Postgraduate School, Monterey, CA, 1989.

[4] R. Alexander and N. Rowe. Geometrical principles for path planning by optimal-path-map construction for linear and polygonal homogeneous-region terrain. Technical report, Computer Science, U.S. Naval Postgraduate School, Monterey, CA, 1989.

[5] R. Alexander and N. Rowe. Path planning by optimal-path-map construction for homogeneous-cost two-dimensional regions. In *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, 1990.

[6] M. Kindl, M. Shing, and N. Rowe. A stochastic approach to the weighted-region problem, II: Performance enhancement techniques and experimental results. Technical report, Computer Science, U.S. Naval Postgraduate School, Monterey, CA, 1991.

[7] M. Lanthier, A. Maheshwari, and J. Sack. Approximating weighted shortest paths on polyhedral surfaces. In *Proceedings of the 13th Annual ACM Symposium on Computational Geometry*, pages 274–283, 1997.

[8] M. Lanthier, A. Maheshwari, and J.-R. Sack. Shortest anisotropic paths on terrains. In *Proceedings of the 26th International Colloquium on Automata, Languages and Programming*, volume 1644 of *Lecture Notes in Computer Science*, pages 524–533, 1999.

[9] C. Mata and J. Mitchell. A new algorithm for computing shortest paths in weighted planar subdivisions. In *Proceedings of the 13th Annual ACM Symposium on Computational Geometry*, pages 264–273, 1997.

[10] J. S. B. Mitchell. Geometric shortest paths and network optimization. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 633–701. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.

[11] 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.

[12] N. Papadakis and A. Perakis. Minimal time vessel routing in a time-dependent environment. *Transportation Science*, 23(4):266–276, 1989.

[13] N. Papadakis and A. Perakis. Deterministic minimal time vessel routing. *Operations Research*, 38(3):426–438, 1990.

[14] J. H. Reif and Z. Sun. Movement planning in the presence of flows. In *Proceedings of the 7th International Workshop on Algorithms and Data Structures*, volume 2125 of *Lecture Notes in Computer Science*, pages 450–461, 2001.

[15] N. C. Rowe. Roads, rivers, and obstacles: optimal two-dimensional route planning around linear features for a mobile agent. *International Journal of Robotics Research*, 9(6):67–74, Dec. 1990.

[16] N. C. Rowe and R. F. Richbourg. An efficient Snell's Law method for optimal-path planning across multiple two dimensional, irregular, homogeneous-cost regions. *International Journal of Robotics Research*, 9(6):48–66, Dec. 1990.

[17] T. R. Smith, G. Peng, and P. Gahinet. Asynchronous, iterative, and parallel procedures for solving the weighted-region least cost path problem. *Geographical Analysis*, 21(2):147–166, 1989.

[18] Z. Sun and J. H. Reif. BUSHWHACK: An approximation algorithm for minimal paths through pseudo-Euclidean spaces. In *Proceedings of the 12th Annual International Symposium on Algorithms and Computation*, volume 2223 of *Lecture Notes in Computer Science*, pages 160–171, 2001.

[19] Z. Sun and J. H. Reif. On energy-minimizing paths on terrains for a mobile robot. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, May 2003. To appear.