Weighted Region Shortest Path Problem

John Reif* Zheng Sun†
Duke University

July 12, 1999

Abstract

In this paper we present an approximation algorithm for solving the following optimal motion planning problem: Given a planar space composed of polygonal regions, each of which is associated with a positive unit weight, and two points $s$ and $t$, find a path from $s$ to $t$ with the minimum weight, where the weight of a path is defined to be the weighted sum of the lengths of the sub-paths within each region.

Some of the previous works on this problem involve constructing a discretization of the planar space and then computing the approximate paths in the discretized space. Discretizations used by these works include uniform and non-uniform discretizations. Our algorithm adopts discretizations proposed by previous work and presents improvement in computing optimal paths in the discretized space by using a “discretized wavefront” method.

Compared to the previous work, our algorithm has the following three advantages: (a) can compute exact solutions for a discrete case of this problem; (b) can be applied to any discretization; and (c) can be applied to a more generic class of problems.

1 Introduction

Robotics motion planning problems are some of the most fundamental problems in robotics research. An important category of this problem is to determine the optimal path (shortest path according to a user-defined metric on paths) between two points, $s$ and $t$, in a 2-D or 3-D space, under various conditions. This problem has drawn great attention from researchers in robotics as well as other areas such as computational geometry, geographical information systems (GIS), and graph theory, due to its practical application. There have been numerous papers on the optimal path problem. Interested readers may refer to survey [10] and [11] for a comprehensive review of this problem.

In this paper we examine the weighted version of this problem in 2-D space. More specifically, a weight function $F : \mathcal{R}^2 \rightarrow \mathcal{R}$ is defined on the space. For any path $p$ from $v_1$ to $v_2$, the weighted length of $p$ is defined to be $\int_{v_1}^{v_2} F(x)dx$, the integral of the weight function along the path. The objective of the minimum weighted path problem is to find the optimal path, or the path with the minimum weighted length, from a given source point $s$ and any destination point $t$.

Observe that the unweighted optimal path problem can be considered as a special case for the weighted optimal path problem: for any point in the “free space”, the weight is defined to be 1; for any point in the “obstacle” (if there is any), the weight is defined to be $+\infty$.

*Department of Computer Science, Duke University, Box 90129, Durham, NC 27708-0129. E-mail: reif@cs.duke.edu. This work was supported in part by Grants NSF/DARPA CCR-9725021, CCR-96-33567, NSF IRI-9619647, ARO contract DAAD-04-96-1-0448, and ONR contract N00014-99-1-0406.

†E-mail: sunz@cs.duke.edu.
Even though currently there is no complexity result regarding finding the exact solution of the weighted optimal path, it is generally considered to be very hard. (Even the unweighted optimal path problem in 3-D space was proven by Canny and Reif to be NP-hard.) As a result, most of the known research work is focused on approximation algorithms on a special case of this problem where the planar space can be divided into polygons where the weight is uniform within each polygon. In some literature, this problem is named “homogeneous-cost-region-path-planning problem”.

Before we give an overview of previous work, we first define some notations that will be used in the rest of this paper. We use \( n \) to denote the number of vertices of polygonal regions of the plane, and \( L \) the length of the longest edge of all the polygonal regions. For our convenience, we assume that all the coordinates of the vertices are integers, and \( N \) is the maximum coordinate. For all the weights of different regions, let \( W_{\text{max}} \) be the maximum weight and \( W_{\text{min}} \) be the minimum weight.

Finally, for any \( s \) and \( t \), we use \( D(s,t) \) to denote the weighted length of an optimal path from \( s \) to \( t \) and use \( D'(s,t) \) to denote the weighted length of an approximate optimal path found by a given approximate algorithm. We decompose the error of an approximation into two components: the absolute error \( \delta \), which is bounded by a constant disregarding the value of \( D(s,t) \); and the relative error \( \epsilon \), which is linear to \( D(s,t) \). Therefore, we can represent the error of an approximation of the optimal path from \( s \) to \( t \) as \( D'(s,t) - D(s,t) = \epsilon \cdot D(s,t) + \delta \).

One of the early algorithms on this problem was given by Mitchell and Papadimitriou [9]. Their algorithm uses “Shells Law” and “continuous Dijkstra method” to give an optimal-path map for any given source point \( s \). The time complexity of their algorithm is \( O(n^3M) \). Here \( M \) is a function of various input parameters including the (relative) error tolerance \( \epsilon \).

Recently Mata and Mitchell [8] presented another approximation algorithm based on discretization—the “pathnet” algorithm. In their scheme, a “pathnet graph” of size \( O(nk) \) is constructed and then an approximate optimal path with relative error of \( \epsilon \) is computed, where \( \epsilon = O\left(\frac{W_{\text{max}}/W_{\text{min}}}{k}\right) \). The time complexity, in terms of \( \epsilon \), \( n \) and other geometric parameters, is \( O\left(\frac{n^3}{W_{\text{max}}/W_{\text{min}}}\right) \).

Lanthier et al [7] and Aleksandrov et al [1] both adopted a natural approach to solving this problem which discretizes the polygonal subdivision by placing Steiner points along the edges of polygonal regions. Both algorithms construct a dense discrete graph by connecting (in a specified way) the Steiner points that share the same region and then compute an approximate optimal path in the resulting discrete graph using Dijkstra algorithm. 

Lanthier et al [7] used a uniform discretization, which adds \( O(n^2) \) points on each edge evenly, to compute approximate paths with a \( LW_{\text{max}} \)-absolute error and \( \epsilon \)-relative error. The absolute error of \( LW_{\text{max}} \) is resulted from the discretization while the relative error of \( \epsilon \) is resulted from using spanner to compute the approximate optimal path in the discrete graph. The relative error \( \epsilon \) is treated as a constant in their complexity analysis. Including \( \epsilon \) also as an input parameter, the complexity is \( O\left(\frac{n^3}{\epsilon} + n^3 \log n\right) \), as in the discrete graph there will be \( O(n^3) \) vertices and \( O\left(\frac{2n^3}{\epsilon}\right) \) edges.

Aleksandrov et al [1] proposed a logarithmic discretization that can guarantee a relative error of \( \epsilon \). The time complexity of this algorithm is \( O(mn \log mn) \), where \( m = O(\log_2(L/r)) \) is the number of points added on each edge. Here \( r \) is the minimum distance from any point to the boundary of the faces adjacent to it, and \( \delta = 1 + \frac{W_{\text{max}}}{W_{\text{min}}} \sin \theta_{\text{min}} \), where \( \theta_{\text{min}} \) is the minimum angle between any two edges, and \( W_{\text{min}} \) is the minimum weight. If all the points have integer coordinates and \( N \) is the maximum coordinate, we have \( r = \Omega(1/N) \) and \( \theta_{\text{min}} = \Omega(1/N^2) \), and thus the time bound is \( O\left(\frac{n^2W_{\text{max}}}{W_{\text{min}}} \log (LN)\right) \).

In this paper we present a wavefront-like algorithm that can compute optimal paths on any discretization (uniform or logarithmic) more effectively than the ones used in [7] and [1]. Instead of computing an approximate optimal path in the discretized space with \( \epsilon \)-relative error, as the above-mentioned two algorithm do, our algorithm can give an exact optimal path.
in less (for the uniform discretization) or equivalent (for the logarithmic discretization) time.

For the purpose of solving the original (continuous) weighted shortest path problem, our algorithm improves Lanthier et al. [7]'s work in terms of both time complexity (\(O(n^3 \log n)\) vs. \(O(\frac{n^2}{2} + n^3 \log n)\)) and accuracy (same absolute error but no relative error) while matches Aleksandrov et al. [1]'s work (\(O(nm \log(mn))\) time and \(\epsilon\)-relative error).

Although our algorithm will not be able to improve the performance of [1] due to the nature of discretization, we found our algorithm interesting for the following reasons:

i It provides an efficient way of solving a discrete version of the weighted optimal path problem, where all the interesting destination points are located on the boundary of triangular regions and only paths that consist of line segments connecting those points are allowed. (This is the exact problem our algorithm solves.)

ii It can be applied to different discretizations to achieve different approximation goals. For example, if absolute error is to be limited, we can apply this algorithm on a uniform discretization. If restricting relative error is the main goal, the logarithmic discretization then can be used.

iii It can be applied to a more general class of problems. As one might have noticed, our algorithm only assumes the following geometric property:

\[
\text{Inside each region (not including the edges), the weighted length of a path } p_1 \\
\text{is less than that of } p_2 \text{ if and only if the Euclidean length of } p_1 \text{ is less than that of } p_2.
\]

Therefore, our algorithm can be applied to optimal path problems on planar regions where weight of a path is defined differently but the above-mentioned property holds.

For other related work, see [2, 3, 4, 6, 12, 13, 14, 15], or see survey [11].

2 Preliminaries

2.1 Overview of Our Approach

Let \(S\) be a polygonal subdivision of a planar space, and let \(n\) be the number of vertices of \(S\). This implies the number of edges and faces is also \(O(n)\). Without loss of generality, we assume that each face of \(S\) is a triangle. In case that there are non-triangular faces in the input subdivision, we can triangulate the subdivision. The resulted subdivision \(S'\) will still have \(O(n)\) vertices, edges and faces. As defined above in section 1, we use \(L\) to denote the length of the longest edge among the \(O(n)\) edges.

Each face \(f\) is associated with a non-negative real number \(w_f\), the weight of \(f\). For simplicity, we assume that \(w_f \in [1, W_{max}]\). For each edge \(e\), the weight \(w_e\) is defined to be \(\min\{w_f, w_f'\}\), where \(f\) and \(f'\) are the two faces adjacent to \(e\).

In general, for any line segment \(\overline{v_1v_2}\) in a triangular region \(f\), we use \(|\overline{v_1v_2}|\) to denote its Euclidean length and \(W(\overline{v_1v_2})\) to denote its weighted length which is defined to be \(|\overline{v_1v_2}| w_f\).

If \(v_1\) and \(v_2\) are on the same edge \(e\) of \(f\), \(W(\overline{v_1v_2})\) is \(|\overline{v_1v_2}| \cdot w_e\). Here \(w_e\) might be different from \(w_f\) in case the other face \(f'\) adjacent to \(e\) has a smaller weight than that of \(f\). For two paths \(p_1\) and \(p_2\), we use \(p_1 + p_2\) to denote the concatenation of \(p_1\) and \(p_2\) and \([p,v_1,v_2]\) to denote the part of path \(p\) between \(v_1\) and \(v_2\).

Let \(s\) and \(t\) be the source and the destination, respectively. We can assume that \(s\) and \(t\) are two vertices of \(S\). (If otherwise, we can add a constant number of edges to construct from \(S\) a new subdivision \(S'\) which has \(s\) and \(t\) as vertices.) The problem is to find a shortest weighted length path (which will be called optimal path) from \(s\) to \(t\).

It has been proven (see [9]) that an optimal path consists of \(O(n^2)\) straight line segments. The endpoints of each segment are on the boundary of a triangular face.
As we mentioned previously, our algorithm can be applied to any discretization that places points on each edge. Therefore, for now, we use \( f(e) \) to denote the number of Steiner points added along a given edge \( e \). Here \( f(e) \) is dependent on \( e \). It may also depend on \( n \) and some other parameters, as we shall see later in this paper. We use \( V \) to denote the set of vertices of \( S \) and use \( V_s \) to denote the set of all Steiner points.

We first focus on a subspace \( P' \) of the original path space \( P \). Each path in \( P' \) is composed of a number of line segments, each of whose endpoints are points in \( V \cup V_s \). Further, the two endpoints of such a line segment are in a same face of \( S \). We call such a path a discrete path.

We first compute an optimal discrete path \( p'_{\text{opt}}(s, t) \) from \( s \) to \( t \) in \( P' \) and use this path to approximate the optimal path \( p_{\text{opt}}(s, t) \) from \( s \) to \( t \) in the original space.\(^1\) Then, we show that \( D'(s, t) \), the weighted length of \( p'_{\text{opt}}(s, t) \), gives a good approximation to \( D(s, t) \).\(^2\)

In the following discussion, a “point” refers to a point in \( V \cup V_s \).

Our algorithm is similar to Dijkstra algorithm. However, by utilizing some geometric properties of this problem, we achieve a time complexity lower than that of Dijkstra algorithm.

### 2.2 Data Structures

Before giving the description of our algorithm, we first present some basic data structures.

Our algorithm keeps a list \( Q\text{LIST} \) of paths. Each path \( p' \in Q\text{LIST} \) is a candidate optimal path from \( s \) to some point \( v \in V \cup V_s \), and is represented by a quadruplet \((v, v_{\text{prev}}, e_{\text{prev}}, l)\). Here \( v_{\text{prev}} \) is the previous point of \( v \) on \( p' \), \( e_{\text{prev}} \) is an edge that contains \( v_{\text{prev}} \), and \( l \) is the length of \( p' \). Each \( p' \) is “almost optimal” in the sense that it is the concatenation of \( p'_{\text{opt}}(s, v_{\text{prev}}) \), the optimal path from \( s \) to \( v_{\text{prev}} \), and line segment \( v_{\text{prev}}v \). Thus, \( l = D'(s, v_{\text{prev}}) + W(v_{\text{prev}}v) \).

\( Q\text{LIST} \) is sorted in ascending order according to the weighted length of candidate optimal paths (i.e., value of \( l \) of the quadruplets). Possible operations on this list are insertion and removal. We can use an AVL tree (see [5]) to implement \( Q\text{LIST} \) so that each operation costs only \( O(\log |Q\text{LIST}|) \) time.

Also, for each edge \( e = uv \) of \( S \), we keep a list \( \text{PLIST}_e \) of discovered points on \( e \). A point \( v \) is said to be discovered when the optimal discrete path from \( s \) to \( v \) is determined. The points in this list are ordered according to the distance to \( v_l \). Possible operations on \( \text{PLIST}_e \) include insertion and finding the closest neighbors. Again, using an AVL tree can guarantee that each operation only costs \( O(\log |\text{PLIST}_e|) = O(\log f(e)) \) time.

Let \( f \) be one of the two faces adjacent to \( e \), and let \( e_1 \) and \( e_2 \) be the other two edges of \( f \). For each point \( v \) in \( \text{PLIST}_e \), we use \( I_{v,e,f} \) to denote the set of points on \( e_1 \cup e_2 \) with the following property: for any point \( v' \) on \( e_1 \cup e_2 \), \( v' \in I_{v,e,f} \) if and only if \( W(\overline{vv'}) + D'(s,v') < W(\overline{vv'}) + D'(s,v) \) for any other point \( v' \) in \( \text{PLIST}_e \). We will show later that each such \( I_{v,e,f} \) is an interval of contiguous points on \( e_1 \cup e_2 \), as indicated in Figure 1.a.

If one \( I_{v,e,f} \) contains points on both \( e_1 \) and \( e_2 \), we split it into two intervals \( I'_{v,e,f} = I_{v,e,f} \cap e_1 \) and \( I''_{v,e,f} = I_{v,e,f} \cap e_2 \). Also, for any \( I_{v,e,f} \subset e_1 (e_2) \), if the point \( v \) closest to \( v \) in \( I_{v,e,f} \) is not an endpoint of \( I_{v,e,f} \), we split \( I_{v,e,f} \) into two intervals \( I'_{v,e,f} \) and \( I''_{v,e,f} \) in such a way that for each interval, the closest point to \( v \) is one of the two endpoints (and the farthest point to \( v \) is the other endpoint). This is to guarantee that the distance to \( v \) forms a monotonically increasing or decreasing series for points in \( I_{v,e,f} \) from left to right along edge \( e \).

From the construction of \( I_{v,e,f} \), it is easy to see that at most four intervals are associated with each point \( v \in V \cup V_s \) due to splitting. For the sake of simplicity, in the following

\(^1\)As a matter of fact, in a single run of our algorithm, an optimal discrete path \( p'_{\text{opt}}(s, v) \) is computed for each \( v \in V \cup V_s \).

\(^2\)The goodness of the approximation will depend on the discretization used.
discussion we may assume that only one interval is associated with each point. Removing this assumption will only increase the time complexity of our algorithm by a constant factor.

![Intervals and Propagating Candidate Optimal Paths](image)

Figure 1: Intervals and Propagating Candidate Optimal Paths

We let \( v'_{e,f} \) and \( v''_{e,f} \) denote the left and right endpoint of \( I_{e,f} \), respectively. As we shall see later, every point \( v' \) in \( I_{e,f} \) will be “processed”, meaning that a candidate optimal path from \( s \) to \( v' \) which enters face \( f \) through \( v \) will be added (and later removed) from QLIST. We let \( \hat{v}_{e,f} \) be the closest point of \( I_{e,f} \) to \( v \) that has not yet been processed. Initially it is either one of the two endpoints of \( I_{e,f} \). An interval \( I_{e,f} \) is said to be active if at least one point \( v' \) in \( I_{e,f} \) has not been discovered.

Let \( \text{ILIST}_{e,f} \) be the list that includes each interval \( I_{e,f} \) for \( v \in \text{PLIST}_e \). The intervals in \( \text{ILIST}_{e,f} \) are maintained in the order according to the order of \( v \) in \( \text{PLIST}_e \). We also use an AVL tree to implement each \( \text{ILIST}_{e,f} \) to keep the cost for each operation on \( \text{ILIST}_{e,f} \) to be \( O(\log |\text{ILIST}_{e,f}|) = O(\log f(e)) \).

3 The Algorithm

We now give the formal specification of our algorithm.

**Function FindOptimalDiscrete**

1. (Initialization) QLIST = \{ (s, s, 0, 0) \}. For each \( e \), \( \text{PLIST}_e = \emptyset \). For each edge-face pair \( (e, f) \), \( \text{ILIST}_{e,f} = \emptyset \). For each point \( v \in V \cup V_s \), \( D'(s, v) = +\infty \).

2. (Main Loop) While QLIST is not empty:

   [a] Remove the first entry of QLIST. Let it be \( (v, v_{\text{prev}}, e_{\text{prev}}, l) \).

   [b] If \( D'(s, v) = +\infty \), i.e., no other candidate optimal path \( p' \) (with weighted length at most \( l \)) has been removed from QLIST:

   [i] Set \( D'(s, v) = l \) and set \( p'_{\text{opt}}(s, v) = p_{\text{opt}}'(s, v_{\text{prev}}) + v_{\text{prev}}v \).

   [ii] Insert \( v \) into \( \text{PLIST}_e \) for each \( e \) that contains \( v \).

   [iii] Let \( e_1, e_2, \cdots, e_t \) be the edges that contain \( v \) and let \( f_{1,j} \) and \( f_{2,j} \) be the two faces adjacent to \( e_j \), \( j = 1, 2, \cdots, t \). Call InsertInterval \( (v, e_j, f_{i,j}) \) for all \( 1 \leq j \leq t, 1 \leq i \leq 2 \).

\(^3\)In case \( v \) is a vertex of \( S \), \( v \) might be adjacent to multiple edges.
[c] Call Propagate\(v, v'_{\text{prev}}, e_{\text{prev}}, l\).

The function InsertInterval is used to update an interval list \(I_{\text{LIST}, e, f}\) when a new point \(v\) on \(e\) is discovered. The primary functionality of this function is to determine \(v'_{e, v, e, f}\) and \(v'_e\), the left and right endpoints of \(I_{e, v, e, f}\).

**Function InsertInterval**\((v, e, f)\)

1. Find the left and right neighbors, \(v_l\) and \(v_r\), of \(v\) in \(\text{PLIST}_e\).
2. Initially, \(v'_{e, v, e, f} = \text{NULL}\), \(v'_{e, v} = \text{NULL}\).
3. Do a binary search on the part of \(c_1 \cup c_2\) to the right of (including) \(v'_{e, v, e, f}\) (the right endpoint of the interval associated with the left neighbor of \(v\)) until we find two adjacent points \(\hat{v}_1\) and \(\hat{v}_2\) such that, if \(\hat{v}_1\) is in the interval associated with \(v'_{e, v, e, f}\), \(\hat{v}_2\) is in the interval associated with \(v'_{e, v}^2\), \(D'(s, v) + W(v\hat{v}_1) < D'(s, v'_{e, v}) + W(v\hat{v}_1)\) but \(D'(s, v) + W(v\hat{v}_2) > D'(s, v'_{e, v}) + W(v\hat{v}_2)\). In another word, to construct a path from \(s\) to \(\hat{v}_1\), it is advantageous to extend \(p'_{\text{opt}}(s, v)\) by adding one more line segment \(v\hat{v}_1\) than to extend \(p'_{\text{opt}}(s, v'_{e, v})\) by adding line segment \(v'_{e, v}^2\); however, to construct a path from \(s\) to \(\hat{v}_2\), extending \(p'_{\text{opt}}(s, v)\) is a better idea.
4. Let \(v'_{e, v, e, f} = \hat{v}_1\).
5. Similarly, compute \(v'_{e, v}\).
6. Insert \(I_{e, v, e, f}\) into \(I_{\text{LIST}, e, f}\). For each interval \(I_{e, v, e, f}\) entirely covered by \(I_{e, v, e, f}\), set \(I_{e, v, e, f} = \emptyset\).
7. Add \((\hat{v}_1, \hat{v}_2, e, D'(s, v) + W(\hat{v}_1, \hat{v}_2))\) to \(\text{QLIST}\).
8. For any interval \(I_{e, v, e, f}\), if \(e\) is partially covered by \(I_{e, v, e, f}\), adjust its boundary by removing all points that are covered by \(I_{e, v, e, f}\). Add \((\hat{v}_1, \hat{v}_2, v^*, e, D'(s, v^*) + W(v^*, v, e, f))\) into \(\text{QLIST}\).

Function Propagate is used to add candidate optimal paths to \(\text{QLIST}\) after \((v, v'_{\text{prev}}, e_{\text{prev}}, l)\) is removed from the list.

**Function Propagate**\((v, v'_{\text{prev}}, e_{\text{prev}}, l)\)

1. If \(v\) is not on \(e_{\text{prev}}\): suppose \(f\) is the face that contains both \(e_{\text{prev}}\) and \(v\), and \(e\) is the edge of \(f\) that contains \(v\):
   
   [1.a] If there has been another quadruplet \((v', v'_{\text{prev}}', e_{\text{prev}}', f')\) removed from \(\text{QLIST}\) previously (i.e., another candidate optimal path in the form of \(p'_{\text{opt}}(s, v'_{\text{prev}}') + v'_{\text{prev}}'\) has been found with a shorter weighted length, where \(v'_{\text{prev}}\) is also on \(e_{\text{prev}}\), return.
   
   [1.b] Let \(v_{\text{new}}\) be the left or right neighbor of \(v\) on \(e\) such that \(|v'_{\text{prev}} v_{\text{new}}| > |v_{\text{prev}} v|\). If \(W(v_{\text{prev}} v_{\text{new}}) \geq W(v_{\text{prev}} v)\), insert \((v_{\text{new}}, v_{\text{prev}}, e, D'(s, v) + W(v_{\text{prev}} v_{\text{new}}))\) into \(\text{QLIST}\); otherwise insert \((v_{\text{new}}, v_{\text{prev}}, e_{\text{prev}}', D'(s, v_{\text{prev}}') + W(v'_{\text{prev}}' v_{\text{new}}))\).
   
   These two cases correspond to the two ways of propagate candidate optimal paths as shown in Figure 1.b.

2. If \(v\) is on \(e_{\text{prev}}\):
   
   [2.a] If there has been another quadruplet \((v, v'_{\text{prev}}', e_{\text{prev}}', f')\) removed from \(\text{QLIST}\), return.
   
   [2.b] Otherwise, let \(v_{\text{new}}\) be the neighbor of \(v\) on \(e_{\text{prev}}\) that is farther from \(v_{\text{prev}}\) than \(v\) is. Insert \((v_{\text{new}}, v, e_{\text{prev}}, D'(s, v) + W(v_{\text{prev}} v_{\text{new}}))\) into \(\text{QLIST}\).

### 4 Correctness and Error Bound

In the first subsection we present the correctness proof for our algorithm to find an optimal discrete path \(p'_{\text{opt}}(s, v)\) for every \(v \in V \cup V_s\). Then we show how an optimal discrete path can provide an approximate solution to the original problem.\(^4\)

\(^4\)It is possible as the weight of \(e\) may be lower than the weight of \(f\).
4.1 Correctness Proof

We first give a lemma that is key to our algorithm.

**Lemma 1** For any active $I_{v,e,f}$, let $\hat{v} \in I_{v,e,f}$ be the closest point (in terms of Euclidean distance) to $v$ that has not been discovered. Assume $I_{v,e,f} \subset e_1$. Then, there must be a point $\hat{v}' \in I_{v,e,f}$ with $W(\hat{v}') \leq W(\hat{v})$ that satisfies at least one of the following two conditions:

(i) Quadruplet $(\hat{v}', v, e, D'(s, v) + W(\hat{v})) \in$ QLIST; (ii) Quadruplet $(\hat{v}'', e_1, D'(s, \hat{v}'') + W(\hat{v}'')) \in$ QLIST and $D'(s, \hat{v}'') + W(\hat{v}'') < D'(s, v) + W(\hat{v})$. Here $\hat{v}'$ is either the left neighbor or right neighbor of $\hat{v}$ on $e_1$. We call a point that satisfies condition (i) (condition (ii)) a “handle point” of Type A (B, respectively). Observe that there might be more than one handle point for an interval.

**Proof** After an interval $I_{v,e,f}$ is first created (and thus inserted into ILIST$_{e,f}$) when the optimal path from $s$ to $v$ is decided, the function Propagate is called which will add $(\hat{v}_{v,e,f}, v, e, D'(s, v) + W(\hat{v}_{v,e,f}))$ into the list QLIST. Here $\hat{v}_{v,e,f}$ is the closest point to $v$ in $I_{v,e,f}$ and thus is a handle point.

There are two events that could possibly make $I_{v,e,f}$ lose handle points: (a) a quadruplet is removed from QLIST; (b) The boundary of $I_{v,e,f}$ is adjusted (as in step [7] of function InsertInterval) so that the handle points are now in another interval $I'_{v',e,f}$. In the following we prove that in any of the above two cases, at least one handle point will be generated unless $I_{v,e,f}$ becomes inactive.

(a) Suppose $I_{v,e,f}$ loses a Type A handle point $\hat{v}'$ after a quadruplet $(\hat{v}', v, e, D'(s, v) + W(\hat{v}))$ is removed from QLIST. The fact that $\hat{v}' \in I_{v,e,f}$ excludes the possibility that another quadruplet $(\hat{v}', v', e', \hat{v}')$ removed from QLIST previously. Therefore, step [b] of function Propagate is executed and quadruplet $(\hat{v}'_{\text{new}}, v, e, D'(s, v) + W(\hat{v}'_{\text{new}}))$ is inserted into QLIST if $W(\hat{v}'_{\text{new}}) \leq W(\hat{v}) + W(\hat{v}'_{\text{new}})$; otherwise, quadruplet $(\hat{v}'_{\text{new}}, \hat{v}'_{1}, D'(s, \hat{v}') + W(\hat{v}'_{\text{new}}))$ is inserted. Here $\hat{v}'_{\text{new}}$ is the (left or right) neighbor of $\hat{v}'$ on $e_1$ such that $W(\hat{v}'_{\text{new}}) > W(\hat{v})$.

If $\hat{v}'_{\text{new}}$ is not in $I_{v,e,f}$, then $\hat{v}'$ must be an endpoint of $I_{v,e,f}$. Furthermore, it must be the point of $I_{v,e,f}$ that is farthest from $v$. Therefore, all the points in $I_{v,e,f}$ must have already been discovered. Thus, $I_{v,e,f}$ is not active anymore. If $\hat{v}'_{\text{new}}$ is in $I_{v,e,f}$, we claim that it must be a handle point of $I_{v,e,f}$. Observe that, for any undiscovered point $\hat{v} \in I_{v,e,f}$, $\hat{v} \neq \hat{v}'$, $W(\hat{v}) > W(\hat{v}')$ and thus $W(\hat{v}) \geq W(\hat{v}')$. Thus, in case quadruplet $(\hat{v}'_{\text{new}}, v, e, D'(s, v) + W(\hat{v}'_{\text{new}}))$ is inserted into QLIST, $\hat{v}'_{\text{new}}$ is a Type A handle point. In case quadruplet $(\hat{v}'_{\text{new}}, \hat{v}', e_1, D'(s, \hat{v}') + W(\hat{v}'_{\text{new}}))$ is inserted into QLIST, we have

\[
\begin{align*}
D'(s, \hat{v}') + W(\hat{v}'_{\text{new}}) &\leq D'(s, v) + W(\hat{v}') + W(\hat{v}'_{\text{new}}) \\
&\leq D'(s, v) + W(\hat{v}'_{\text{new}}) \\
&\leq D'(s, v) + W(\hat{v}).
\end{align*}
\]

Hence, $\hat{v}'_{\text{new}}$ is a Type B handle point.

Similarly, we can prove that if $I_{v,e,f}$ loses a Type B handle point $\hat{v}'$, another point $\hat{v}'_{\text{new}} \in I_{v,e,f}$ will become a new Type B handle point, unless $I_{v,e,f}$ is not active anymore.

(b) Suppose interval $I_{v,e,f}$ loses all of its handle points as a result of a boundary adjustment. However, as specified in step [7] of InsertInterval, a new quadruplet $(\hat{v}_{v,e,f}, v, e, D'(s, v) + W(\hat{v}_{v,e,f}))$ is inserted into the list QLIST. Here $\hat{v}_{v,e,f}$ is the closest point to $v$ in the (adjusted) $I_{v,e,f}$ and thus is a handle point.

Therefore, at any time, as long as $I_{v,e,f}$ is active, there is at least one handle point in $I_{v,e,f}$. □

With the above lemmas, we are able to prove the following theorem:
Theorem 1  Function $FindOptimalDiscrete$ determines an optimal discrete path $p'_{opt}(s, v)$ for every point $v \in V \cup V_s$.

Proof It suffices to prove that, at each time step $[b]$ of function $FindOptimalDiscrete$ is executed, the optimal path $p'_{opt}(s, v) = p'_{opt}(s, v_{prev}) + \nu_{prev}$ is determined.

Suppose for the sake of contradiction that $p^*$ is an optimal discrete path from $s$ to $v$ with $W(p^*) \leq W(p'_{opt}(s, v))$. Let $v^*$ be the discovered point on $p^*$ that is closest to $v$. $v^*$ can not be $v$ as $v$ is not discovered before quadruplet $(v, v_{prev}, e, f)$ is removed from QLIST. Let $v^*_{next}$ be the successor of $v^*$ on $p^*$.

First assume $v^*$ and $v^*_{next}$ are not on the same edge of $S$. Let $f^*$ be the face that contains both $v^*$ and $v^*_{next}$ and let $e^*$ be the edge of $f^*$ that contains $v^*$. Since $p^*$ is an optimal path from $s$ to $v$, $p'[s, v^*_{next}]$ is an optimal path from $s$ to $v^*_{next}$. Thus, $v^*_{next}$ must be in $I_{v^*, e^*, f^*}$.

From Lemma 1 we know that there is at least one handle point in $I_{v^*, e^*, f^*}$. Let $v'$ be a handle point of Type A. (The proof for the case when $v'$ is a Type B handle point is similar.) Let $v^*_1, v^*_2, \ldots, v^*_k = v^*_{next}$ be the points of $I_{v^*, e^*, f^*}$ between $v'$ and $v^*_{next}$. Since $D'(s, v^*) + W(v^*_i v^*_{next}) \leq D'(s, v^*) + W(v^*_{next} v^*_{next}) < W(p^*) < D'(s, v_{prev}) + W(v_{prev} v^*_{next})$, for $i = 1, 2, \ldots, k$, (equality holds only when $v' = v^*_k$) quadruplet $(v^*_i, v^*, e^*, D'(s, v^*) + W(v^*_i v^*_{next}))$ should in turn be inserted into (and then removed from) QLIST before quadruplet $(v, v_{prev}, e, D'(s, v_{prev}) + W(v_{prev} v^*_{next}))$ is removed, for $i = 1, 2, \ldots, k$. Therefore, $v^*_1, v^*_2, \ldots, v^*_k$ are already discovered before $v$ is discovered. A contradiction to the assumption that $v^*_{next}$ is not discovered before $v$ is discovered.

The case when $v^*$ and $v$ are on the same edge of $S$ can be handled similarly. Therefore, there can not be such an optimal path $p^*$ with $W(p^*) < p'_{opt}(s, v)$, and hence $p'_{opt}(s, v) = p'_{opt}(s, v_{prev}) + \nu_{prev}$ is an optimal path. □

4.2 Approximating the Optimal Path

As for each $v \in V \cup V_s$ our algorithm uses an optimal path $p'_{opt}(s, v)$ in the discretized path space $P^*$ to approximate the optimal path $p_{opt}(s, v)$ in the original path space $P$, the goodness of our approximation is totally determined by the discretization on which our algorithm is based upon.

If the uniform discretization as in [7] which puts $O(n^2)$ Steiner points on each edge, the absolute error of our approximation is bounded by a constant of $LW_{max}$. However, since we calculate the exact optimal path in the discretized space instead of using spanner to approximate it (as in [7]) our algorithm eliminates the extra relative error of $\epsilon$ introduced by the usage of spanner.
If we adopt the logarithmic discretization proposed by [1] so that \( m = O(\log_\delta (L/r)) \) Steiner points are placed on each edge, the approximation will have a relative error of \( \epsilon \). Although, similar to [7], [1] also uses an approximate algorithm in finding optimal paths in the discretized space, it does not make a difference as a relative error of the same magnitude has already been introduced in the first phase of the algorithm (discretization). Thus, our algorithm will not provide a better performance over [1].

5 Complexity Analysis

In the following complexity analysis, we assume \( f(e) = m \) for all \( e \) in the polygonal subdivision \( S \) for simplicity. (Thus, the number of Steiner points added to each edge is the same.)

To analyze the complexity of our algorithm, we need to examine the number of quadruplets inserted into QLIST. Observe that, when a quadruplet is removed from QLIST, there may be two occasions when new quadruplets are inserted into QLIST. One is occurred within function Propagate and the other is occurred within function InsertInterval.

Quadruplets added through InsertInterval: For each \( v \in V \cup V_s \), when the optimal discrete path \( p_{opt}(s, v) \) is determined, function InsertInterval\( (v, e, f) \) is executed once for each edge-face pair \( (e, v) \) such that \( v \in e \subset f \). In InsertInterval\( (v, e, f) \), a new interval is added into ILIST_{e,f} and a quadruplet is inserted into QLIST, as indicated in line [7] of function InsertInterval. Also, a new quadruplet is inserted into QLIST for each interval partially covered by the new interval \( I_{e, v, f} \). As there are at most two intervals partially covered by the new interval, at most three quadruplets are inserted into QLIST for each \( I_{e, v, f} \). Thus the total number of quadruplets added for point \( v \) is \( 6E(v) \), with \( E(v) \) defined as the number of edges that contains \( v \). For \( v \in V_s \), \( E(v) = 2 \). For \( v \in V \), \( E(v) \) varies but we have \( \sum_{v \in V} E(v) = O(n) \). Therefore, the total number of quadruplets added in executions of function InsertInterval is \( \sum_{v \in V} 6E(v) + \sum_{v \in V_s} 6E(v) = O(|V_s|) + O(n) = O(nm) \).

Quadruplets added through Propagate: Now we calculate the number of new quadruplets inserted (corresponding to candidate optimal path) through executions of function Propagate\( (v, v_{prev}, e_{prev}, l) \) for each \( v \in V \cup V_s \). First we assume that \( v \) is a Steiner point, i.e., \( v \in V_s \). Let \( e \) be the edge that contains \( v \) and let \( f_1, f_2 \) be the two faces adjacent to \( e \). Assume \( e_{i,1}, e_{i,2} \) are the other two edges of face \( f_i \) for \( i = 1, 2 \). Observe that \( e_{prev} \) is either \( e \), or one of \( e_{i,1} \). There are at most six quadruplets in the form of \( (v, *, *, *) \) whose removal from QLIST can lead to an insertion of a new quadruplet into QLIST. The six possible quadruplets are \( (v, v_{left}, e, D'(s, v_{left}) + W(v_{left}v)) \), \( (v, v_{right}, e, D'(s, v_{right}) + W(v_{right}v)) \), and \( (v, v_{prev}, e_{i,j}, D'(s, v_{prev}) + W(v_{prev}v)) \) for \( i = 1, 2, j = 1, 2 \). Here \( v_{left} \) and \( v_{right} \) are the left and right neighbors of \( v \) on \( e \), respectively. This is guaranteed by line [1.a] and [2.a] of function Propagate. Similarly, for \( v \in V \), the number of quadruplets in the form of \( (v, *, *, *) \) whose removal can lead to an insertion of a new quadruplet is bounded by \( O(E(v)) \). Therefore, the total number of quadruplets added through Propagate is \( \sum_{v \in V} O(1) + \sum_{v \in V} O(E(v)) = O(|V_s|) + O(n) \), which is again \( O(nm) \).

From the above discussion, we can conclude that the total number of quadruplets inserted into QLIST is \( O(nm) \). The time cost for maintaining QLIST is thus \( O(nm \log(nm)) \). Also, as there are \( O(nm) \) different intervals \( I_{e, v, f} \), and it takes \( O(\log m) \) time to create such an interval, (observe that a binary search is needed in step [3] of InsertInterval,) the total time
cost for creating these intervals is $O(nm \log m)$. Hence, our algorithm takes $O(nm \log(nm))$ time in total.

For the uniform discretization used in [7], we have $m = n^2$ and thus the time complexity is $O(n^3 \log n)$. For the logarithmic discretization used in [1], the time complexity is $O(mn \log(mn))$ which matches [1]'s result, here $m = O(\log_\delta(L/r))$.

6 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 provide a more effective way of finding optimal paths in the discretized space (resulted from either uniform or non-uniform discretization). Our algorithm has the following three advantages over previous algorithms: (a) can compute exact solutions for a discrete case of this problem; (b) can be applied to any discretization; and (c) can be applied to a more generic class of problems.

We expect to find more applications for our algorithm.

References


