On Finding Energy-Minimizing Paths on Terrains

Zheng Sun, *Member, IEEE,* and John H. Reif, *Fellow, IEEE*

***Abstract*—In this paper, we discuss the problem of computing optimal paths on terrains for a mobile robot, where the cost of a path is defined to be the energy expended due to both friction and gravity. The physical model used by this problem allows for ranges of impermissible traversal directions caused by overturn danger or power limitations. The model is interesting and challenging, as it incorporates constraints found in realistic situations, and these constraints affect the computation of optimal paths. We give some upper- and lower-bound results on the combinatorial size of optimal paths on terrains under this model. With some additional assumptions, we present an efficient approximation algorithm that computes for two given points a path whose cost is within a user-defined relative error ratio. Compared with previous results using the same approach, this algorithm improves the time complexity by using 1) a discretization with reduced size, and 2) an improved discrete algorithm for finding optimal paths in the discretization. We present some experimental results to demonstrate the efficiency of our algorithm. We also provide a similar discretization for a more difficult variant of the problem due to less restricted assumptions.**

***Index Terms*—Computational geometry, mobile-robot motion planning, optimization, road vehicles.**

I. Introduction

A. Motivation and Related Work

With the growth of geographical information systems (GIS), now it is possible to find a terrain map (such as the one for Kaweah River basin shown in Fig. 1) for virtually any location in the world. The availability of these high-resolution maps makes computing energy-minimizing paths for mobile robots possible. However, despite the obvious potential applications in both commercial and military areas, there has not been enough interest in the energy-minimizing path problem, as compared with its practical significance in the real world. The extensively studied Euclidean shortest-path problems fail to capture some of the characteristics of optimal path planning for a mobile robot, such as the variance of the friction coefficient in different areas, the limitations of the driving force of the robot, and the stability of the robot on a steep plane.

Manuscript received February 23, 2004. This paper was recommended for publication by Associate Editor N. Amato and Editor S. Hutchinson upon evaluation of the reviewers' comments. This work was supported in part by the National Science Foundation ITR under Grant EIA-0086015 and Grant 0326157; in part by the National Science Foundation QuBIC under Grant EIA-0218376 and Grant EIA-0218359; in part by the Defense Advanced Research Projects Agengy/Air Force Office of Scientific Research under Contract F30602-01-2-0561; and in part by the Research Grant Council under Grant HKBU2107/04E. This paper was presented in part at the IEEE International Conference on Robotics and Automation, Taipei, Taiwan, R.O.C., September 2003.

Z. Sun is with the Department of Computer Science, Hong Kong Baptist University, Kowloon, Hong Kong (e-mail: sunz@comp.hkbu.edu.hk).

J. H. Reif is with the Department of Computer Science, Duke University, Durham, NC 27708 USA (e-mail: reif@cs.duke.edu).

Fig. 1. Terrain map of Kaweah River basin.

In this paper, we study the problem of optimal path planning on terrains for a mobile robot. Our work is based on the model introduced by Rowe and Ross [1]. In their model, the *cost* of any path on the surface of a terrain is defined to be the energy loss due to both friction and gravity. The goal is to find an *optimal path*, a path with the minimum energy loss between given source and destination points s and t. This model adds anisotropism to optimal path planning by taking into consideration impermissible traversal directions resulting from overturn danger or power limitations. This problem is a generalization of the more extensively studied weighted-region optimal-path problem (see [2]–[11] or survey [12]), yet conceivably more difficult.

Although this model addresses some of the characteristics of optimal path planning for mobile robots that are not considered by Euclidean shortest-path problems, it still does not take into consideration nonholonomic constraints found in real contexts. We refer readers to [13] for a review of works on nonholonomic motion planning.

Rowe and Ross [1] studied the characteristics of optimal paths on terrains; they showed that there are only four ways that an optimal path could traverse a terrain face. They also provided rules for transition on the boundary edges for each combination of the four traversal types. Rowe and Kanayama [14] applied the same model to the surface of a vertical-axis ideal cone. In this case, there are 22 ways that an optimal path could traverse a conic surface patch. They also provided an approximation algorithm that uses these characteristics; this approximation algorithm is able to produce approximate optimal paths much smoother than the ones found by the traditional approximation algorithm using grid-based discretization. Rowe [15] later generalized their work to other path-planning problems with anisotropic cost functions, such as path planning for a missile in a three-dimensional space.

***Preprint of paper appearing* in IEEE Transaction on Robotics, Volume: 21, Issue: 1, February 2005, pages 102-114.**

Before we give a review of previous works, we first define some notations. We let V be the set of vertices of all regions, and let E be the set of all boundary edges. We use w_r to denote the unit weight of any region r. For a boundary edge e separating two regions r_1 and r_2, the unit weight w_e of e is defined to be $\min\{w_{r_1}, w_{r_2}\}$. We define *unit weight ratio* μ to be $\frac{w_{max}}{w_{min}}$, where w_{max} (w_{min}, respectively) is the maximum (minimum, respectively) unit weight among all regions. We use $|p|$ to denote the Euclidean length of path p, and use $p_1 + p_2$ to denote the concatenation of two paths p_1 and p_2.

The first ϵ-approximation algorithm on this problem was given by Mitchell and Papadimitriou [2]. Their algorithm uses "Snell's Law" and "continuous Dijkstra method" to give an optimal-path map for any given source point s. The time complexity of their algorithm is $O(n^8 \log \frac{n\mu}{\epsilon})$. In practice, however, the time complexity is expected to be much lower. Later Mata and Mitchell [3] presented another ϵ-approximation algorithm based on constructing a "pathnet graph" of size $O(nk)$, where $\epsilon = O(\frac{\mu}{k})$. The time complexity, in terms of ϵ and n, is $O(\frac{n^3\mu}{\epsilon})$.

Some of the existing algorithms construct from the original continuous space a weighted graph $\mathcal{G}_\epsilon(V', E')$ by placing discretization points, called *Steiner points*, on boundary edges. The node set V' of $\mathcal{G}_\epsilon$ contains all Steiner points as well as vertices of the regions. The edge set E' of $\mathcal{G}_\epsilon$ contains every edge $\overline{v_1v_2}$ such that v_1 and v_2 are on the border of the same region. The weight of edge $\overline{v_1v_2}$ is determined by the weighted length of segment $\overline{v_1v_2}$ in the original weighted space. $\mathcal{G}_\epsilon$ guarantees to contain an ϵ-good approximate optimal path between s and t, and therefore the task of finding an ϵ-good approximate optimal path is reduced to computing a shortest path in $\mathcal{G}_\epsilon$, which we call *optimal discrete path*, using a discrete search algorithm such as Dijkstra's algorithm or BUSHWHACK [5, 7].

In the remainder of this paper, we will mainly discuss techniques for approximation algorithms using this approach. Since an optimal discrete path from s to t in $\mathcal{G}_\epsilon$ is used as an ϵ-approximation for the *real* optimal path, the phrases "optimal discrete path" and "ϵ-good approximate optimal path" are used interchangeably, and are both denoted by $p'_{opt}(s,t)$.

Aleksandrov *et al.* [4, 6] proposed two discretization schemes that place $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\log\mu)$ Steiner points on each boundary edge to construct $\mathcal{G}_\epsilon$ for a given ϵ. Combining the discretization scheme of [6] with a "pruned" Dijkstra's algorithm, they provided an ϵ-approximation algorithm that runs in roughly $O(\frac{n}{\epsilon}(\frac{1}{\sqrt{\epsilon}} + \log n)\log\frac{1}{\epsilon}\log\mu)$ time.

It is important to note, however, that the discretization size (and therefore the time complexity) for these approximation algorithms also depends on various geometric parameters, such as the smallest angle between two adjacent boundary edges, maximum integer coordinate of vertices, etc. These parameters are omitted here since they are irrelevant to our discussion.

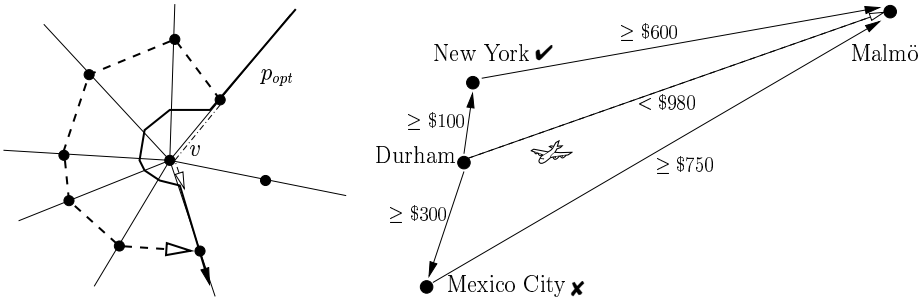
In this paper we present the following results on finding ϵ-good approximate optimal paths in weighted regions:

Compact discretization scheme. The complexity of each of the approximation algorithms we have mentioned above depends more or less on μ, either linearly ([3]) or logarithmically ([2, 4, 6]). This dependency is caused by the cor-

responding discretization scheme used. In particular, the discretization scheme of Aleksandrov *et al.* [6] places $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\log\mu)$ Steiner points on each boundary edge. Here again we omit the other geometric parameters.

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 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 1.a, 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. 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 2 an improvement on the discretization scheme of Aleksandrov *et al.* [6]. 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.



(a) "Bad" optimal path

(b) Searching for the cheapest flight

Fig. 1.

Adaptive discretization method. The traditional approximation algorithms construct from the original space a graph $\mathcal{G}_\epsilon$ and compute with a discrete search algorithm an optimal discrete path in $\mathcal{G}_\epsilon$ in a one-step manner. We call this method the *fixed discretization method*. For single query problem, this method is rather inefficient in that, although the goal is to find an ϵ-good approximate optimal path $p'_{opt}(s,t)$ from s to t, it actually computes an ϵ-good approximate optimal path from s to any point v in $\mathcal{G}_\epsilon$, as long as the cost of such a path is

less than that of $p'_{opt}(s,t)$. Much of the effort is unnecessary as most of these points would not help to find an ϵ-good approximate optimal path from s to t.

We use flight ticket booking as an example. When trying to find the cheapest flight from Durham to Malmö with one stop (supposing no direct flight is available), a travel agent does not need to consider Mexico City as a candidate for the connecting airport if she knows the following: a) there is *always* a route from Durham to Malmö with one stop that costs less than \$980; b) any direct flight from Durham to Mexico City costs no less than \$300; and c) any direct flight from Mexico City to Malmö costs no less than \$750. Therefore, she does not need to find out the exact prices of the direct flights from Durham to Mexico City and from Mexico City to Malmö, saving two queries to the ticketing database.

Analogously, we do not need to compute $p'_{opt}(s,v)$ and $p'_{opt}(v,t)$ for a point $v \in \mathcal{G}_\epsilon$ if we know in advance that v does not connect any optimal discrete path between s and t. However, while the travel agent can rely on knowledge she previously gained, the approximation algorithms using the fixed discretization method have no prior knowledge to draw upon. In Section 3 we discuss a multiple-stage discretization method that we call *adaptive discretization method*. It starts with a coarse discretization $\mathcal{G}' = \mathcal{G}_{\epsilon_1}$ for some $\epsilon_1 > \epsilon$ and adaptively refines $\mathcal{G}'$ until it guarantees to contain an ϵ-good approximate optimal path from s to t. Approximate optimal path information acquired in each stage is used to identify the areas where no optimal path from s to t will pass through and therefore no further Steiner point needs to be inserted in the next stage.

Heuristics for BUSHWHACK The BUSHWHACK algorithm is an alternative algorithm for computing optimal discrete paths in $\mathcal{G}_\epsilon$. It uses a number of complex data structures to keep track of all potential optimal paths. When m, the number of Steiner points placed on each boundary edge, is small, the efficiency gained by accessing only a subgraph of $\mathcal{G}_\epsilon$ is outweighed by the cost of establishing and maintaining these data structures. Another weakness of BUSHWHACK is that its performance improvement diminishes when the number of regions in the space is large. These weaknesses affect the practicability of BUSHWHACK as in most cases the desired quality of approximation does not require too many Steiner points for each boundary edge, while in the given $2\mathcal{D}$ space there can be arbitrary number of regions. In Section 4 we introduce two cost-saving heuristics for the original BUSHWHACK algorithm to overcome the weaknesses mentioned above.

2 Compact Discretization Scheme

In this section we provide an improvement on the discretization scheme of Aleksandrov *et al.* [6] by removing the dependency of the size of $\mathcal{G}_\epsilon$ on the unit weight ratio μ.

For any point v, we let $E(v)$ be the set of boundary edges incident to v and let $d(v)$ be the minimum distance between v and boundary 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 a region, 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.* [6], 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 $C(e) = O(|e|/d(e) \cdot \log(|e|/\sqrt{r(v_1) r(v_2)})) = O(|e|/d(e) \cdot (\log(|e|/\sqrt{r'(v_1) r'(v_2)}) + \log \mu))$. 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 2 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 state the following lemma:

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

Proof. In the following, for a path p and two points $u_1, u_2 \in p$, we use $p[u_1, u_2]$ to denote the subpath of p between u_1 and u_2.

Refer to Figure 2. Suppose path p passes through the vertex vicinity $S(v)$ of v, as shown in Figure 2. 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 border of the union of all regions incident 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 this 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:

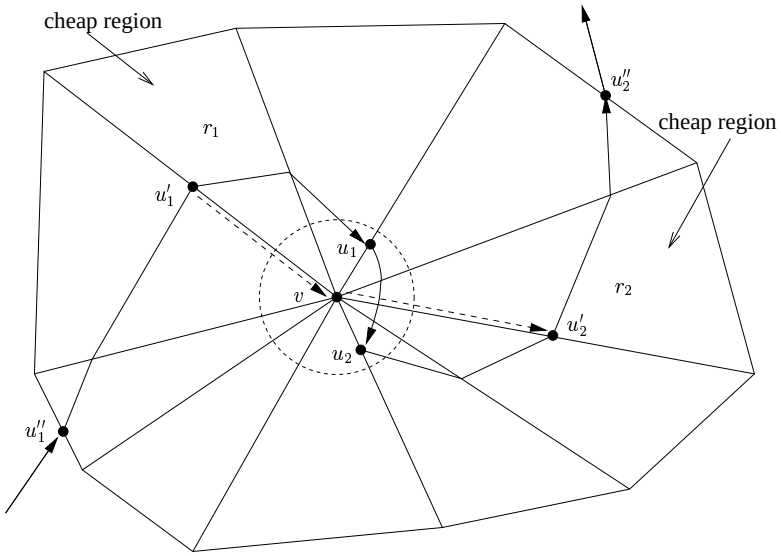


Fig. 2. Path passing through vicinity of a vertex

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

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 $\|\hat{p}\| \leq \|p\| + \frac{\epsilon}{4} \sum_{i=1}^{k} \|p_i\| \leq (1 + \frac{\epsilon}{2})\|p\|$. □

We call a segment of a boundary edge bounded by two adjacent Steiner points a *Steiner segment*. Each segment $\overline{u_1 u_2}$ of a normalized path $\hat{p}$ is significantly long as compared to the Steiner segment on which u_1 or u_2 lies. Therefore, it is easy to find a discrete path in $\mathcal{G}_\epsilon$ that is an ϵ-approximation of $\hat{p}$. With Lemma 1, we can prove the claimed error bound for this modified discretization:

Theorem 1. *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 two vertices s and t.*

Proof. We first construct a normalized path $\hat{p}$ such that $\|\hat{p}\| \leq (1 + \frac{\epsilon}{2})\|p_{opt}\|$. Then we can use a proof similar to the one provided in [6] to show that, for

any normalized path $\hat{p}$, there is a discrete path p' so 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 of each vertex vicinity, for each boundary edge e the number of Steiner points placed on e is reduced to $C'(e)\cdot\frac{1}{\epsilon}\log\frac{1}{\epsilon}$, where $C'(e) = O(|e|/d(e)\log(|e|/\sqrt{r'(v_1)r'(v_2)}))$. Note that $C'(e)$ is independent of μ.

The significance of this compact discretization scheme is that, combining it with either Dijkstra's algorithm or BUSHWHACK, we can get an approximation algorithm whose time complexity does not depend on μ. To our best knowledge, all previous ϵ-approximation algorithms have time complexities dependent on μ.

3 Adaptive Discretization Method

Even with the compact discretization scheme, the size of $\mathcal{G}_\epsilon$ can still be very large even for a modest ϵ, as the number of Steiner points placed on each boundary edge is also determined by a number of geometric parameters. Therefore, computing an ϵ-good approximate optimal path by directly applying a discrete search algorithm to $\mathcal{G}_\epsilon$ may be very costly. In particular, a discrete search algorithm such as Dijkstra's algorithm will compute an optimal discrete path from s to every point $v \in \mathcal{G}_\epsilon$ that is closer to s than t is, meaning that it has to search through a large space with the same (small) error tolerance ϵ.

Here we further elaborate the flight ticket booking example. With the knowledge accumulated through past experiences, the travel agent may know, for any intermediate airport A, a *lower bound* $L_{D,A}$ of the cost of a direct flight from Durham to A as well as a *lower bound* $L_{A,M}$ of the cost of a direct flight from A to Malmö. Further, she also knows an *upper bound*, say, $980, of the cost of the *cheapest* flight (with one stop) from Durham to Malmö. In that case, the travel agent would only consider airport A as a possible stop between Durham and Malmö if $L_{D,A} + L_{A,M} < 980$. For example, it at least worths the effort to check the database to find out the exact cost of the flight from Durham to Malmö via New York, as shown in Figure 1.b.

The A* algorithm partially addresses this issue as it would first explore points that are estimated using a heuristic function to be closer to the destination point t. However, if the unit weights of the regions vary significantly, it is difficult for a heuristic function to provide a close estimation of the weighted distance between any point and t. As a result, the A* algorithm may still have to search through many points in $\mathcal{G}_\epsilon$ unnecessarily.

Here we introduce a multi-stage approximation algorithm that uses an adaptive discretization method. For each i, $1 \leq i \leq d$, this method computes an ϵ_i-good approximate path from s to t in a subgraph $\mathcal{G}'_{\epsilon_i}$ of $\mathcal{G}_{\epsilon_i}$, where $\epsilon_1 > \epsilon_2 > \cdots > \epsilon_{d-1} > \epsilon_d = \epsilon$. In each stage, with the approximate optimal path information acquired through the previous stage, the algorithm can identify for each boundary edge the portion of the edge where more Steiner points need to placed to guarantee an approximate optimal path with a reduced error bound. For the rest portion of the boundary edge, no further Steiner point needs to be placed.

We say that a path p' *neighbors* an optimal path p_{opt} if, for any Steiner segment that p_{opt} crosses, p' passes through one of the two Steiner points that bound the Steiner segment. Our method requires that the discretization scheme satisfy the following property (which is the case for the discretization schemes of [4, 6] and the one described in Section 2):

Property 1. For any two vertices v_1 and v_2 in the original (continuous) space and any optimal path p_{opt} from v_1 and v_2, there is a discrete path from v_1 to v_2 in the discretization with a cost no more than $(1+\epsilon) \cdot \|p_{opt}(v_1, v_2)\|$ that neighbors p_{opt}.

For any two points $v_1, v_2 \in \mathcal{G}'_{\epsilon_i}$, we denote the optimal discrete path found from v_1 to v_2 in the i-th stage by $p'_{\epsilon_i}(v_1, v_2)$. We say that a point $v \in \mathcal{G}'_{\epsilon_i}$ is a *searched point* if an optimal discrete path $p'_{\epsilon_i}(s, v)$ from s to v in $\mathcal{G}'_{\epsilon_i}$ is determined. For each searched point v, we also compute an optimal discrete path $p'_{\epsilon_i}(v, t)$ from v to t. We say that a point v is a *useful point* if either $\|p'_{\epsilon_i}(s, v)\| + \|p'_{\epsilon_i}(v, t)\| \leq (1 + \epsilon_i) \cdot \|p'_{\epsilon_i}(s, t)\|$ or v is a vertex; we say that a Steiner segment is a *useful segment* if at least one of its endpoints is useful. An optimal path p_{opt} will not pass through a useless segment, and therefore in the next stage the algorithm can avoid putting more Steiner points in this segment.

1. $i \leftarrow 1$
2. construct a discretization $\mathcal{G}'_{\epsilon_i} = \mathcal{G}_{\epsilon_i}$.
3. **repeat**
4. compute $p'_{\epsilon_i}(s, t)$ in $\mathcal{G}'_{\epsilon_i}$.
5. **if** $i = d$ **then return** $p'_{\epsilon_i}(s, t)$.
6. continue to compute $p'_{\epsilon_i}(s, v)$ for each point v in $\mathcal{G}'_{\epsilon_i}$ until $\|p'_{\epsilon_i}(s, v)\|$ grows beyond $(1 + \epsilon_i) \cdot \|p'_{\epsilon_i}(s, t)\|$.
7. apply Dijkstra's algorithm in a reversed way, and compute $p'_{\epsilon_i}(v, t)$ for any searched point v.
8. $\mathcal{G}'_{\epsilon_{i+1}} \leftarrow \emptyset$
9. **for** each useful point $v \in \mathcal{G}'_{\epsilon_i}$
10. add v into $\mathcal{G}'_{\epsilon_{i+1}}$
11. **for** each point $v \in \mathcal{G}_{\epsilon_{i+1}}$
12. **if** v is located inside a useful Steiner segment of $\mathcal{G}'_{\epsilon_i}$ **then**
13. add v into $\mathcal{G}'_{\epsilon_{i+1}}$
14. $i \leftarrow i + 1$

Algorithm 1: Adaptive

Each stage contains a forward search and a backward search. These two searches can be performed simultaneously using Dijkstra's two-tree algorithm [8].

To prove the correctness of our multiple-stage approximation algorithm, it suffices to show the following theorem:

Theorem 2. *For any optimal path $p_{opt}(s, t)$, in each $\mathcal{G}'_{\epsilon_i}$ there is a discrete path $p'(s, t)$ with a cost no more than $(1 + \epsilon_i) \cdot \|p_{opt}(s, t)\|$ that neighbors $p_{opt}(s, t)$.*

Proof. We prove by induction.
Basic Step: When $i = 1$, $\mathcal{G}'_{\epsilon_i} = \mathcal{G}_{\epsilon_1}$, and therefore the proposition is true, according to Property 1.
Inductive Step: We assume that, for any optimal path $p_{opt}(s,t)$, $\mathcal{G}'_{\epsilon_i}$ contains a discrete path $p'(s,t)$ neighboring $p_{opt}(s,t)$ such that $\|p'(s,t)\| \leq (1+\epsilon_i) \cdot \|p_{opt}(s,t)\|$. We first show that $p_{opt}(s,t)$ will not pass through any useless Steiner segment $\overline{u_1 u_2}$ in $\mathcal{G}'_{\epsilon_i}$. Suppose otherwise that $p_{opt}(s,t)$ passes through a point between u_1 and u_2. According to the induction hypothesis, we can construct a discrete path $p'(s,t)$ from s to t with a cost no more than $(1+\epsilon_i) \cdot \|p_{opt}(s,t)\|$ that neighbors $p_{opt}(s,t)$. This implies that $p'(s,t)$ passes through either u_1 or u_2. W.L.O.G. we assume that $p'(s,t)$ passes through u_1. Because $\|p_{opt}(s,t)\| \leq \|p'_{\epsilon_i}(s,t)\|$, the cost of $p'(s,t)$ is no more than $(1+\epsilon_i) \cdot \|p'_{\epsilon_i}(s,t)\|$. This is a contradiction to the fact that $\|p'_{\epsilon_i}(s,u_1)\| + \|p'_{\epsilon_i}(u_1,t)\| > (1+\epsilon_i) \cdot \|p'_{\epsilon_i}(s,t)\|$, as $p'(s,t)$ cannot be better than the concatenation of $p'_{\epsilon_i}(s,u_1)$ and $p'_{\epsilon_i}(u_1,t)$.

Since any optimal path from s to t will not pass through a useless Steiner segment, $\mathcal{G}'_{\epsilon_{i+1}}$, which includes all the Steiner points of $\mathcal{G}_{\epsilon_{i+1}}$ except those inside useless Steiner segments, contains every discrete path in $\mathcal{G}_{\epsilon+1}$ that neighbors one of the optimal paths from s to t. This finishes the proof. □

The adaptive discretization method has both pros and cons when compared against the fixed discretization method. It has to run a discrete search algorithm on d different graphs, and each time it involves both forward and backward searches. However, in the earlier stages it explores approximate optimal paths with high error tolerance, while in later stages, as it gradually reduces the error tolerance, it only searches approximate optimal paths in a small subspace (that is, the useful segments of the boundary edges) instead of the entire original space (all boundary edges). Our experimental results show that, when the desired error tolerance ϵ is small, the adaptive discretization method performs more efficiently than the fixed discretization.

This discretization method can also be applied to other geometric optimal path problems, such as the time-optimum movement planning problem in regions with flows [9], the anisotropic optimal path problem [10, 11], and the $3\mathcal{D}$ Euclidean shortest path problem [12, 13].

4 Heuristics for BUSHWHACK

The BUSHWHACK algorithm was originally designed for the weighted region optimal path problem [5] and was later generalized to a class of *piecewise pseudo-Euclidean optimal path problems* [7]. BUSHWHACK, just like Dijkstra's algorithm, is used to compute optimal discrete paths in a graph $\mathcal{G}_\epsilon$ generated by a discretization scheme. Unlike Dijkstra's algorithm, which applies to any arbitrary weighted graph, BUSHWHACK is adept at finding optimal discrete paths in graphs derived from geometric spaces with certain properties, one of which being the following:

Property 2. Two optimal discrete paths that originate from a same source point cannot intersect in the interior of any region.

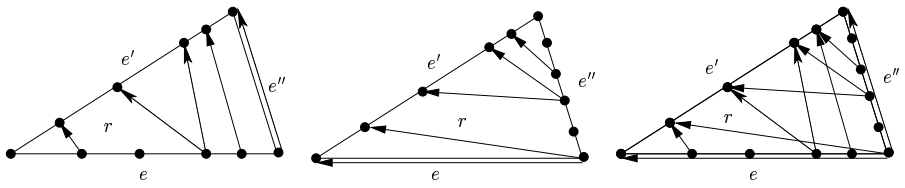
One implication of Property 2 is that, if two edges $\overline{v_1v_2}$ and $\overline{u_1u_2}$ of $\mathcal{G}_\epsilon$ intersect inside region r, they cannot both be *useful*. An edge is said to be useful if it contributes to optimal discrete paths that originate from s. To exploit this property, BUSHWHACK maintains a list $\text{ILIST}_{e,e'}$ of *intervals* for each pair of boundary edges e and e' such that e and e' are on the border of the same region r. A point v is said to be *discovered* if an optimal discrete path $p'_{opt}(s,v)$ has been determined. $\text{ILIST}_{e,e'}$ contains for each discovered point $v \in e$ an interval $I_{v,e,e'}$ defined as the following: $I_{v,e,e'} = \{v^* \in e' | w_r \cdot |\overline{vv^*}| + \|p'_{opt}(s,v)\| \leq w_r \cdot |\overline{v'v^*}| + \|p'_{opt}(s,v')\| \forall\ v' \in \text{PLIST}_e\}$. Here PLIST_e is the list of all discovered points on e. We say that edge $\overline{vv^*}$ is *associated* with interval list $\text{ILIST}_{e,e'}$ if $v \in e$ and $v^* \in I_{v,e,e'}$.

It is easy to see that any edge $\overline{vv^*}$ that crosses region r is useful only if it is associated with an interval list inside r. If m is the number of Steiner points placed on each boundary edge, the total number of edges associated with interval lists inside a region r is $\Theta(m)$. Dijkstra's algorithm, on the other hand, has to consider all $\Theta(m^2)$ edges inside r. By avoid accessing most of the useless edges, BUSHWHACK takes only $O(nm \log nm)$ time to compute an optimal discrete path from s to t, as compared to $O(nm^2 + nm \log nm)$ time for Dijkstra's algorithm.

In this section we introduce BUSHWHACK$^+$, a variation of BUSHWHACK. On the basis of the original BUSHWHACK algorithm, BUSHWHACK$^+$ uses several cost-saving heuristics. The necessity of the first heuristic is rather obvious. Let r be a triangular region with boundary edges e, e' and e''. There are six interval lists for each triangular region r, one for each ordered pair of boundary edges of r. Although the edges associated with the same interval list do not intersect with each other, two edges associated with different interval lists may still intersect inside r. Therefore, BUSHWHACK may still use some intersecting edges to construct candidate optimal paths. Figure 3.a and 3.b show the edges associated with $\text{ILIST}_{e,e'}$ and $\text{ILIST}_{e'',e'}$, respectively. Figure 3.c shows that these two sets of edges intersect with each other, meaning that some of them must be useless.

To address this issue, BUSHWHACK$^+$ merges $\text{ILIST}_{e,e'}$ and $\text{ILIST}_{e'',e'}$ into a single list $\text{ILIST}_{r,e'}$. Any point $v^* \in e'$ is included in one and only one interval in this list. (In BUSHWHACK, every such point is included in two intervals, one in $\text{ILIST}_{e,e'}$ and one in $\text{ILIST}_{e'',e'}$.) More specifically, for any discovered point $v \in e \cup e''$, $v^* \in I_{v,r,e'}$ if and only if $w_r \cdot |\overline{vv^*}| + \|p'_{opt}(s,v)\| \leq w_r \cdot |\overline{v'v^*}| + \|p'_{opt}(s,v')\|$ for any other discovered point $v' \in e \cup e''$. Therefore, any two edges associated with $\text{ILIST}_{r,e'}$ will not intersect with each other inside r. As BUSHWHACK$^+$ constructs candidate optimal paths using only edges associated with interval lists, it would avoid using both of two intersecting edges $\overline{v_1v_1^*}$ and $\overline{v_2v_2^*}$ if $v_1, v_2 \in e \cup e''$ and $v_1^*, v_2^* \in e'$.

The second heuristic is rather subtle. It reduces the size of QLIST, the list of candidate optimal paths. Possible operations on this list include inserting a



(a) Edges associated with $\text{ILIST}_{e'',e'}$

(b) Edges associated with $\text{ILIST}_{e'',e'}$

(c) Edges associated with either interval lists

Fig. 3. Intersecting Edges Associated with Two Interval Lists

new candidate optimal path and deleting the minimum cost path in the list. On average, each such operation costs $O(\log(nm))$ time. As each iteration of the algorithm will invoke one or more such operations, it is very important to contain the size of QLIST.

In the original BUSHWHACK, for any point $v \in e$, QLIST may contain six or more candidate optimal paths from s to v. Among these paths, four of them are propagated through edges associated with interval lists, while the remaining ones are extended to v from left and right along the edge e. This is a serious disadvantage against Dijkstra-based approximation algorithm, which keeps only one path from s to v in the Fibonacci heap for each Steiner point v. When n is relatively large, the performance gain of BUSHWHACK by accessing only a small subgraph of $\mathcal{G}_\epsilon$ will be totally offset by the time wasted on a larger path list.

If multiple candidate optimal paths for v are inserted into QLIST, BUSHWHACK keeps each of them until it is time to extract that path from QLIST, even though it can be immediately decided that all of those paths except one cannot be optimal (by comparing the costs of those paths). This is because BUSHWHACK would generate new candidate optimal paths using these paths in different ways. A (non-optimal) path may lead to the generation of a true optimal discrete path and therefore it cannot be simply discarded. What BUSHWHACK does is to keep the path in QLIST until this path becomes the minimum cost path. At that time, it will be extracted from QLIST and a new candidate optimal path generated from the old path will be inserted into QLIST.

BUSHWHACK$^+$, however, uses a slightly different propagation scheme to avoid keeping multiple paths with the same ending point. Let $p(s, v')$ be a candidate optimal path from s to v' that has just been inserted into QLIST. If there is already another candidate optimal path $p'(s, v')$ in QLIST, instead of keeping both of them in QLSIT, BUSHWHACK$^+$ will take the more costly one, say $p'(s, v')$, and immediately extract it from QLIST. This extracted path will be processed as if it had been extracted in the normal situation (in which it would have been the minimum cost path in the list). This is, in essence, a "propagation-in-advance" strategy that is somewhat contradictory to "lazy" propagation scheme

of BUSHWHACK. It may cause accessing edges unnecessarily. It is a trade-off between reducing the path list size and reducing the number of edges accessed.

5 Preliminary Experimental Results

In order to provide a performance comparison, we implemented using Java the following three algorithms: 1) BUSHWHACK$^+$; 2) *pure Dijkstra's algorithm*, which searches every incident edge of a Steiner point in $\mathcal{G}_\epsilon$; 3) two-stage adaptive discretization method, which uses pure Dijkstra's algorithm for each stage and chooses $\epsilon_1 = \frac{\epsilon}{2}$. All the timed results were acquired from a Sun Blade-1000 workstation with 4GB memory.

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 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. Therefore, in each example there are 5192 triangular faces. For each triangular face r, we assign to r a unit weight w_r that is equal to $1 + 10 \tan \alpha_r$, where α_r is the angle between r and the horizontal plane.

Table 1. Statistics of running time (in seconds) and number of visited edges per region

Algorithm	BUSHWHACK$^+$	pure Dijkstra	adaptive discretization
$\frac{1}{\epsilon} = 3$	156.9 / 2371	243.0 / 16558	281.3 / 10877
$\frac{1}{\epsilon} = 5$	290.7 / 4603	711.0 / 55797	570.2 / 24041
$\frac{1}{\epsilon} = 7$	440.6 / 7098	1506.0 / 124086	1054.7 / 40827
$\frac{1}{\epsilon} = 9$	631.9 / 9795	2672.5 / 224987	1528.9 / 60495

For each TIN, we ran the three 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. From Table 1, it is easy to see that, when $\frac{1}{\epsilon}$ grows, the running times of the BUSHWHACK$^+$ algorithm and adaptive discretization method are growing much slower than that of the pure Dijkstra's algorithm. 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 and the running time are closely correlated.

6 Conclusion and Acknowledgement

In this paper we provided several improvements on the approximation algorithms for the weighted region optimal path problem: 1) a compact discretization scheme that removes the dependency on the unit weight ratio; 2) an adaptive discretization that selectively put Steiner points with high density on boundary edges; and 3) a revised BUSHWHACK algorithm with two cost-saving heuristics. This

work is supported by NSF ITR Grant EIA-0086015, DARPA/AFSOR Contract F30602-01-2-0561, NSF EIA-0218376, and NSF EIA-0218359.

References

1. Mitchell, J.S.B.: Geometric shortest paths and network optimization. In Sack, J.R., Urrutia, J., eds.: Handbook of Computational Geometry. Elsevier Science Publishers B.V. North-Holland, Amsterdam (2000) 633–701
2. Mitchell, J.S.B., Papadimitriou, C.H.: The weighted region problem: Finding shortest paths through a weighted planar subdivision. Journal of the ACM **38** (1991) 18–73
3. Mata, C., Mitchell, J.: A new algorithm for computing shortest paths in weighted planar subdivisions. In: Proceedings of the 13th Annual ACM Symposium on Computational Geometry. (1997) 264–273
4. Aleksandrov, L., Lanthier, M., Maheshwari, A., Sack, J.R.: 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. (1998) 11–22
5. Reif, J.H., Sun, Z.: An efficient approximation algorithm for weighted region shortest path problem. In: Proceedings of the 4th Workshop on Algorithmic Foundations of Robotics. (2000) 191–203
6. Aleksandrov, L., Maheshwari, A., Sack, J.R.: Approximation algorithms for geometric shortest path problems. In: Proceedings of the 32nd Annual ACM Symposium on Theory of Computing. (2000) 286–295
7. Sun, Z., Reif, J.H.: 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. (2001) 160–171
8. Helgason, R.V., Kennington, J., Stewart, B.: The one-to-one shortest-path problem: An empirical analysis with the two-tree dijkstra algorithm. Computational Optimization and Applications **1** (1993) 47–75
9. Reif, J.H., Sun, Z.: 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. (2001) 450–461
10. Lanthier, M., Maheshwari, A., Sack, J.R.: 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. (1999) 524–533
11. Sun, Z., Reif, J.H.: On energy-minimizing paths on terrains for a mobile robot. In: Proceedings of the 2003 IEEE International Conference on Robotics and Automation. (2003) To appear.
12. Papadimitriou, C.H.: An algorithm for shortest-path motion in three dimensions. Information Processing Letters **20** (1985) 259–263
13. Choi, J., Sellen, J., Yap, C.K.: Approximate Euclidean shortest path in 3-space. In: Proceedings of the 10th Annual ACM Symposium on Computational Geometry. (1994) 41–48