Optimistic Shortest Paths on Uncertain Terrains

Chris Gray*
Department of Computer Science
University of British Columbia

William Evans†
Department of Computer Science
University of British Columbia

Abstract

We show that finding the optimistic shortest path on an uncertain terrain is NP-hard using a reduction similar to Canny and Reif's reduction of 3SAT to 3D Euclidean shortest path.

1 Introduction

Shortest path problems are a well-studied class of problems in theoretical computer science. One particularly applicable type of shortest path problem is to find the geodesic shortest path on a terrain. This type of algorithm finds the shortest path between two points that stays on the surface of a terrain. The most popular methods for finding such a shortest path involve a variant of Dijkstra's algorithm and run in time approximately $O(n^2)$ in the size of the terrain [5, 4].

These algorithms for calculating shortest paths on a terrain require a precise input; any errors in measuring the terrain translate into errors in the output of the algorithms. What appears to be a shortest path according to the given input may turn out to be longer than an alternate path in reality. Uncertain terrains are a new model for acknowledging and dealing with these errors.

In this paper, we consider one version of the shortest path problem on uncertain terrains: the optimistic shortest path. Essentially, we would like to find the path whose length is smallest over all paths *and* over all possible real terrains.

This seems to be a slight generalization of the traditional geodesic shortest path problem. We show that it is, in fact, more akin to the problem of finding the shortest path in three dimensions that avoids polyhedral obstacles. This problem was shown to be NP-hard by Canny and Reif [3] in 1986. It is from their proof that our work is derived.

1.1 Definitions of Terms

A *terrain* is a two-dimensional surface in three-dimensional space obtained by triangulating the projection into the xy-plane of a set of n vertices in 3D (no two of which share the same x, y coordinates) and using linear interpolation to obtain the surface within each triangle.

An *uncertain terrain* is a set of n vertices in the xy-plane, each of which has an associated z interval, and a triangulation of the (xy-projection of the) vertices. Any terrain whose projection into the xy-plane produces the same triangulation and whose vertices have a z value within the corresponding z interval is *consistent* with the uncertain terrain. A vertex in an uncertain terrain with a z interval that contains more than a single value is called an *uncertain vertex*.

A *path* is a sequence of x, y coordinates. A path can be placed on a terrain to create a sequence of 3D points by associating with each x, y coordinate the corresponding z value from the terrain. The length of a path on a terrain is the sum of the Euclidean distances between adjacent 3D points after placing the path on the terrain.

The *shortest route* from s to x is a sequence of terrain edges traversed by a shortest path from s to x. A *path class* is an equivalence class of points x that all have the same shortest route(s) from s.

1.2 Statement of the Problem

Given an uncertain terrain and two points s and t in the xy-plane, we want to find a path from s to t that has the shortest length among all paths from s to t where the path length is measured on the consistent terrain that minimizes the length. Cast as a decision problem, this becomes, "Is there a path from s to t on some consistent terrain of length at most K?" We call this the optimistic shortest path since we are measuring a path on the consistent terrain that minimizes its length.

1.3 Overview

The decision problem is NP-hard. It is not clearly in NP due to the algebraic complexity possible in shortest paths on terrains [2].

To show that the problem is NP-hard, we will use a reduction similar to Canny and Reif's reduction of 3SAT to the problem of finding shortest paths in 3D with obstacles [3]. In fact, we construct gadgets (splitters, shufflers, etc.) that perform the same functions as the corresponding Canny-Reif gadgets, but are part of an uncertain terrain rather than being 3D objects.

Let Φ be a 4SAT formula with n variables and m clauses. Our construction uses n path splitters to create 2^n path classes, each corresponding to a truth assignment for the n variables. Then it directs the paths through m clause boxes, each of which has 4 literal filters. The purpose of each clause

*cmg@cs.ubc.ca. Supported by GEOIDE Networks of Centres of Excellence

†will@cs.ubc.ca. Supported in part by a Natural Sciences and Engineering Research Council of Canada Research Grant

box is to lengthen each path class that does not have a setting that satisfies the clause. After the clause boxes, the paths are directed through another cascade of n path splitters, this time reversed. After the last one, there is only one path class left. We place t in this path class, and by measuring the shortest path length from s to t, we can determine if any one of the 2^n shortest routes have not been lengthened. If so, then we know that our formula is satisfiable.

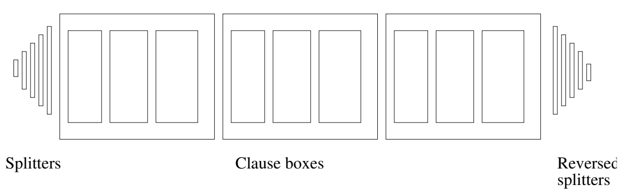

Figure 1: Overview of the construction

We will now look at the gadgets in greater detail.

2 The Gadgets

2.1 Rotator

A basic operation that our gadgets must perform is to rotate the orientation of the group of path classes by $90°$. This is important as a building block of other gadgets and is performed by a *rotator*, which is like a square of paper folded along its diagonal. See Figure 2 for an example of a rotator of width 1 located at $(0,0,0)$ with a thickness parameter γ that is essentially 0.

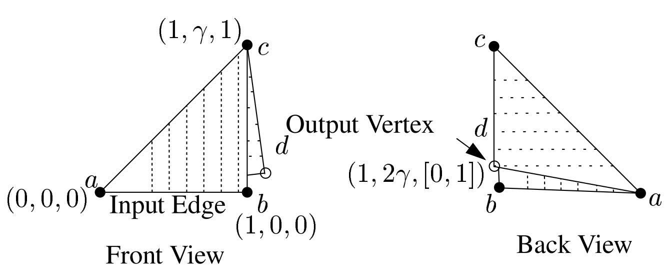

Figure 2: Front and back view of a rotator showing shortest paths across it as dotted lines

Since the shortest paths leave the rotator in a vertical line, we have changed the orientation of the paths from horizontal to vertical.

2.2 Path Splitter

The purpose of a path splitter, as its name suggests, is to take n path classes and split them into $2n$ path classes.

To create the splitter, we construct the uncertain terrain so that any shortest path from s to t must enter at or near the uncertain vertex d, cross either edge ef or fg, and cross output edge eg. The height of the input vertex d closely determines the point at which the path crosses the output edge. See Figure 3.

Note that splitters are like two rotators that have been joined together.

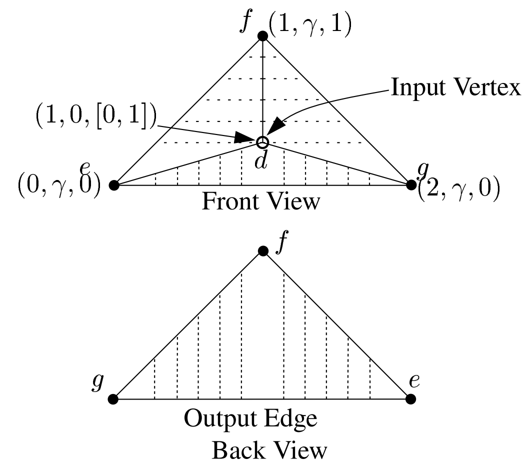

Figure 3: The splitter, showing the doubling of the number of path classes

Given these gadgets, we are ready to describe the first part of the construction. In front of the initial point s, we construct a wall that is high enough that any shortest path from s to t must go around the wall to the left or right. This creates two shortest path classes. Following this wall is a sequence of n connected rotators and splitters, joined at their common vertex d and increasing in size by a factor of 2 each time. When we join the rotators and the splitters at the vertex d, we leave a gap between vertex a on the rotator and vertex e on the splitter. It is possible to fill this gap with a very high wall (or a very low trench), but for now we will leave gaps unfilled and assume that no shortest path can afford to pass over a gap.

The output edge of the last splitter may be reached from s via 2^n shortest routes. Each of these shortest routes intersects the final output edge at a different location along the output edge.

We number the resulting path classes along this edge from 0 to $2^n - 1$ and associate them with the 2^n possible truth assignments to the variables of the formula Φ.

The rest of the construction lengthens the paths in those path classes that correspond to unsatisfying truth assignments.

2.3 Path Shuffler

The path shuffler is designed to output a perfect shuffle of the path classes input to it. A perfect shuffle takes the sequence 01234567 and permutes it to 04152637, for example. Notice that this is the same as doing a circular bitwise right shift of each number in the sequence. Thus, if 01234567 is the order of path classes on the input edge of a shuffler, 041526374 is the order on the output vertex.

The implementation of the shuffler is similar to the implementation of the splitter. The shuffler can also be seen as being composed of two rotators put together. Assume that the paths are δ apart. We will split the group of paths into two parts, so that half go to one side and half go to the other. When the group has been split, one of the halves will be raised a slight amount $(1/2 + \delta/4)$ and the other half will be lowered the same amount. The paths will then be rotated so that they interleave vertically.

In order to ensure that paths do not cross into the wrong group, we put a gap between the two groups of paths. This is shown in Figure 4 as the gray triangle.

The shortest paths leave the rotators through point a or point b in Figure 4. They then travel along edge ac or bc to point c. Point c is the input vertex to a reversed rotator (not shown) that rotates the group of paths back to horizontal.

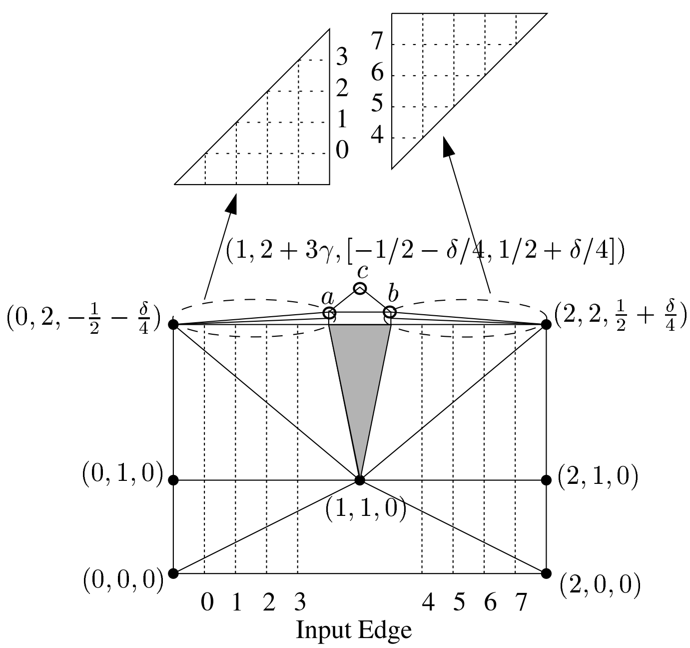

Figure 4: The shuffler viewed from above. The circled regions contain rotators, shown projected into the xz plane above the shuffler at the correct relative heights. The vertex marked c is the output vertex.

2.4 Literal Filter

The literal filter is a meta-gadget. It consists of n path shufflers and one barrier. We use the literal filter in order to stretch out paths that have a truth value that is opposite the truth value asked for in a clause. For example, if we want to express $\overline{x_3}$, we stretch all paths in those path classes whose corresponding truth assignment has a 1 in the third position. To do this, we take the paths through three shufflers. At this point we have all the paths with a 1 in their third bit in the left half of the output edge and all the paths with a 0 in the right half. Then we put a barrier on the side with the ones to stretch all of them. Finally, we put $n-3$ shufflers to get the paths back in their original positions.

2.5 Clause Box

A clause box consists of two cascaded splitter / rotator combinations that have the effect of splitting each path into four copies. These are followed by a literal filter for each literal in the clause put in parallel so that each of the copies of a given path goes through exactly one literal filter. Finally, we add two reversed splitter / rotator combinations.

A short path P exists through this gadget if and only if P belongs to a path class that corresponds to a truth assignment that satisfies at least one literal in the clause.

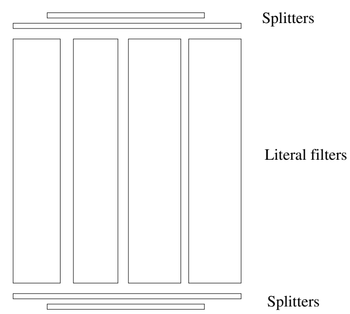

Figure 5: The clause box

3 The Hardness of Optimistic Shortest Paths

We will now show that each gadget increases the length of the shortest path traveling through it by a specified amount. We will also show that a path that travels through a gadget in a non-shortest manner will be detectable at the end.

Lemma 1 *For a rotator of width w, there exists a path from a point in $([x_0, x_1], 0, 0)$ on the input edge to the output vertex of length less than or equal to $w+2\gamma$ only if the path exits through the output vertex in the range $(w, 2\gamma, [x_0-\varepsilon, x_1+\varepsilon])$ where $\gamma = \frac{\varepsilon^2}{8w}$.*

Proof. In order to find the locally optimal path across a rotator, first we consider the unfolding of the rotator when the output vertex is at its lowest possible elevation and $\gamma = 0$. The shortest path from the point $(x_0, 0, 0)$ is clearly the straight line across to $(x_0, w, 0)$. If we refold the rotator, this point goes to $(w, 2\gamma, x_0)$. Since points can only exit via the output vertex, the output vertex must move to that point.

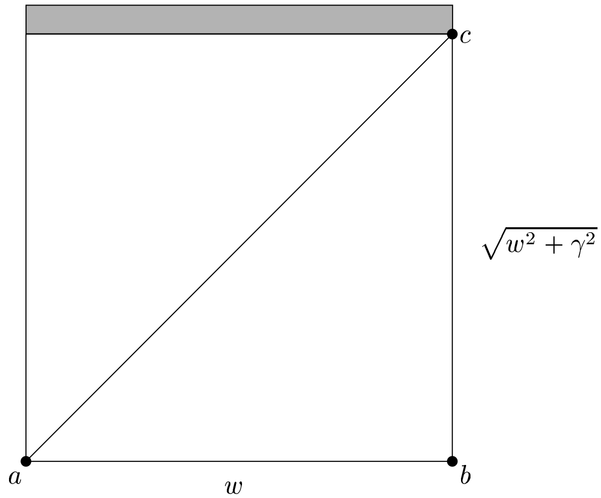

Figure 6: The unfolding has three fixed vertices and the fourth inside the shaded strip.

Now consider the case when $\gamma \neq 0$. Whereas the unfolding of the rotator was a square with edge length w, it is now a quadrilateral. In the unfolding (see Figure 6), the triangle abc forms a right-angled triangle with leg lengths w and $\sqrt{w^2+\gamma^2}$. The fourth vertex d falls within a strip of width γ above the $w \times \sqrt{w^2+\gamma^2}$ rectangle. We can construct an approximation of the quadrilateral that is a $w \times w$ square with a 2γ strip added to the top to represent the uncertain location of d; the output vertex d must be inside this strip.

If we consider a path of length $w + 2\gamma$ across this quadrilateral from an arbitrary input point, the width of the region that the output point could potentially be in is $4\sqrt{\gamma(w+\gamma)}$. Therefore, if $\gamma = \frac{\varepsilon^2}{8w}$, this region has a width that is less than 2ε. Thus, if the input point starts in range $[x_0, x_1]$, it will exit in the range $[x_0 - \varepsilon, x_1 + \varepsilon]$. □

Lemma 2 *For a shuffler of width* $2w$*, there exists a path from a point in* $([x_0, x_1], 0, 0)$ *on the input edge to the output vertex of length less than or equal to* $2w + \sqrt{w^2 + (w/2 + \delta/4)^2} + 2\gamma$ *only if the output vertex is either in* $(w, 2w+3\gamma, [x_0 - w/2 - \delta/4 - \varepsilon, x_1 - w/2 - \delta/4 + \varepsilon])$ *or in* $(w, 2w+3\gamma, [x_0 + w/2 + \delta/4 - \varepsilon, x_1 + w/2 + \delta/4 + \varepsilon])$*, depending whether* $x_0 \leq x_1 < 1$ *or not.*

Proof. The locally optimal motion for a path entering a shuffler is for the path to travel in a straight line to the rotator across from it. Once it is at the rotator, Lemma 1 applies. □

From Lemma 1 and Lemma 2, we can deduce the maximum distance possible for any locally optimal path that corresponds to a truth assignment that satisfies the formula Φ. This distance will always be less than the minimum distance of any path that is either not locally optimal or that corresponds to a truth assignment that does not satisfy the formula Φ. We ensure that no path crosses from a path class representing one truth assignment to a path class that represents another by making the initial wall very wide. Since the width of a path class only grows by 2ε for each gadget, the path classes are never close enough that changing from one to the other is a locally optimal motion. Thus we have

Theorem 3 *The optimistic shortest path problem is NP-hard. That is, we can find a path that has length* $\leq K$ *if and only if we can find a setting of variables that satisfies the formula* Φ *in 4-CNF form.*

References

[1] Tetsuo Asano, David Kirkpatrick, and Chee K. Yap. d_1-optimal motion for a rod. In *Proceedings of the Twelfth Annual Symposium On Computational Geometry (ISG '96)*, pages 252–263, New York, May 1996. ACM Press.

[2] Chanderjit Bajaj. The algebraic degree of geometric optimization problems. *Discrete Comput. Geom.*, 3(2):177–191, 1988.

[3] John Canny and John Reif. New lower bound techniques for robot motion planning problems. In Ashok K. Chandra, editor, *Proceedings of the 28th Annual Symposium on Foundations of Computer Science*, pages 49–60, Los Angeles, CA, October 1987. IEEE Computer Society Press.

[4] Sanjiv Kapoor. Efficient computation of geodesic shortest paths. In ACM, editor, *Proceedings of the thirty-first annual ACM Symposium on Theory of Computing: Atlanta, Georgia, May 1–4, 1999*, pages 770–779, New York, NY, USA, 1999. ACM Press.

[5] Joseph S. B. Mitchell, David M. Mount, and Christos H. Papadimitriou. The discrete geodesic problem. *SIAM J. Comput.*, 16(4):647–668, 1987.