# GEOMETRIC BICRITERIA OPTIMAL PATH PROBLEMS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Christine Diane Piatko

August 1993

Geometric Bicriteria Optimal Path Problems

Christine Diane Piatko, Ph.D.
Cornell University 1993

A bicriteria optimal path simultaneously satisfies two bounds on two measures of path quality. The complexity of finding such a path depends on the particular choices of path quality. This thesis studies bicriteria path problems in a geometric setting using several pairs of path quality, including: path length measured according to different norms ($L_p$ and $L_q$); Euclidean length within two or more classes of regions; total turn and Euclidean length; total turn and number of links; and Euclidean length and number of links.

For several cases, finding the bicriteria optimal path is shown to be NP-hard. These NP-hard cases include minimizing path length in two different norms, minimizing travel through two regions, and minimizing length and total turn. In the last case, an $O(En^2N^2)$ pseudo-polynomial time algorithm to find an approximate answer is presented. In contrast, when the two measures of path quality are total turn and number of links, an $O(E^3 n \log^2 n)$ exact algorithm is given.

A main result of this thesis examines minimizing the Euclidean length and number of links of a path. When the geometric setting of this problem is a polygon without holes, this thesis presents an $O(n^3k^3 \log(Nk/\epsilon^{1/k}))$ algorithm to find a $k$-link path with Euclidean length at most $1+\epsilon$ times the length of the shortest $k$-link path. A faster algorithm for a relaxed case, when the output path is allowed to have $2k$ links, is presented for a polygon with or without holes.

Finally, some approximation algorithms are outlined for finding a minimum link path among polyhedral obstacles.

# Biographical Sketch

Christine Piatko was born in Queens, New York on October 12, 1965. She lived in Brooklyn for the first 21 years of her life. She attended New York University where as a Presidential Scholar she was sent to visit Russia, Egypt, Greece, and Hawaii. She also found time to earn a Bachelor of Arts Degree in Honors Computer Science and Mathematics by June 1986. She entered the Cornell Computer Science Department Fall 1986 and received a Master of Science degree from Cornell University in May 1989. Along the way she met the love of her life, Richard Chang, and married him in July 1991. Following him to Maryland in Fall 1992, she decided to face the real world by becoming a federal employee at the National Institute of Standards and Technology in January 1993.

# Acknowledgements

I extend my warmest thanks to my advisor and friend, Joe Mitchell. His enthusiasm for computational geometry and for brainstorming on problems in general has been quite inspiring. He has always been willing to share his advice about everything from geometry to job-hunting to house-buying. Many thanks, Joe!

I am also grateful to my collaborator, Estie Arkin. Special thanks to Estie, Joe, Shira, and Roni for their warm hospitality during my visits to Stony Brook.

The other members of my committee, Dan Huttenlocher, Bruce Donald, and Jim Renegar also have my thanks for their advice and encouragement, and for helping me learn about computer vision, robotics and the theory of the reals.

Everyone who participated in the computational geometry worksessions made geometry seem exciting and fun. Special thanks to Paul Chew, Robert Freimer, Paul Heffernan, Klara Kedem, Scott Mitchell, Shmuel Onn, and Erik Wynters for many insightful and interesting discussions. Thanks also to David Shmoys for insight about fractional knapsack problems.

I am especially grateful to my husband Richard Chang, who has been my best friend, confidante and constant companion. He waited in orbit for me in Ithaca for an extra year, and patiently supported me no matter how exasperated (or exasperating!) I became. Thanks for cooking so many dinners, and making so many pots of coffee. I am also very grateful for your knowledge of LaTeX and for your help with drawing figures.

My mom, my dad, and my brother Peter have provided unconditional support during my years in graduate school. They have always encouraged me to learn, even if they were a bit puzzled as to how I could live in Ithaca for so many years.

I thank my friends at Cornell for companionship, great dinners, conversations, bridge games, puzzle-solving, hockey-playing, and for providing general distraction from thesis work. Special thanks to Jennifer, Robert, Bill, Paul, Gretta, Daniela, Chet, Jim, Samir, Beate, and Lisa for making Ithaca a more bearable place! To the Flying Diskettes, the computer science women's intramural hockey team, thanks for winning the championship while I was a captain in 1990! Thanks also to my friends from New York University, Roberto, Adela, and Thomas, for providing long-distance friendship throughout my graduate school career.

I have also been grateful for the spiritual guidance of Father Joseph Chupeck and the parish of the Holy Trinity Orthodox Church in Elmira.

I have received a generous amount of financial support for my studies. I gratefully acknowledge the support I received from Cornell University, the Local 32BJ Service Workers' Union, the National Science Foundation, and the Air Force. The views reflected in this thesis are my own, and do not reflect the views of these organizations.

Finally, I thank Computer Science Department of the University of Maryland at Baltimore County for allowing me to use their facilities while finishing this document.

# Table of Contents

# List of Figures

# Chapter 1

# Introduction

Consider planning the motion of a robot across a room cluttered with obstacles. This problem of motion planning has been considered from many perspectives. From the perspective of *computational geometry* it has been approached by modelling the robot and obstacles as geometric objects and devising algorithms to decide if there is a feasible path for the robot to follow from its start to its goal. The efficiency of the decision procedure is usually expressed as a function of the complexity of the geometric objects involved.

However, in addition to deciding if a feasible path exists, we may also want to optimize some quality of the path. For example, we may wish to optimize the length of the path, and thus find the shortest travel route for a robot to follow. Unfortunately, it has been shown by Canny and Reif [CR87] that finding a shortest length path from a point $s$ to a point $t$ in three dimensions among polyhedral obstacles is NP-hard. Hence no fast algorithm for this problem is likely to exist.

For this reason, algorithms concerned with planning optimal motions are usually derived for restricted environments. For example, we can consider planning shortest length paths when the environment consists of orthohedral obstacles. Alternatively, we can lower the dimensionality and consider the problem in a planar environment, modelling obstacles as simple polygons. Yet another approach to deal with NP-hardness is to find provably good approximations to the optimal path.

In this thesis we will mainly restrict our attention to a planar environment with polygonal obstacles. We will also consider approximation algorithms to some problems.

## 1.1 Previous and Related Work

There have been many algorithms in computational geometry that produce optimal paths according to some notion of "shortness" in a planar environment. One such natural measure is the total Euclidean length of the path.

The problem of finding shortest Euclidean length paths among obstacles in the plane is well-studied. It is well known that the shortest Euclidean path among obstacles in the plane is a "taut-string" path following segments between obstacle vertices. These segments correspond to a line of sight, or visibility, between two vertices. Alt and Welzl [AW88] showed how to compute this *visibility graph* between vertices efficiently. When there are $n$ obstacle vertices, their algorithm computes the visibility graph in $O(n^2)$ time. Using the standard Dijkstra's shortest path algorithm on this graph one can find the shortest path between the start point and the goal point in $O(n^2)$ time. Later, Ghosh and Mount [GM91] gave an algorithm to compute the visibility graph in output sensitive time. Their algorithm computes the visibility graph in $O(E+n\log n)$ time, where $E$ is the number of visibility graph edges in the scene. Again, using Dijkstra's algorithm to search this graph, one can find the Euclidean shortest path in time $O(E+n\log n)$. Since $E$ can be $O(n^2)$ in the worst case the running time of this algorithm is $O(n^2)$in the worst case. However, it might be expected to be smaller in some applications. Recently, Mitchell [Mit93] gave the first algorithm for this problem that has a subquadratic worst case running time. This algorithm shows promise for the eventual development of an $O(n\log n)$ algorithm for the Euclidean shortest path problem in the plane.

There has also been work on the problem of finding shortest paths according to other notions of "length." These include $L_1$ and fixed orientation metrics. More efficient shortest path algorithms have often been developed in these settings. An $O(n\log^2 n)$ algorithm for finding shortest $L_1$ (rectilinear) length paths among rectilinear (not necessarily disjoint) obstacles was given by Clarkson, Kapoor and Vaidya [CKV87]. This can be improved to $O(n\log n)$ for disjoint rectilinear obstacles [dRLW89]. Mitchell [Mit92] gives $O(n\log^2 n)$ algorithm, improved to $O(n\log n)$ for disjoint obstacles, for shortest $L_1$ length paths amongst obstacles without orientation restrictions (where obstacles are not restricted to being rectilinear). The notion of computing path length using fixed orientation metrics, or convex distance functions, has also been studied [WWW87,Cla87].

Other measurements of path length which are not strictly geometric have also been considered. Mitchell and Papadimitriou [MP91] consider the *weighted region problem*, in which the length of the path depends on its position in the plane. In this problem the plane is divided into regions with associated weights. The length of a path through a region is its Euclidean length multiplied by the weight associated with that region. This can be used to more realistically model different kinds of obstacles. A brick wall might be modelled as an obstacle with infinite weight, a river might have smaller weight, while a smooth highway might have the least weight. Mitchell and Papadimitriou give a polynomial time approximation algorithm for this problem. Using this approximation one can get arbitrarily close to the exact solution.

In another setting, we may want to find a path among the obstacles so that a robot travelling along this path could obey certain physical constraints. A robot may not be able to stop immediately, or to turn quickly. Algorithms that respect physical constraints on path quality have also been studied. Fortune and Wilfong [FW88] consider bounds on the path curvature. They give an exponential decision procedure for reachability when a moving point is not allowed to change direction too quickly. Canny, Donald, Reif and Xavier [CDRX88] consider paths where the travelling object is subject to kinematic and dynamic constraints. Given the object's start position and velocity and its goal position and velocity they find an obstacle-avoiding shortest-time path that respects dynamic constraints on velocity and acceleration. These algorithms are provably good polynomial time approximation algorithms. Canny, Rege and Reif [CRR91] give an exact polynomial space, exponential-time algorithms for a similar problem in the plane, when the object starts and ends at rest, and when the velocity and acceleration are in bounded in the $L_\infty$ norm.

Yet another measure of path length, introduced by Suri [Sur86,Sur87,Sur90], is the number of piecewise linear segments of the path. This is also known as the *link length* of the path. The shortest path is then the one with the fewest segments. In this measure turns are considered to be very expensive, while straight line travel is easy. Suri gave a linear time algorithm to find the minimum-link path in a simple polygon. Ghosh [Gho91] also gave a linear time algorithm for this problem. Hershberger and Snoeyink [HS91] give a more general algorithm that computes the minimum-link path among polygonal obstacles. In their algorithm, the homotopy type of the path (the way a path winds around the obstacles) is specified in advance. For a simple polygon with polygonal holes, when the homotopy type is not known in advance, Mitchell, Rote and Welzl [MRW92] gave an $O(E\alpha(n)\log^2 n)$ algorithm to find the minimum-link path.

Most of the algorithms described above consider paths that are optimal with respect to just a *single* criterion. However, in most applications there is more than one criterion by which path quality can be measured. For example, Euclidean shortest paths among obstacles may be too "kinked", with too many links or too much total curvature, for the motion of a mobile robot through an obstacle course.

Several authors have looked at using more than one objective on path quality. Recently, de Berg et al. [dBvKNO90,dB91,dBvKN91] have studied a version of a path problem that involves link distance and path length in a "combined metric." They study the problem of finding a shortest rectilinear path among rectilinear obstacles, where the measure of path cost is a "combined metric" equal to the sum of the path length ($L_1$ distance, since they work in a rectilinear world) and a constant times the number of turns. For this problem, they obtain an $O(n^2)$ algorithm. Yang, Lee, and Wong [YLW91] also study a general class of rectilinear planar problems in which the obstacle set is rectilinear, and path costs are a function of $L_1$ length *and* the number of bends (rectilinear link distance). They obtain worst-case time bounds of $O(n\log^2 n)$ for their combined metric. Hershberger and Snoeyink [HS91] have examined the special case of finding a shortest $k^*$-link path (where $k^*$ denotes the number of links in a minimum-link path), for paths and obstacle boundaries required to have edges from among a given fixed set of orientations.

All of these results on bicriteria paths in a fixed-orientation world exploit the fact that there exists a *path-preserving graph* that is guaranteed to contain the desired optimal path. The full grid graph is trivially path-preserving, so the rectilinear problems are clearly polynomial-time. The novelty of the above results, therefore, is their clever use of structure to obtain faster than naive algorithms.

This thesis introduces a new set of results to the field of geometric shortest path planning algorithms: we study "bicriteria" path problems in which there are *two* separate objectives in the selection of a path. These results are among the first algorithmic results known for geometric versions of the "bicriteria" optimal path problem, which requires one to find a path that is "short" with respect to *separate* bounds on *two* different criteria. This is in contrast to much of the previous work which has mainly considered bounds on linear combinations of criteria. Also, we do not limit ourselves to problems where objects or paths are restricted to have fixed orientations.

We consider several pairs of criteria for planar paths, including: path length measured according to two different norms ($L_p$ and $L_q$), path length within two or more classes of regions, total turn and Euclidean length, total turn and number of links, and Euclidean length and link distance. It would be interesting to extend this work to deal with more than two criteria simultaneously. However, as we shall see, many of these problems are NP-hard, and hence are believed to be intractable. We leave open the even more difficult multicriteria problems for further research.

Our work on the problem of finding shortest $k$-link paths may also have applications to problems other than path planning, such as map simplification. Replacing complex objects with simpler ones is a useful operation in cartography that has been done manually, and must now be done automatically in geographic information systems. The notion of approximating polygons and subdivisions with fewer links has been studied by Guibas, Hershberger, Mitchell and Snoeyink [GHMS91]. They show that computing the minimum-link subdivision and finding a minimum-link simple polygon of a given homotopy type are NP-hard.

Finally, we note that the problem of computing optimal enclosing $k$-gons is related to our problem of optimizing the length of $k$-link paths. Aggarwal et al. [ACY85] study the minimum *area* enclosing $k$-gon problem, obtaining an exact algorithm of time complexity $O(n^2 \log k \log n)$. In his thesis, Chang [Cha86] discusses some of the local optimality properties that minimum *perimeter* enclosing $k$-gons must satisfy, but leaves open the problem of actually computing them for $k > 3$. It is likely that our methods will be applicable to this open problem.

## 1.2 Overview

This thesis is organized in the following way. In Chapter 2 we review some preliminaries and background material. We review the result that the bicriteria path problem in a graph is NP-hard, and discuss some algorithmic results for this and related bicriteria graph problems. We review some previous results for shortest path planning in a geometric setting. Finally, we prove several geometric lemmas that we will reference later in thesis.

In Chapter 3 we show that two bicriteria length problems are NP-hard. In the first problem we consider finding a path that is short in two different norms simultaneously. In the second problem we consider limiting the length of travel in two regions simultaneously.

In Chapter 4 we show positive results when one criterion is total turn. We show the problem of finding a path with bounded Euclidean length and bounded total turn is NP-hard, but we can give a pseudo-polynomial time approximation algorithm for the problem, whose running time depends on the size of the underlying grid from which the vertices are chosen. We also present a polynomial time algorithm for the problem of computing a path with a bounded number of links and a bounded amount of total turn.

Chapter 5 concerns finding paths with short Euclidean length and a bounded number of links. We show how to find an approximation to this path in a polygon with holes. For the problem in a simple polygon (without holes), we give a full characterization of the structure of shortest $k$-link paths and prove necessary local optimality conditions. Based on our structural results, we give an more accurate approximation algorithm for computing a path with $k$ links and length within $(1 + \epsilon)$ times the length of the shortest $k$-link path.

Finally, in Chapter 6, we consider several approaches for approximating link distance in a polyhedral environment. These algorithms find paths with link length at most a constant factor times the optimal number of links.

# Chapter 2

# Preliminaries

In this chapter, we will cover some basic geometric properties needed to prove the results in this thesis. We will also review what is known about bicriteria path problems in graphs. We also take this opportunity to define some notation and mention previous techniques to compute geodesic and minimum-link paths in a simple polygon. Finally, we prove some geometric lemmas that we will reference throughout the thesis.

## 2.1 Multi/Bicriteria Path Problems

The first observation that one can make about multi-criteria optimization problems is that they tend to be hard. As we will see, even the bicriteria path problem in a graph is NP-hard [GJ79]. One way to address this difficulty is to convert the multicriteria problem into a single-criterion optimization question. For example, if we wish to optimize with respect to two objective bounds $(\mathcal{O}_1, \mathcal{O}_2)$, we could form the linear combination $\mathcal{O}_3 = a\mathcal{O}_1 + b\mathcal{O}_2$, and then optimize with respect to $\mathcal{O}_3$. The resulting path problem is often solvable in polynomial time. However, this approach does not really address the original question. A solution which satisfies $\mathcal{O}_3$ will almost certainly not respect the original desired separate bounds.

To appreciate the difficulty of multicriteria problems, we review the general result for bicriteria paths in graphs [GJ79, p. 214], which forms the basis for several of our constructions. Recall that the following well known Partition problem is NP-complete:

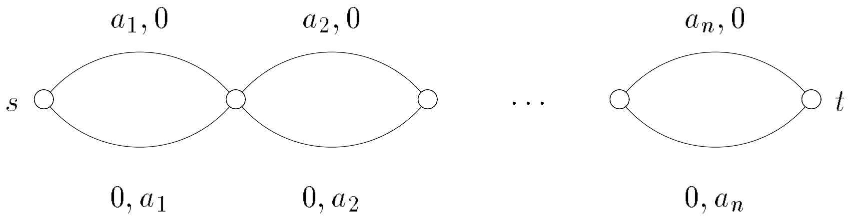



Figure 2.1: Reduction from Partition.

**Partition**

*Instance:* A set $N$ of $n$ items, each with a positive integer weight $a_i$ for $1 \le i \le n$.

*Question:* Does there exist a subset $S \subset N$ such that $\sum_{i \in S} a_i = \frac{1}{2} \sum_{i=1}^{n} a_i$?

Using a reduction from Partition it can be shown that the following bicriteria problem in a graph is also NP-complete:

**Bicriteria Path Problem in a Graph**

*Instance:* A graph $(V, E)$ with positive integer weights $w_i$ and positive integer lengths $l_i$ on its edges, and two distinguished nodes $s$ and $t$.

*Question:* Does there exist a path from $s$ to $t$ with weight $\le W$ and length $\le L$?

**Theorem 2.1** *The Bicriteria Path Problem in a Graph is NP-complete.*

**Proof.** Use a reduction from Partition. Consider a graph with $n+1$ nodes, with $s = v_1$ and $t = v_{n+1}$. Draw two edges joining node $v_i$ to $v_{i+1}$, a "top" edge with length 0 and weight $a_i$, and a "bottom" edge with length $a_i$ and weight 0. Set $L = W = \frac{1}{2} \sum_{i=1}^{n} a_i$. (See Figure 2.1.) A path of length $\le L$ and weight $\le W$ on this graph yields a partition into top and bottom edges that solves the Partition problem. □

The above proof holds if we add a constant $C$ to the lengths (weights) of the top and bottom edges joining $v_i$ and $v_{i+1}$, while adding $nC$ to $L$ ($W$). It is only important that the *difference* between the two lengths (weights) equals $a_i$.

Partition is a weakly NP-complete problem, so it is not surprising that there are pseudo-polynomial time algorithms for this bicriteria problem. In fact, the Bellman-Ford dynamic programming method for shortest paths [Law76, p. 74] provides a polynomial time algorithm for the equal-length (or equal-weight) version.

If the lengths or the weights of the edges are bounded, we can solve the problem in polynomial time, by breaking the arcs into "unit" length or weight segments. This implies a pseudo-polynomial time algorithm for the general bicriteria problem on graphs.

Unfortunately, using the Bellman-Ford dynamic programming method directly results in a rather high running time. Since we will make use of such a dynamic programming method several times we describe a slightly more efficient variation for integer weights here:

**Lemma 2.2** *Consider a graph with $n$ nodes and $E$ edges, with each edge $(u, v)$ having an associated length $\ell(u, v)$ and a positive integer weight $w(u, v)$. Then, a shortest length $s$–$t$ path whose weight is at most $k$ can be computed in time $O(kE)$, for any positive integer $k$.*

**Proof.** Fix a destination node $t$. Let $f(i, u)$ denote the length of a shortest path from $u$ to $t$, subject to the weight of the path being at most $i$. We would like to compute $f(k, s)$. We can use dynamic programming as follows:

1. Let $f(0, t) = 0$, and $f(i, u) = \infty$ for all $i$ ($-k \leq i \leq k$ suffices) and for $u \neq t$.

2. For $i = 1, \ldots, k$ compute the following:

   For each node $u$, compute $f(i, u) = \min_{v \in \nu(u)}\{\ell(u, v) + f(i - w(u, v), v)\}$, where $\nu(u)$ denotes the set of nodes adjacent to $u$. This step takes time $O(E)$. Note that $f(i - w(u, v), v)$ is already known by the time we need to use it, since $w(u, v) > 0$.

Thus, in time $O(kE)$ we will have found the optimal value $f(k, s)$. □

It is not always the true that an obvious polynomial time algorithm results when the lengths or weights of edges are restricted or related in some way. Note the following very similar problems, in which the ratio of edge length to edge weight is closely bounded, are NP-complete (Arkin and Onn, private communication). (In the following, let $l_e$ and $w_e$ be the non-zero length and weight of an edge $e$.)

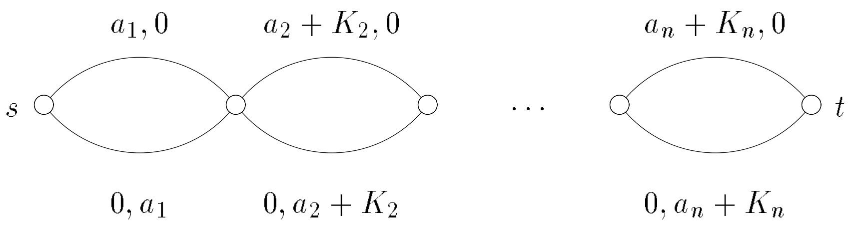



Figure 2.2: An exponential number of pareto-optimal paths.

**Theorem 2.3** *The Bicriteria Path Problem in a Graph with edge lengths and weights restricted so that for every edge e,* $1 - \epsilon \leq l_e/w_e \leq 1 + \epsilon$*, is NP-complete.*

**Proof.** Add a large constant C to both the length and the weight of all arcs, and adding $n \cdot C$ to $L$ and $W$. As noted previously, adding constants in this way will not affect the NP-completeness proof. If C is large enough, then $l_e/w_e$ will be close to one. □

**Theorem 2.4** *The Bicriteria Path Problem in a Graph with edge lengths and weights restricted so that for every edge e,* $l_e < l_f \Rightarrow w_e \leq w_f$ *and* $w_e < w_f \Rightarrow l_e \leq l_f$ *is NP-complete.*

**Proof.** Add $a_i$ to both edges between $v_i$ and $v_{i+1}$. The top edge becomes an edge with length $a_i$ and weight $2a_i$. Split the bottom edge into 2 pieces, one with length and weight $a_i$, the other with length $a_i$ and weight 0. Now add $\epsilon$ to the length of the upper edge. Add $\frac{1}{3}\epsilon$ to one piece of the bottom edge and $\frac{2}{3}\epsilon$ to the other piece of the bottom edge. Also add $n \cdot \epsilon$ to $L$. The desired implications hold for this new graph. Since we have added the same constant to both the top and bottom edges the NP-completeness proof still holds. □

Another approach to find a path which satisfies several criteria is to search for all *pareto-optimal* paths. A path is called *pareto-optimal* if no other path has a better value for one criterion without having a worse value for the other criterion. In the case of a bicriteria path, if the path has values $X$ and $Y$ for the two criteria, the path is pareto-optimal if there is no other path with values $x \leq X$ and $y \leq Y$. Hansen [Han80] gave an algorithm to find all pareto-optimal paths in a graph, in time polynomial in the number of paths and the size of the graph. Experimental studies suggest that the average number of pareto-optimal paths remains very small in practice (e.g., see [MMO]), although in theory this number may be exponential. (See Figure 2.2 for an example of a graph that has an exponential number of different pareto-optimal paths from $s$ to $t$.)

For the two geometric versions of the bicriteria path problem we consider, minimizing (links, length) and (links, total turn), the number of pareto-optimal pairs of *values* of the objectives is clearly polynomial; since the number of links is an integer at most equal to $n$, there can be at most $n$ pareto-optimal pairs of values (links, —). We will show, however, that the number of different *paths* that achieve a given pareto-optimal value pair can be exponential, even in a simple polygon (without holes).

Finally we note that for certain special cases we can find exact solutions to the bicriteria path problem. For example, when the ratio of length to weight is exactly a constant, we can solve the bicriteria problem in polynomial time, by using the normal shortest path algorithm.

## 2.2 Geometric Notation and Review

We briefly review some standard definitions and results, and refer the reader to texts on computational geometry [PS85,Ede87] for further reading on geometric algorithms.

We will usually work in the Euclidean plane, $E^2$, the set of ordered pairs of reals with the Euclidean distance metric. If the two points $u$ and $v$ have $(x,y)$-coordinates $(u_x, u_y)$ and $(v_x, v_y)$ the Euclidean distance between them is

$$((|v_x - u_x|)^2 + (|v_y - u_y|)^2)^{1/2}.$$

A d-dimensional Euclidean space, $E^d$, is the set of $d$-tuples of $d$ real numbers, $(x_1, \cdots, x_d)$, with the Euclidean metric.

Sometimes we will use $L_p$ distance between two points instead of Euclidean distance. The $L_p$ distance between two points $u$ and $v$ is

$$((|v_x - u_x|)^p + (|v_y - u_y|)^p)^{1/p}.$$

The Euclidean distance is also called the $L_2$ distance. Given two points $u$ and $v$ we define the *line* between them as a linear combination $\alpha u + (1 - \alpha)v$ for all real

$\alpha$. A *line segment* between two points is the convex combination of the two, where $\alpha$ is restricted to be between a real number between 0 and 1, i.e. $0 \leq \alpha \leq 1$.

In the plane, a *polygon* of $n$ vertices is an ordered set of points $(v_1, v_2, \cdots, v_n)$ and the $n$ line segments between every pair of points $(v_i, v_{i+1})$ for $1 \leq i < n$ and the pair $(v_n, v_1)$.

A polygon is called *simple* if two edges that do not share a vertex do not intersect. A simple polygon partitions the plane into two distinct bounded and unbounded regions. Often when referring to a polygon $P$ we will mean the union of its outer boundary and the bounded region in its interior.

A graph $G = (V, E)$ can be embedded in the plane by mapping the vertices to distinct points in the plane and drawing straight line segments between the points if the corresponding edge between the vertices exists. A graph that can be embedded in the plane so that no two segments cross is called planar. A planar subdivision with all triangular regions is called a triangulation. A simple polygon with $n$ vertices can be triangulated in linear time [Cha91]. A simple polygon with simple polygonal holes and a total of $n$ vertices can be triangulated in $O(n \log n)$ time [PS85]. We will usually assume that a triangulation of a polygon or a polygon with holes has been performed as a preprocessing step.

We say that two points, $u, v \in P$, are *mutually visible* if the line segment $\overline{uv}$ lies entirely in $P$. The visibility graph of $P$, denoted $VG(P)$, is the graph on the vertex set of $P$ whose edges join mutually visible pairs of vertices. $VG(P)$ can be computed in time $O(E + n \log n)$, where $E$ denotes the number of edges in the graph [GM91].

Given a visibility graph edge $\overline{ab}$, the *extension* of $\overline{ab}$ refers to the connected component of $\ell \cap P$ that contains $\overline{ab}$, where $\ell$ is the line passing through $a$ and $b$. We may sometimes need to compute the *arrangement* of these visibility graph extensions.

The arrangement of a set of line segments in the plane is a description of segments and vertices that are formed by intersecting the line segments. Chazelle and Edelsbrunner [CE92] give an algorithm to construct the arrangement of $n$ segments in time $O(n \log n + I)$, where $I$ is the total number of intersections.

Let $P$ be a polygon (possibly with polygonal holes) in the plane, with a total of $n$ vertices. For some problems, we will assume that the coordinates of the vertices of $P$ are positive integers. In those cases we will let $N$ denote the largest $x$- or

$y$-coordinate among the vertices.

Given a polygonal path $\pi$, an interior edge $e \in \pi$ is called an *inflection edge* if the predecessor and the successor edges of $e$ in $\pi$ lie on opposite sides of the line containing $e$. An edge $e$ is said to be (rotationally) *pinned* if it passes through two vertices $u$ and $v$ of $P$ in such a way that $e$ is locally tangent at $u$ and $v$ on opposite sides of $e$.

It is well-known (e.g., [LP84]) that, for any two points $s$ and $t$ in a *simple* polygon $P$, the (Euclidean) shortest path, also called the "geodesic" path, $\pi_G(s,t)$ is unique, lying in $VG(P)$ and all of its edges, except perhaps for the first and the last edge, lie in $VG(P)$. If $P$ has holes, then, the geodesic path $\pi_G(s,t)$ is also unique, *except* in certain degenerate circumstances. For a triangulated simple polygon, linear time algorithms are known that find a shortest path tree from one polygon vertex to all other vertices [GHL+87].

In a polygon with holes one can use the visibility graph described above to find the shortest path in $O(E + n \log n)$ time. Alternatively one can compute a *shortest path map* from a source point $s$ to every vertex.

Given two points $s$ and $t$ in $P$, a *minimum-link path* between them is a polygonal path inside $P$ with a minimum number of edges (or, "links"). Figure 2.3 shows an example. The *link distance* $d_L(s,t)$ refers to the number of links in a minimum-link path from $s$ to $t$.

A minimum-link path is generally not unique, even in a simple polygon. In a simple polygon $P$, we define uniquely one particular minimum-link path, which we call the *greedy* path, $\pi_L(s,t)$, from $s$ to $t$. Specifically, the path $\pi_L(s,t)$ is obtained by using the "windows" (and their extensions) defined by the *window partition* of $P$ with respect to $s$, where the last link is chosen to pass through the last vertex of the geodesic path $\pi_G(s,t)$.

The following notion of a *window partition* was introduced by Suri [Sur90]. Given a point or a line segment $s$, let $WP(s)$ denote the subdivision of the polygon $P$ into maximal connected regions with the same link distance from $s$. We call $WP(s)$ the *window partition* of $P$ with respect to $s$. A window partition of $P$ can be computed in $O(n)$ time. $WP(s)$ has an associated set of *windows*, which are the chords of the polygon that serve as the boundaries between adjacent regions of the partition. We can preprocess a window partition $WP(s)$, in additional $O(n)$ time, for point location queries, after which a link distance query from the fixed

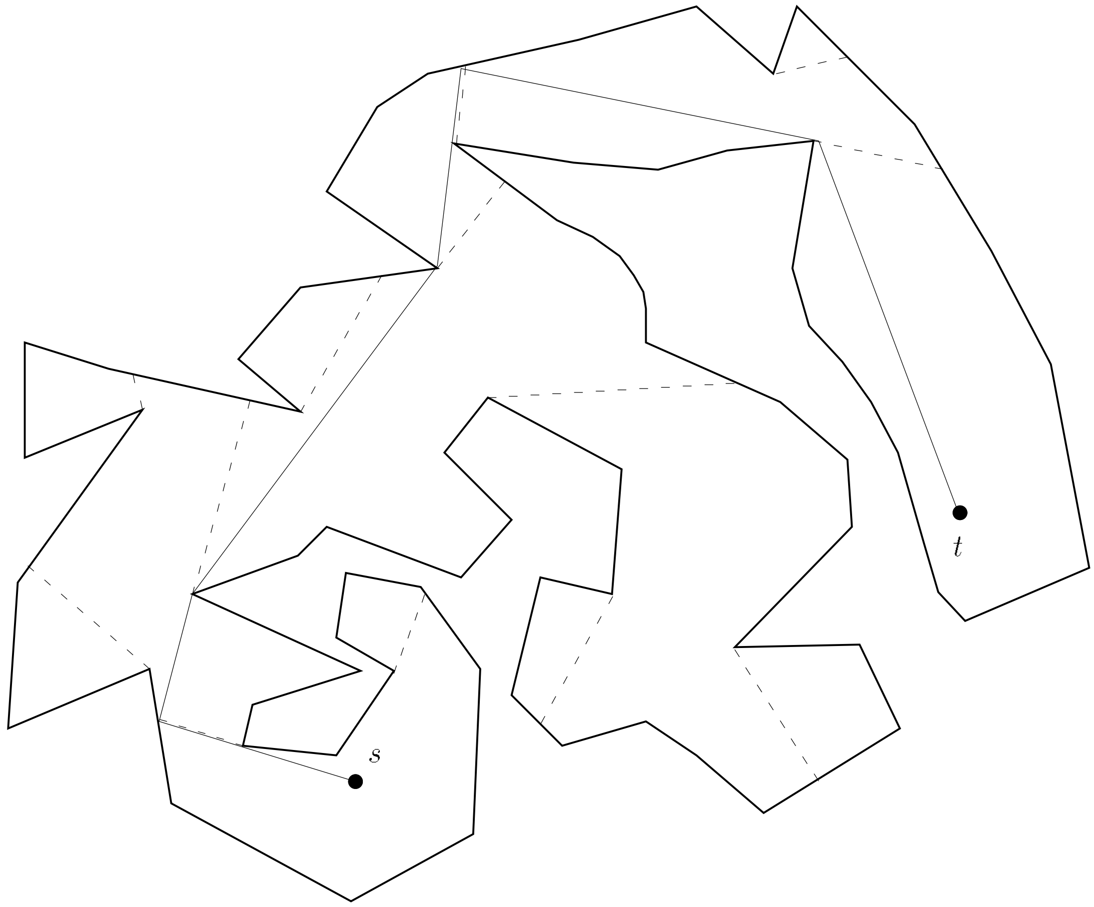



Figure 2.3: A window partition $WP(s)$ and a corresponding minimum-link path.

source $s$ can be answered in time $O(\log n)$. A minimum-link path from $s$ to the query point $t$ can be traced in additional constant time per link. An example of a window partition is given in Figure 2.3.

Aggarwal, Booth, O'Rourke, Suri and Yap [ABO+89] presented an algorithm to find the minimum-link enclosing polygon contained between a convex outer and convex inner polygon in $O(n \log k)$ time. Ghosh [Gho91] generalized this result to find the minimum-link polygon between any outer polygon and any convex inner polygon, using results on convex visibility.

Given an inner convex polygon $C$ and a general outer polygon $P$, Ghosh defined the *complete visibility polygon* (sometimes called the convex visibility polygon) of

$P$ from $C$ as the set (containing $C$) of all points in $P$ that are mutually visible from $C$. (In this setting, two points are visible if the line segment joining them lies within $P$, where $C$ is not considered to be an obstacle.) This complete visibility polygon can be found in $O(n)$ time. Ghosh also gave a linear time algorithm to compute the minimum-link path between $s$ and $t$ using these results on convex visibility.

Throughout this work, the the start and goal points $s, t \in P$ will remain fixed. We assume that $s$ and $t$ are in general position, so that $\pi_G(s,t)$ is uniquely defined. When using link distance we will often talk about finding $k$ link paths. We will assume that we have fixed the number $k$ to be some integer satisfying $d_L(s,t) \leq k \leq |\pi_G(s,t)|$, where $|\pi_G(s,t)|$ denotes the number of edges in the geodesic path $\pi_G(s,t)$. (Clearly, integers $k$ outside of the above range are of no interest.)

## 2.3 Geometric Lemmas

In this section we state some useful geometric properties that will be used in later chapters.

### 2.3.1 Fixed Orientation Approximations

Some of our approximation algorithms will use the fact that the Euclidean metric can be approximated by a fixed-orientation metric. In a fixed orientation metric a set of $k$ orientations is given. The distance between two points is measured as the length of the shortest path between them where the segments of the path are restricted to follow the fixed orientations. For example, the $L_1$ or rectilinear metric, is a fixed orientation metric based on the two directions 0 and $\pi/2$. By taking a large enough set of equally spaced directions, the fixed orientation metric can be used to closely approximate the Euclidean metric. We state the following lemma, sketch a proof and refer the reader to Yao [Yao82], and Clarkson [Cla87] for further details.

**Lemma 2.5** *The Euclidean distance between two points in the plane can be approximated to accuracy $\epsilon$ by a fixed-orientation metric of $k = O(1/\sqrt{\epsilon})$ equally-spaced directions.*

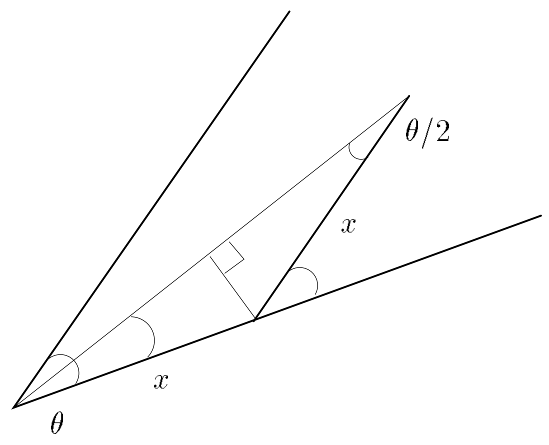



Figure 2.4: Error ratio for a restricted orientation metric.

**Proof.** Let $a$ and $b$ be two points. We want to approximate the distance between the two points by a fixed orientation metric of $k$ equally spaced directions. We would like to know how many directions to use so that distance in this fixed orientation metric will be close to the true answer. That is, if we want the approximate distance in the fixed metric to be $1+\epsilon$ times the true Euclidean distance, how should we choose $k$ based on $\epsilon$?

Let us consider the error ratio between the approximate distance and the true Euclidean distance $r$ between points $a$ and $b$. We move the origin of our $k$ equally spaced directions to $a$ and find the cone formed by two of the directions that contains $b$. Suppose that cone has angle $\theta$. The worst case error between the true distance and the approximate distance occurs when $b$ lies on the bisector of the cone. (See Figure 2.4.) In this case the approximate distance is $2x$, where $\cos(\theta/2) = (r/2)/x$. The error ratio of the approximation to the true distance is $2x/r$. Thus we know the error ratio is bounded by $1/(\cos\theta/2)$. Using a Taylor expansion we can bound this approximately by

$$1/(1-(\theta/2)^2+\cdots) = 1+(\theta/2)^2 + \text{a small error term.}$$

If we use $k$ equally spaced directions then we have $\theta = 2\pi/k$. Thus, if we want the Taylor expansion above to be $(1+\epsilon)$, we should use $O(1/\sqrt{\epsilon})$ equally spaced directions. $\square$

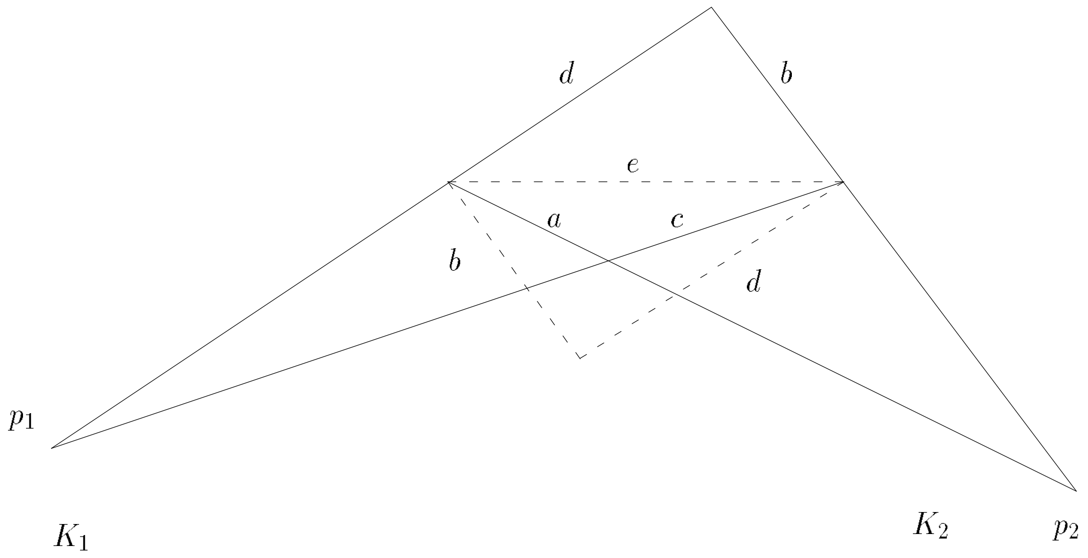



Figure 2.5: Intersection of two cones: $(a + c) < (d + b)$.

### 2.3.2 An Observation about Cones

We will also make use of the following geometric observation about the intersection between two cones. Let $K_1$ and $K_2$ be cones between a left ray and a right ray (with origins $p_1$ and $p_2$ respectively) such that the left and right rays of the cones are less than 180 degrees apart. Furthermore, let $p_2 \neq p_1$, let $p_2$ be to the right of the right ray of $K_1$, and let the intersection of $K_1$ and $K_2$ be a nonempty bounded region. Let $a$ be the intersection of the left ray of $K_2$ with $K_1$ and $b$ be the intersection of the right ray with $K_1$. Similarly, let $d$ be intersection of the left ray of $K_1$ with $K_2$ and $c$ be the intersection of the right ray with $K_1$. (See the solid lines Figure 2.5.) Then the following Lemma holds:

**Lemma 2.6** *Let $K_1$ and $K_2$ be cones with intersections $a$, $b$, $c$ and $d$ as described above. Then $(a + c) < (d + b)$.*

**Proof.** This observation can be shown in the following manner. Translate a copy of $b$ towards along the left ray of $K_1$, towards the origin $p_1$. Similarly, translate a copy of $d$ towards along the right ray of $K_2$, towards the origin $p_2$. (See the dashed lines in Figure 2.5.) Since $b$ will extend beyond the right ray of $K_1$ and $d$ will extend beyond the left ray of $K_2$ it is clear that the triangle formed by $\overline{bde}$

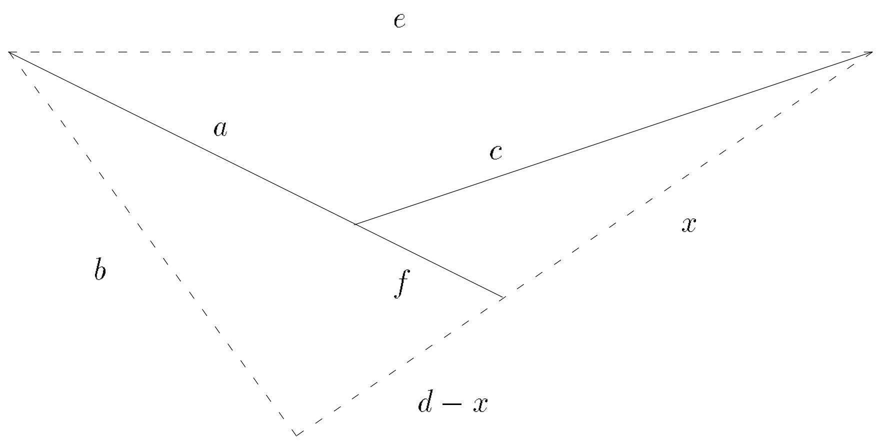



Figure 2.6: Why $(a+c) < (d+b)$.

will contain the triangle $\overline{ace}$.

We can then use this to show that $(a+c) < (d+b)$. Let $f$ be the extension of $a$ until it hits $d$, and let $x$ be length of the side. See Figure 2.6. Then, by using the triangle inequality, we observe that $(b+d-x) > (a+f)$ and $(f+x) > c$. Combining these we get the desired inequality, since

$$(b+d) = b + (d-x) + x > (a+f) + x > (a+c).$$ □

### 2.3.3 Geometric Properties on a Grid

We will also use the following fact about polygons whose vertices come from an $N$-by-$N$ integer grid.

**Lemma 2.7** *The smallest angle formed by 3 vertices coming from an $N$-by-$N$ integer grid has size at least $c/N^2$. Also, the largest angle formed by 3 vertices is at most $\pi - c/N^2$.*

**Proof.** The smallest angle in the grid is in the triangle formed by the upper two left points and the last point of the second row (refer to Figure 2.7). The edges of this triangle have length 1, $\sqrt{1+N^2}$ and $\sqrt{1+(N-1)^2}$. Note that

$$\sin\phi = 1/\sqrt{N^2+1}.$$

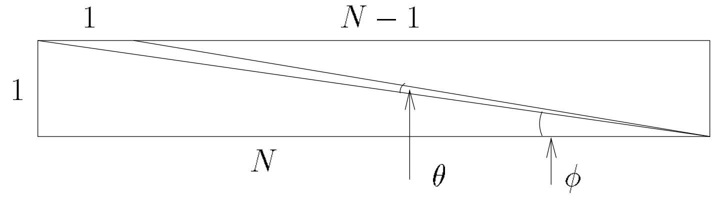



Figure 2.7: How small can an angle from a grid be?

Using the law of sines we find that $\sin\phi/\sqrt{1+(N-1)^2} = \sin\theta$. Thus,

$$\sin\theta = 1/\sqrt{(N^2+1)(N^2-2N+2)} = 1/\sqrt{N^4-2N^3+3N^2-2N+2}.$$

Call the denominator of this expression $w(N)$. We know that

$$\cos\theta = \sqrt{1-(1/w(N))^2}.$$

Using the Maclaurin series for this we can estimate $\sqrt{1+x}$ as $1 + x/2 + \epsilon(N)$, where $\epsilon(N)$ is a small error term. Thus we can estimate $\cos\theta = 1 - 1/2N^4 + \epsilon(N)$. By a Taylor expansion for $\cos\theta$ we know that $\cos\theta = 1 - \theta^2/2 + \epsilon(\theta)$, for a small error term $\epsilon(\theta)$. Thus we know that $\theta$ is $\Omega(1/N^2)$. $\square$

**Corollary 2.8** *The sine of the smallest angle formed by 3 vertices coming from an $N$-by-$N$ integer grid is at least $c/N^2$. Also, the largest sine of any angle formed by 3 vertices is at most $\pi - c/N^2$.*

**Corollary 2.9** *The cosine of smallest angle formed by 3 vertices coming from an $N$ by $N$ integer grid has size at most 1- $c/N^4$. Also, the cosine of the largest angle formed by 3 vertices is at least $1/N$.*

**Corollary 2.10** *Let $v$ be any point within an $N \times N$ grid (not necessarily a grid point). Then, a cone with vertex $v$ and angle less than $\theta_{\min}$ cannot contain non-collinear grid points.*

# Chapter 3

# Bicriteria Length Problems[1]

In this chapter we begin our study of various *bicriteria* path problems in a geometric setting. We consider two pairs of criteria for planar paths: path length measured according to two different norms ($L_p$ and $L_q$), and path length within two or more classes of regions. Unfortunately, as is the case for the general bicriteria path problem on graphs, these problems are NP-complete. In later chapters we will show some more positive results when either total turn or link distance are one of the two criteria.

## 3.1 Minimizing Both $L_p$ and $L_q$ Length

Suppose we would like to simultaneously minimize the $L_p$ and $L_q$ lengths of a path from $s$ to $t$ for some $p \neq q$. This is a somewhat artificial problem, but it will serve to illustrate how we can use a reduction from Partition is NP-hard. Also, this result is not as surprising as it may seem at first glance. It highlights what seems to be a fundamental difference between computing the shortest path using one norm or another. Consider that the best known shortest path algorithm between 2 points among obstacles in the plane is $O(n \log n)$ [Mit92], but that the current similar bound for the Euclidean metric was only very recently shown to be subquadratic [Mit93]. Clearly, a path which is short for one metric may not be the best path in another metric, and this result highlights this fact.

Here we give a proof for the $L_1$ and $L_2$ norms. This proof can be generalized to any two $L_p$ and $L_q$ norms (where $p \neq q$). It can also be generalized to two convex

[1] Parts of this chapter represent joint work with Esther Arkin and Joseph S. B. Mitchell.

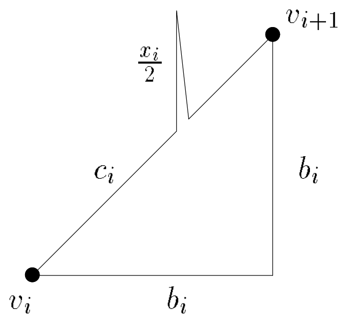



Figure 3.1: Corridors for $L_1$ and $L_2$ length.

distance functions that are not similar under scaling. Below we state the problem where the lengths are bounded just for the $L_1$ and $L_2$ norms:

**Dual Length Minimization Problem**
*Instance:* A simple polygon $P$, possibly with holes, a source $s$ and a destination $t$, a bound $A$ and a bound $B$.
*Question:* Does there exist a path from $s$ to $t$ whose $L_1$ length is $\leq A$ and whose $L_2$ length is $\leq B$?

**Theorem 3.1** *The Dual Length Minimization Problem is NP-complete.*

**Proof.** We use a reduction from Partition, similar to the one for the Bicriteria Path Problem in a Graph. First, we make a gadget that corresponds to the nodes $v_i$ and $v_{i+1}$ and the 2 edges between them (Figure 3.1). We start with an isosceles right triangle with base $b_i$, height $b_i$ and hypotenuse $c_i$. We add a skinny vertical "hump" of total length $x_i$ to the hypotenuse. (Note that the lengths we refer to here will be off by a small amount. By adding the vertical "hump" we take a small amount away from the length of $c_i$, and the hump cannot be perfectly vertical. However, such differences can be made small enough so that they do not affect the structure of the proof.) The exact values of $b_i$ and $x_i$ will be chosen later. The $L_2$ length of $c_i$ is $\sqrt{2}\,b_i$, and the $L_1$ length of $c_i$ is $2b_i$. There will be only 2 paths from $v_i$ to $v_{i+1}$. The upper path, following the hypotenuse and the hump, has $L_2$ length $x_i + \sqrt{2}\,b_i$, and $L_1$ length $x_i + 2b_i$. The lower path has $L_2$ and $L_1$ length $2b_i$. We choose $x_i$ so that the upper path is longer than the lower path by $a_i$ (the value of the $i$th item in the Partition problem) in the $L_1$ norm and shorter by $a_i$ in the $L_2$ norm. We want $x_i = (2b_i + a_i) - 2b_i$, that is, $x_i = a_i$. We also want $\sqrt{2}\, b_i + a_i = 2b_i - a_i$, which implies we should choose $b_i = (2 + \sqrt{2})\, a_i$. We connect $n$ of these gadgets along a diagonal line, and take as obstacles the complements of the paths drawn. □

For correctness we must note that there is some difficulty with choosing the horizontal and vertical lengths $b_i$ in this way: we cannot generate irrational coordinates in polynomial time. However, since there are rational points arbitrarily close to irrational points, we can choose rational coordinates in polynomial time such that the constructed length $b_i$ differs by at most $\epsilon$ from the desired length. Since we have $n$ such gadgets we can choose an $\epsilon$ such that $n\epsilon$ is much smaller than 1. In particular, if there are $n$ items we can choose rational approximations to the desired points such that each difference between a desired point and its rational approximation is bounded by $1/(2^n)$ since we can use a number of bits polynomial in $n$. Thus $n\epsilon$ will be much smaller than 1. Since the weights $a_i$ are all integers, the sum of these small differences in length does not add up to be an integer, this approximation will not affect the optimal solution.

## 3.2 Bichromatic Bicriteria Path Problem

Here we consider a variation of the Weighted Region Problem. In the Weighted Region Problem the plane is partitioned into disjoint regions, each with an associated weight. The length of a path in a region is the Euclidean length multiplied by the factor for that region. The total length of a path is the sum of the lengths of the pieces of the path in each separate region. This problem was solved in Mitchell's thesis work by using an analysis of local optimality conditions [MP91]. A locally optimal path that crosses from one region to another will obey a reflective law like Snell's law.

Here we would like to consider a version of the problem where the amount of travel in each region is limited, rather than providing a bound on the total path length. This might correspond to a simplified version of an "exposure" problem. The robot or agent is not willing to risk too much total exposure in either type of region.

Suppose the plane is partitioned into red and blue regions. We can ask for the path from $s$ to $t$ that minimizes travel in *both* the red and blue regions.

For any $L_p$ metric this problem is also NP-hard. To prove this we first show a special version of the Knapsack problem is NP-complete. The reduction for multiple regions will mimic this proof.

In Constrained Fractional Knapsack we are given a set $N$ of items, each with a value $v_i$ and weight $w_i$, and bounds $V$ and $W$. A solution to Constrained Fractional Knapsack, will be a set $S \subset N$ of whole items and a set $F \subset N$ of fractional items, with $S \cap F = \emptyset$, such that the knapsack has value $\geq V$ and weight $\leq W$. Let $f_i$ be the fraction of item $i$ taken, i.e. $0 \leq f_i \leq 1$. The value of a knapsack is the sum of the value of whole items taken, plus the fractional *weight* of fractional items taken, i.e. $\sum_{i \in S} v_i + \sum_{i \in F} f_i \cdot w_i$. Alternatively, we can think of the value of a knapsack as the value of items completely taken plus any remaining capacity, i.e., $\sum_{i \in S} v_i + (W - \sum_{i \in S} w_i)$. The weight of the knapsack is just the weight of whole items plus the appropriate fraction of the weight of fractional items, i.e. $\sum_{i \in S} w_i + \sum_{i \in F} f_i \cdot w_i$. We state the problem formally below:

**Constrained Fractional Knapsack**
*Instance:* A set $N$ of $n$ items, each item $i$ for $1 \leq i \leq n$ having a value $v_i$ and a weight $w_i$, and bounds $V$ and $W$.
*Question:* Do there exist $S \subset N$ and $F \subset N$ (where $S \cap F = \emptyset$), and a set of fractions $f_i$, $0 \leq f_i \leq 1$ for each $i \in F$, such that the value of $S \cup F$, computed as $\sum_{i \in S} v_i + \sum_{i \in F} f_i \cdot w_i$, is $\geq V$, and the weight of $S \cup F$, computed as $\sum_{i \in S} w_i + \sum_{i \in F} f_i \cdot w_i$, is $\leq W$?

**Theorem 3.2** *The Constrained Fractional Knapsack problem is NP-complete.*

**Proof.** We use a reduction from Partition. Let $W = \frac{1}{2} \sum_{i \in N} a_i$. For item $i$, let $w_i = a_i$ and let $v_i = 2(W+1) \cdot a_i$. Let $V = 2(W+1) \cdot \frac{1}{2} \sum_{i \in N} a_i$. Suppose we are given a solution to Constrained Fractional Knapsack. We know the weight of the knapsack is $\leq W$, i.e. $\sum_{i \in S} a_i + \sum_{i \in F} f_i \cdot a_i \leq W = \frac{1}{2} \sum_{i \in N} a_i$. The value of the knapsack is $\geq V$, i.e.

$$\sum_{i \in S} 2(W+1) \cdot a_i + \sum_{i \in F} f_i \cdot a_i \geq (W+1) \sum_{i \in N} a_i = V.$$

Since $\sum_{i\in F} f_i \cdot a_i \leq W$ we can subtract $\sum_{i\in F} f_i \cdot a_i$ from the left and $W$ from the right to get

$$2\,(W+1)\sum_{i\in S} a_i \geq (W+1)\sum_{i\in N} a_i - W$$
$$\Longrightarrow \sum_{i\in S} a_i \geq \frac{1}{2}\sum_{i\in N} a_i - \frac{W}{2\,(W+1)}.$$

We know that $\frac{W}{2\,(W+1)}$ is a fraction, and in particular it is less than 1/2. The sum on left hand side of the equation is an integer. The sum on the right hand side of the equation is either an integer or an integer plus 1/2. In either case, since the fraction $\frac{W}{2\,(W+1)}$ is small enough, we know that $\sum_{i\in S} a_i \geq \frac{1}{2}\sum_{i\in N} a_i$. We already know the weight of the knapsack implies that $\sum_{i\in S} a_i \leq \frac{1}{2}\sum_{i\in N} a_i$. Thus, the Constrained Fractional Knapsack solution solves Partition. □

We can now use a similar proof technique to show that minimizing travel through two regions simultaneously (using the $L_1$ norm) is NP-complete. This proof can be modified to extend to any $L_p$ norm.

**Bichromatic Bicriteria Path Problem**

*Instance:* A simple polygon $P$ with holes, where the holes are colored red, and the interior of the polygon is colored blue, a source $s$, a destination $t$, a bound $R$ and a bound $B$.

*Question:* Does there exist a path from $s$ to $t$ whose $L_1$ length in red is $\leq R$ and whose $L_1$ length in blue is $\leq B$?

**Theorem 3.3** *The Bichromatic Bicriteria Path Problem is NP-complete.*

**Proof.**

We use a reduction from Partition, based on the Constrained Fractional Knapsack reduction, where the $v_i$'s, $w_i$'s, $V$ and $W$ are chosen the same way as above (note that the $v_i$'s are larger than all $w_i$'s). We create "tunnels" between $s$ and $t$ such that the length of travel in the blue region for the $i$th item is at least $c - w_i$ (where $c$ is chosen so $c - w_i > W$ for all $i$). We then create a red barrier such that travelling in red would cost $w_i$, corresponding to choosing the item, but going around the red barrier through a blue tunnel would cost $(v_i - w_i)/2 + w_i + (v_i - w_i)/2$, i.e. $v_i$, corresponding to leaving the item (see Figure 3.2).

We can then think of the value of choosing an item as the amount of "savings" if we shortcut through the red region instead of travelling through blue. For now,

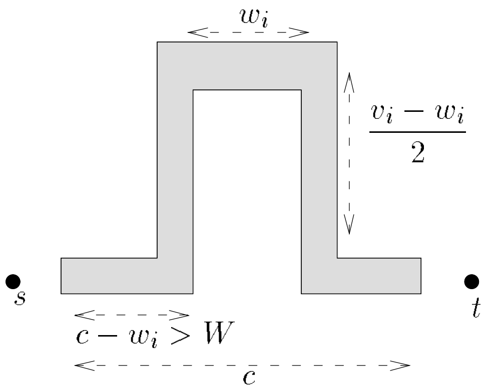



Figure 3.2: Blue tunnel thru red region for the $i$th item.

assume that if an item is not chosen, the entire blue path is followed. Thus, the length of the path in blue is $c \cdot |N| - \sum_{i \in N} w_i + \sum_{i \notin S} v_i$. We choose $R = W$ and $B = c \cdot |N| - \sum_{i \in N} w_i + \sum_{i \in N} v_i - V$. Thus, if the length in blue is $\leq B$, $\sum_{i \notin S} v_i \leq \sum_{i \in N} v_i - V \Longrightarrow \sum_{i \in S} v_i \geq V$.

However, the path can "cut corners" through the red region, i.e. it can cut corners of the path to allow a little red to be chosen. This corresponds to choosing a fractional item. If an item is partially selected its value will be its length in red (which corresponds to its weight in the Constrained Fractional Knapsack problem). By connecting $n$ of these blue tunnels together, the same proof technique used for Constrained Fractional Knapsack can be used to prove We can modify this proof to work for any $L_p$ metric, by using a variation of Constrained Fractional Knapsack in which fractional items contribute an appropriate constant times their fractional weight as value to the knapsack. For example, using the $L_2$ norm, if a shortcut goes across the red region at a $45°$ angle then the weight of the item is $\sqrt{2}$, while the savings in blue, or its value, is 2. Thus, the value constant to multiply the weight for the $L_2$ norm case is $\sqrt{2}$. (Of course this constant must also be included in the total value $V$.) In general, the length of a shortcut in red will be $2^{1/p}$ for a given $L_p$ norm, which will correspond to the item's weight, while the savings from blue, which corresponds to the item' value, will still be 2. Thus the appropriate fraction to multiply times the weight will be $2/(2^{1/p})$ for a given $L_p$ norm.

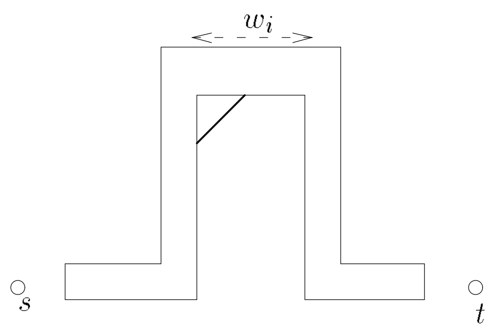



Figure 3.3: Possible shortcuts.

**Lemma 3.4** *A shortcut through the red region will be at a 45° angle to the 90° corner made by the blue region. Also the sum of the lengths of all such shortcuts (over all the red regions) will be less than the largest $w_i$.*

**Proof.** We will prove this lemma for any $L_p$ norm. Suppose we have a shortcut of length $L$ across the red region. We consider $(0,0)$ as the origin of the upper left corner of the blue rectangle, and call the length across the top of the blue region $x$, and the length on the left side of the blue region $y$. We note that $L = (x^p + y^p)^{1/p}$. Since the value of choosing a fractional item is just $x + y$ we want to know when this is maximized. Using the previous equation we know $y = (L^p - x^p)^{1/p}$. Thus we can write $x + y$ as

$$f(x) = x + (L^p - x^p)^{1/p}.$$

Finding

$$df/dx = 1 + (-1/p)(L^p - x^p)^{(1/p)-1}px^{p-1}$$

and setting this to zero we see that this function is maximized when

$$x^{p-1} = (L^p - x^p)/((L^p - x^p)^{p-1}) = y^p/y = y^{p-1}$$

which implies that $y$ must be equal to $x$, and therefore the shortcut must be at a 45° angle.

Finally, we note that if the total length of all the shortcuts added up to be more than the largest $w_i$ then we would be better off taking any one item $l$ that we had formerly left and did not take. The amount of savings for blue would then be

$v_l$. Since we have used the same proof as for Constrained Fractional Knapsack we have chosen every $v_i = 2(W+1) \cdot a_i$. Since the sum of all the small shortcuts can be at most $2W$, $v_l$ is greater than any possible savings by small shortcuts. This must be weighted by the appropriate constant, $2/(2^{1/p})$, for norms other than the $L_1$ norm. $\square$

Again, we note that we need slight $\epsilon$ perturbations to construct the blue corridors which will each have some small width. However, again these slight perturbations will add up to $n\epsilon$ and we can choose the $\epsilon$ so the sum over all the perturbations is not an integer and so these differences will not affect the final answer.

# Chapter 4

# Turn with Length or Links[1]

Suppose we prefer to drive longer on a flat stretch of highway than to travel a shorter distance along winding mountain roads. We would like to consider a path to be short when it does not spiral or turn too much, as well as when it is short in the usual sense of Euclidean length. A path with a large amount of total turn may represent a path that would be difficult for a robot to execute because of the acceleration changes that may be required.

Consider a polygon $P$ with holes. One measure of how much a path in $P$ is "kinked" is the "total turn" (TT) of the path, defined to be the sum of absolute values of changes in orientations of a piecewise linear path. We can think of travelling along a segment in a particular direction, then at a vertex of the path we change orientation by an amount $\theta$ to line up with the next segment of the path. The total turn over the path is the sum of all these $\theta$ values. We can also think of total turn as measuring the total motion of the steering wheel when we drive a car along the path.

In this chapter we will consider total turn as one of our criteria. First we will address total turn in combination with length and then we will consider total turn in combination with the number of links in the path. We leave open the problems of maintaining a bound on the amount of turn per link, or maintaining a bound on the length of each link.

[1]Parts of this chapter represent joint work with Joseph S.B. Mitchell and Esther Arkin.

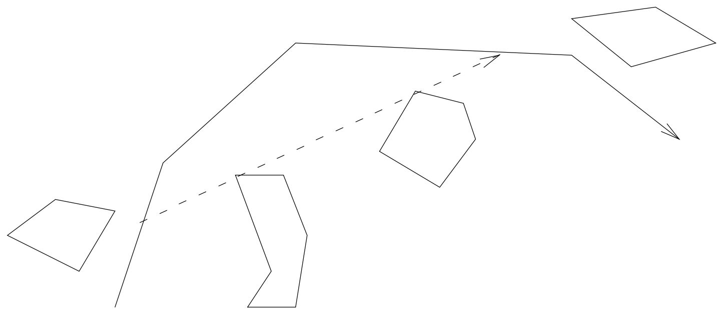


Figure 4.1: Pareto-optimal TT paths must use VG edges.

## 4.1 Total Turn and Length

One version of the geometric bicriteria path problem in the plane asks us to find a piecewise linear path from $s$ to $t$ that minimizes the length and total turn of the path. Here, the *total turn* of a path is the sum of the absolute values of changes in $\theta$ over the path. This simple definition of the total turn problem does not address all the issues in finding a path that is not too kinked, but as we shall see even this problem is hard. The formal problem is stated below:

**The Total Turn and Length Problem**
*Instance:* A simple polygon $P$, possibly with holes, a source $s$, a destination $t$, a bound $L$ for Euclidean length, and a bound $\theta$ for total amount of turn.
*Question:* Does there exist a piecewise linear path from $s$ to $t$ whose Euclidean length is $\leq L$ and whose total turn is $\leq \theta$?

**Lemma 4.1** *Any pareto-optimal path for total turn and length must lie on visibility graph edges.*

**Proof.** Recall that a *pareto-optimal* path is one that cannot be improved in one of the two criteria without increasing the other criteria, i.e. we cannot find a path with shorter length that has a smaller amount of turn, and vice versa. If a path does not follow visibility graph edges, we can shortcut along a chord of the path that passes through two vertices, improving both the length and the total turn of the path. (See Figure 4.1.) We can continue this process, using shortcuts until every edge of the path lies on a visibility graph edge. $\square$

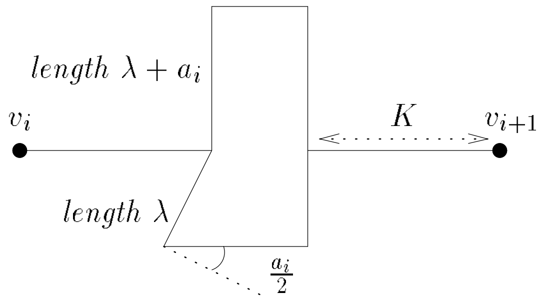



Figure 4.2: Gadget for total turn and length.

**Corollary 4.2**
*In a simple polygon the (unique) shortest path is the only pareto-optimal path.*

For polygons with holes, however, we have the following result:

**Theorem 4.3** *The Total Turn and Length Problem is NP-hard.*

**Proof.** The proof is based on the graph construction used for bicriteria paths in graphs, with constants added to the edges so there are no zero lengths or weights (see Theorem 2.1). We again use a reduction from Partition, scaled so that each $a_i$ is less than $\pi$ (and thus may not be integer). We construct a planar graph with $n+1$ nodes, with $s = v_1$ and $t = v_{n+1}$. We draw two "edges" from each $v_i$ to $v_{i+1}$, one with length $\lambda + a_i$ and total turn $2\pi$ and one with length $\lambda$ and total turn $2\pi + a_i$, where $\lambda$ is a constant. The claim is that the first edge can be drawn in the plane with three bends, with length $\lambda + a_i$ and turn $2\pi$, and the second edge can be drawn with 3 edges, with total length $\lambda$ and total turn $2\pi + a_i$ (see Figure 4.2). We draw a corridor of constant length $K$ (where $K$ is bigger than any $\lambda + a_i$) so that consecutive gadgets will not overlap.

Our obstacles will be the complement of the edges we have drawn. Thus, a partition will exist if and only there exists a path with total turn $\leq 2\pi n + \frac{1}{2}\sum_{i=1}^{n} a_i$ and length $\leq n(\lambda + K) + \frac{1}{2}\sum_{i=1}^{n} a_i$. □

For this problem, it is known that the number of pareto-optimal paths can be exponential [Sta89]. However, the problem is not strongly NP-complete:

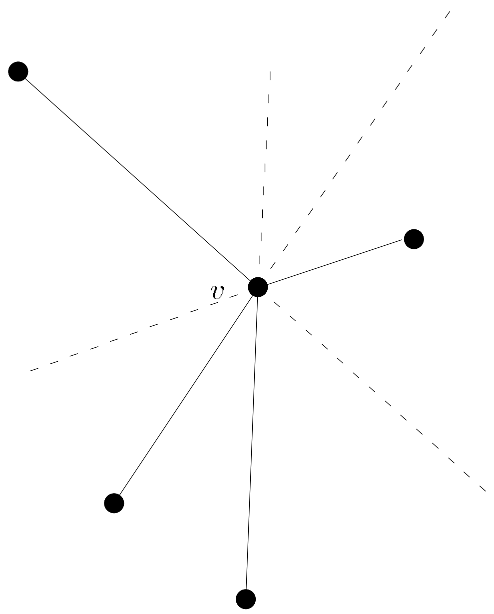



Figure 4.3: Visibility graphs edges and their extensions incident on $v$.

**Theorem 4.4** *There exists a pseudo-polynomial time algorithm for the problem of Total Turn and Length which runs in time $O(En^2N^2)$, when the vertices of the obstacle scene lie on an $N$-by-$N$ grid.*

**Proof.** By Lemma 4.1 above, we know the optimal path must lie on the visibility graph, so we can map visibility graph edges to a graph $\mathcal{G}$. Each visibility graph edge $e$ between $u$ and $v$ will be split into two directed edges. The directed edge from $u$ to $v$ is changed into an edge between the nodes $u_{e\_out}$ and $v_{e\_in}$. Similarly, the edge from $v$ to $u$ becomes an edge between $v_{e\_out}$ and $u_{e\_in}$. Both edges are given length $\|u, v\|$ and weight (corresponding to turn) zero.

We can obtain a data structure such that we know the visibility graph (VG) edges and their extensions in sorted order around the vertex $v$. The visibility graph algorithm of Ghosh and Mount [GM91] gives such a data structure in time $O(E + n \log n)$. For each VG edge $e$ directed into $v$ we find the VG edge $f$ directed out of $v$ that is clockwise to $e$'s extension. We connect $v_{e\_in}$ to $v_{f\_out}$, and give the new edge a weight $\theta$, where $\theta$ is the amount of turn from $e$ to $f$, and length 0. (See Figures 4.3 and 4.4.) Similarly, we connect $v_{e\_in}$ to $v_{g\_out}$ where $g$ is the

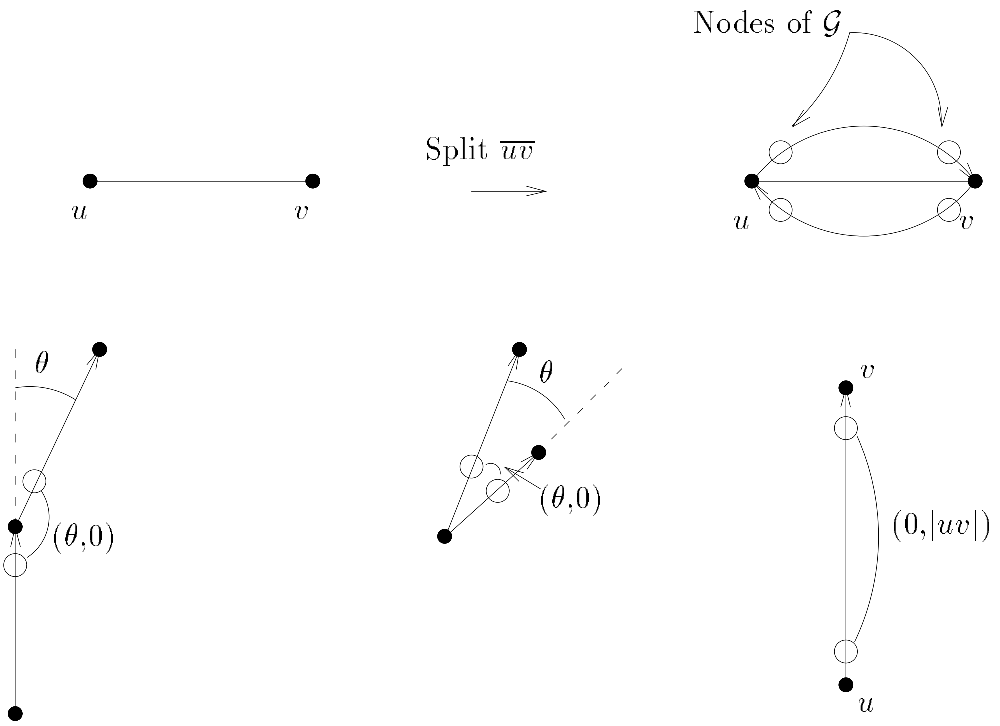



Figure 4.4: Assignment of (length, weight) to (directed) edges of $\mathcal{G}$.

VG edge counterclockwise to $e$'s extension. This graph will have $O(E)$ nodes and $O(E)$ edges.

If the $n$ obstacle vertices have integer coordinates (of maximum size $N$), the smallest angle formed by any pair of VG edges is $\theta_{\min} = c/N^2$, for a constant $c$. (See Lemma 2.7.) We would like to round all of the angles to "integer" values, so we can use the dynamic programming technique to find the optimal solution to our graph problem. Rounding angles to multiples of $\theta_{\min}$ may be too coarse. The optimal solution may use up to $n$ links, so the accumulated error of adding rounded angles may add up to as much as $n\theta_{\min}/2$. Therefore, we refine further. We define $\Delta\theta = \theta_{\min}/(2n)$, and round all weights $\theta$ to integer multiples of $\Delta\theta$. Since no paths of interest have more than $n$ turns, measuring angles to within the resolution $\Delta\theta$ is sufficient for solving the bicriteria path problem. By replacing an edge whose weight is approximately $k \cdot \Delta\theta$ by an edge with the integer weight $k$, we will still have a graph with $O(E)$ edges and $O(E)$ nodes. The total amount of turn a path can take is bounded by $O(2\pi n)$. By the dynamic programming technique in Lemma 2.2 we can solve this bicriteria path problem with integer weights in time $O(En^2N^2)$. □

## 4.2 Minimum Total-Turn $k$-Link Paths

In contrast to the previous negative results described for bicriteria paths, we give a polynomial time algorithm to find paths minimizing number of turns and total turn.

Different $k$-link paths may have different amounts of total turn. In this section we ask if there is a $k$-link path with a given total bound on the amount of turn. While this does not provide a local guarantee on the shape of the path, in some sense the overall path cannot spiral or make too many sharp zigzag turns. This problem is more tractable than previous bicriteria problems we have considered because we can prove structural and combinatorial lemmas about optimal paths. We can use this information to find locally optimal paths that can be combined into a globally optimal solution. We state the problem formally below:

**The Total Turn and Links Problem**
*Instance:* A simple polygon $P$, possibly with holes, a source $s$, a destination $t$, bound $\theta$ on total turn, and a bound $k$ on the number of links.
*Question:* Does there exist a piecewise linear path from $s$ to $t$ with total turn $\leq \theta$ and with $\leq k$ links?

Note that unlike the previous problem for total turn and length, the optimal path for turn and links might not use visibility graph edges. In fact there may be an exponential number of homotopically distinct, locally optimal paths from $s$ to $t$. (See Figure 4.5.)

We will prove some structural lemmas about Total Turn and Links paths and use them to decompose the problem and find the optimal solution.

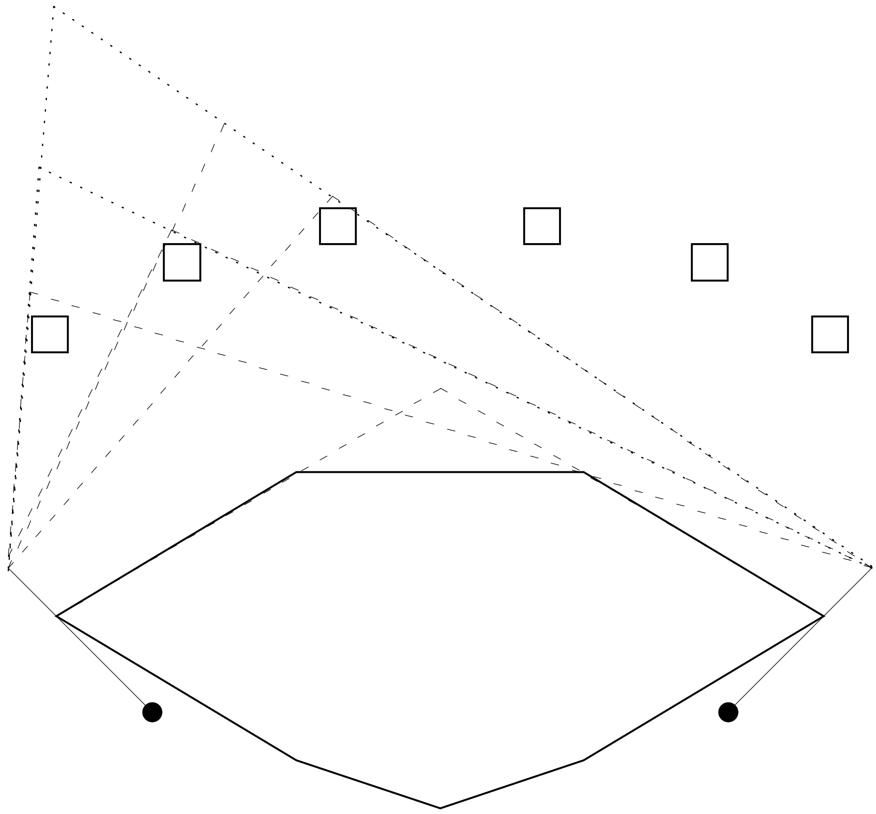


Figure 4.5: Many threadings of 4-link paths with the same total turn.

### 4.2.1 Decomposing the Problem

First, we prove that a path $\pi^*$ which is optimal for turn and links must contain the *inflection edges* that lie along the geodesic path homotopically equivalent to $\pi^*$. Recall that inflection edges are visibility graph edges with obstacles on opposite sides.

**Lemma 4.5** *The inflection edges in a minimum total turn and link path from $s$ to $t$ must lie along the extension of visibility graph inflection edges.*

**Proof.** Suppose there is an inflection edge $e$ in the optimal path where the path changes from convex turns to concave turns. (See Figure 4.6.) We can "pull the path taut" and thus obtain the geodesic shortest path for that specific homotopy type. Suppose $e$ does not lie along the extension of a visibility graph inflection

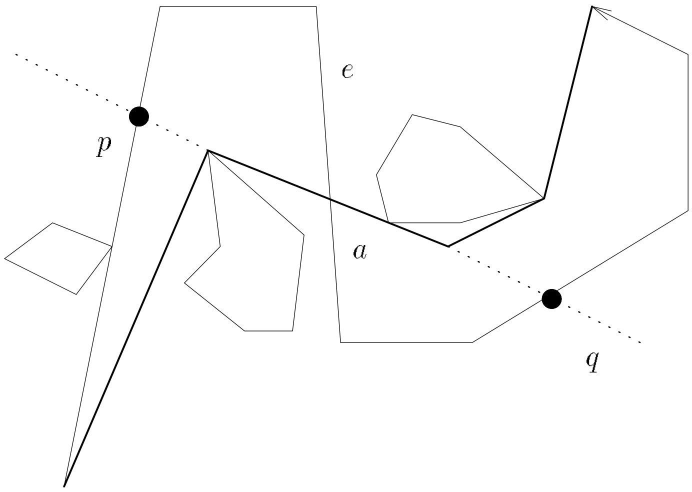



Figure 4.6: Total Turn and Link path $\pi^*$ has VG inflection edges.

edge. Then $e$ must cross some inflection edge in the geodesic path. Call that inflection edge $a$. Let $p$ be the first point that crosses the extension of edge $a$. Let $q$ be the last point of the path that crosses the extension of $a$. By using a shortcut from $p$ to $q$ for the path $\pi^*$ we obtain a path that has the same (or fewer links) and the same (or less) total turn. Thus we may as well have used the extension of visibility graph edge $a$. □

**Lemma 4.6** *A minimum total turn and link path will not cross itself.*

**Proof.** A path that visits a point $p$ twice can be shortcut by skipping the loop the path makes between the first time it visits $p$ and the second time it visits $p$. This shortcut means the path has fewer links and total turn. This process can be repeated until the path eventually has no self-crossings. □

**Corollary 4.7** *A minimum total turn and link path will have a convex or convex-spiralling simple path between extensions of visibility graph inflection edges.*

**Proof.** By Lemma 4.5 we know that any inflection edges can be perturbed to lie along visibility graph edges. The portion of the path between two consecutive inflection edges must be convex, or convex-spiralling, because there is no inflection edge and hence no change in the direction of spiralling between these two edges. By Lemma 4.6 we know that this spiral does not cross itself. □

By the above lemmas, the path with the smallest number of links that minimizes total turn is just the minimum-link path that uses the geodesic inflection edges, because the amount of turn of any convex path between the inflection edges is fixed. The complete minimum-link path from $s$ to $t$ which uses geodesic inflection edges can be found in $O(n)$ time using the algorithm of Ghosh [Gho91]. His algorithm decomposes the problem between inflection edges of the geodesic shortest path and uses a subroutine to compute the minimum nested vertex path in each subpolygon. Alternatively, the original minimum-link path algorithm given by Suri [Sur87] (which does not necessarily give a minimum-link path that follows geodesic inflection edges) can be perturbed as described in Lemma 4.5 to give the path with minimum total turn. Therefore, we obtain the following result:

**Theorem 4.8** *The Total Turn and Links problem can be solved in $O(n)$ time for a polygon $P$ with $n$ vertices.*

We can extend this approach to solve the more general problem in a polygon with holes in polynomial time. In a polygon with holes we will use Lemmas 4.5, 4.6 and 4.7 to derive a brute force approach.

In our general strategy, we will first identify all candidate inflection edges from the visibility graph (i.e., VG edges that are locally supporting on opposite sides of the edge). To start the algorithm we will add the visibility graph edges around $s$ and $t$ as "inflection" edges. Inflection edges can be labelled to indicate whether they are clockwise ($cw$) or counterclockwise ($ccw$) pinned (i.e. they cannot be moved in the corresponding direction). The special "inflection" edges (VG edges for $s$ and $t$) are similarly labelled.

We would like to compute all optimal (convex or convex-spiralling) paths between two candidate inflection edges. Once we have found these paths we will be able to combine the answers we found between pairs of inflection edges using dynamic programming and find the best path between $s$ and $t$.

To store the answers, we create a directed graph with two nodes for each inflection edge, where we consider the inflection edge to be oriented in one direction or the other. We also create a *supersource* node connected to all of the visibility graph edges with one endpoint on $s$ and a *supersink* node connected to all of the visibility graph edges with one endpoint on $t$. We connect the supersource node to all the visibility edges around $s$ with arcs (0,0) representing 0 turn and 0 links. We do the same for the supersink and the VG edges around $t$.

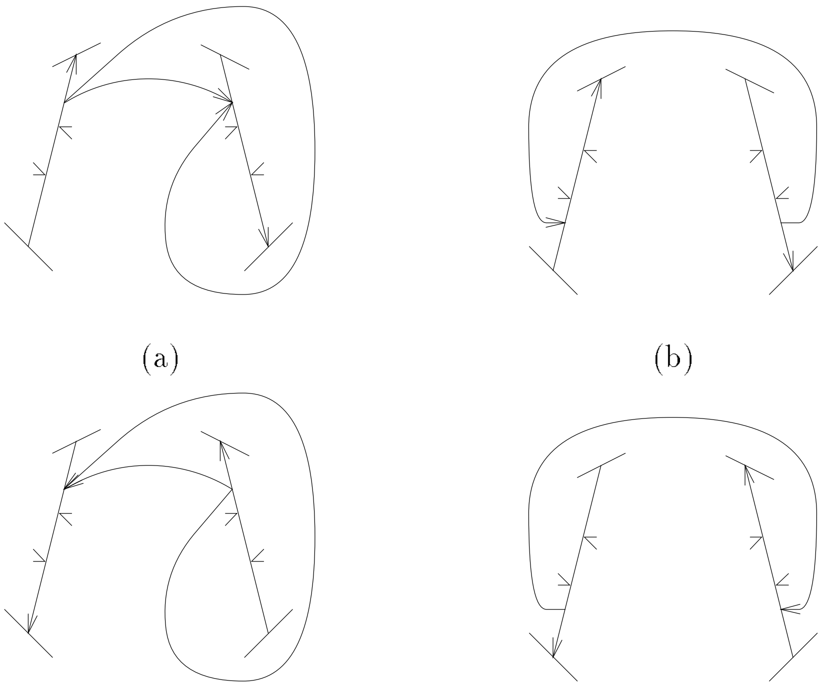
We first compute the degenerate paths between inflection edges, where they cross. To do this we compute the arrangement of the $O(E^2)$ inflection edges. Suppose two inflection edges $e_1$ and $e_2$ with orientation $\theta_1$ and $\theta_2$ cross. We connect the nodes corresponding to the oriented edges $e_1$ and $e_2$ with arcs labelled $(T, 1)$, where $T$ represents the amount of turn ($|\theta_1 - \theta_2|$ or $2\pi - |\theta_1 - \theta_2|$) appropriate for the change in direction between the two oriented edges.

Now, we need only compute the optimal convex or convex-spiralling paths between every pair of inflection edges that are pinned in opposite directions. Without loss of generality let $e_S$ be *cw*-pinned with orientation $\theta_S$ and $e_G$ be *ccw*-pinned with orientation $\theta_G$. If there were no possibility of a path between $e_S$ and $e_G$ spiralling then the total amount of turn between $e_S$ and $e_G$ would be either $|\theta_S - \theta_G|$ or $2\pi - |\theta_S - \theta_G|$. However, a path between them may spiral around $i$ times, adding a value $2\pi i$ to the amount of total turn.

In fact there may be a path with few links spiralling around several times, giving the combination of few links but a large amount of total turn, while another path may have many links but no spirals resulting in a smaller amount of total turn. The particular bounds on total turn and number of links would determine which of these paths would be deemed "better." When the number of links is restricted to be very small, the first path would be preferable, whereas when the amount of turn is very restricted the second path would be preferable. If both the number of links and the amount of turn are very limited then both paths would be considered undesirable.

Thus, for each pair of inflection edges $e_S$ and $e_G$ we must determine the number of links in a minimum-link convex-spiralling path between them which spirals at most $i$ times, for every $i$ between 0 and $k$. Then, for each value of $i$, we will have an arc in our graph between the nodes corresponding to $e_S$ and $e_G$ indicating the number of links in a minimum-link path between them.

There are four possible paths to consider from $e_S$ to $e_G$ for a particular value of $i$, depending on how each edge is pinned and whether we are looking for a convex or concave-spiralling path. (See Figure 4.7.) For each of these paths we will compute the value $l_i$, which is the number of links in the path with minimum total turn



(a) (b) (c) (d)

Figure 4.7: Different spiralling paths between inflection edges.

from $e_S$ to $e_G$ and where each path is subject to the constraint that it has total turn less than $\Delta\theta + 2\pi i$. The value $\Delta\theta$ is the appropriate change in angle from inflection edge $e_S$ to $e_G$ which is either $|\theta_S - \theta_G|$ or $2\pi - |\theta_S - \theta_G|$, depending on which of the four cases we are considering.

Now suppose that we have computed the four appropriate values between every pair of inflection edges. There are $O(E^2)$ possible pairs of inflection edges. Between each pair of them we will maintain up to $O(n)$ labels. Alternatively, we can keep $O(nE^2)$ total arcs in the graph, each with unique labels. Finally, by using the procedure described in Lemma 2.2, we can combine our solutions using dynamic programming and the information in such a graph. Thus we have found the "shortest" total turn path with $k$ links in $O(knE^2)$ time.

The time bound for combining the graph information via dynamic programming will be dominated by the time taken to compute the values for each arc. In the following section, we present a method of computing the arc information in time

$O(E\alpha(n)\log^2 n)$ for each appropriate pair of arcs. This gives a total time bound of $O(E^3 n\alpha(n)\log^2 n)$ over all $O(E^2)$ pairs of inflection edges. This implies we can compute the "shortest" total turn path with $k$ links in a simple polygon with holes in $O(E\alpha(n)\log^2 n)$ time.

### 4.2.2 Finding Intermediate Solutions

In this section, we present an algorithm to solve a subproblem of the Total Turn and Links problem. Given a fixed pair of inflection edges $e_S$ and $e_G$, and a fixed number of spirals $i$ between them, we want to find a convex or convex-spiralling minimum-link path between the two inflection edges. We consider only one case, where the paths are right-turning (see Figure 4.7(a)). The other cases are similar. To find the answer to this subproblem, we modify an algorithm by Mitchell, Rote and Woeginger [MRW92] for computing the minimum-link path between a start point and a goal point among polygonal obstacles in the plane. We need to make some modifications because their minimum-link path was not restricted to be convex or convex-spiralling. We will not review every detail of the algorithm given in their paper, but we will summarize the main concepts and indicate the main modifications needed to find the restricted (convex or convex-spiralling) minimum-link paths.

The basic idea used to compute minimum-link paths is to compute regions of visibility. The region $\mathrm{VIS}_1$ is the set of points that can be reached in one link from starting point or edge. The region of visibility $\mathrm{VIS}_r$ contains the points reachable from the start point with $r$ links. The next region, $\mathrm{VIS}_{r+1}$, is found by "illuminating" the boundary of $\mathrm{VIS}_r$. The points touched by light are the points reachable with $r+1$ links — i.e., the points in $\mathrm{VIS}_{r+1}$. However, when the obstacle scene has $n$ vertices, the complete description of the set of points at link distance $r$, $\mathrm{VIS}_r$, may have complexity $\Omega(n^4)$. Two key ideas in [MRW92] help to avoid this high complexity.

Their first idea is to apply a goal directed search. Rather than computing *all* the regions that can be reached in $k$ links, attention is restricted to the cell that *contains* the goal point or edge. They compute only the edges which will bound this cell. Their second idea is to simplify the relevant portion of the illumination boundary. Instead of illuminating directly from $\mathrm{VIS}_r$ to obtain $\mathrm{VIS}_{r+1}$, the boundary of what has been illuminated after $r$ steps is replaced by the relative convex hull of $\mathrm{VIS}_r$ with respect to the obstacle scene. They show that a point is visible from $\mathrm{VIS}_r$ if and only if it is visible from this relative convex hull. Amortized over all the illumination steps and the $O(E)$ visibility graph edges, the simplified visibility regions can be computed efficiently.

The overall algorithm proceeds in the following manner. Given a description of what was visible in $r$ steps, the algorithm computes a description of the next illuminated region. This description is a covering of the region by triangles and can be found efficiently. Then, the edges of the boundary that contribute to the cell of interest are found. The relevant boundary is simplified by computing the relative convex hull. Finally, a clean-up stage finds the cell containing the goal, and removes any extra cells that may be created by the boundary simplification step. Over all illumination steps, this algorithm takes $O(E\alpha(n)\log^2 n)$ time to find a minimum-link path from a start point $s$ to a goal point $t$.

In order to use this algorithm we must make modifications to the illumination steps to ensure that the visibility region $\mathrm{VIS}_r$ will be a region reachable by a *convex* or *convex-spiralling* $r$-link path. We assume that the visibility graph has already been computed in time $O(E + n\log n)$ [GM91]. In this same time bound, we also know the extension points of each visibility edge – the point where the extension of a visibility graph edge hits an obstacle. In time $O(E\log n)$ the extension points along common obstacle edges can also be sorted to give the extension points in order along each edge.

Now, let $G$ be the first vertex of the goal inflection edge. Our task is to cross $e_G$ with an illumination edge *from the correct side.* (See Figure 4.8.) By this we mean that the path enters from the appropriate direction with respect to the case we are considering. If the goal point $G$ is illuminated so that the last link $\overline{G'G}$ is convex with respect to the rest of the path then we can stop, because we have found the desired right-turning path from $e_S$ to $e_G$.

If $G$ is illuminated but from the wrong direction, so that the last link $\overline{G'G}$ is not convex with respect to the rest of the path, then we stop and do not join $e_S$ to $e_G$ for this particular value of spiralling $i$ since there is a shortcut that would use a different inflection edge between $e_S$ and $e_G$. (Recall that we are looking for short paths between $e_S$ and $e_G$ that have no inflection edges between them.) Refer to Figure 4.9. The dashed line that illuminates $\overline{G'G}$ from the wrong side can be

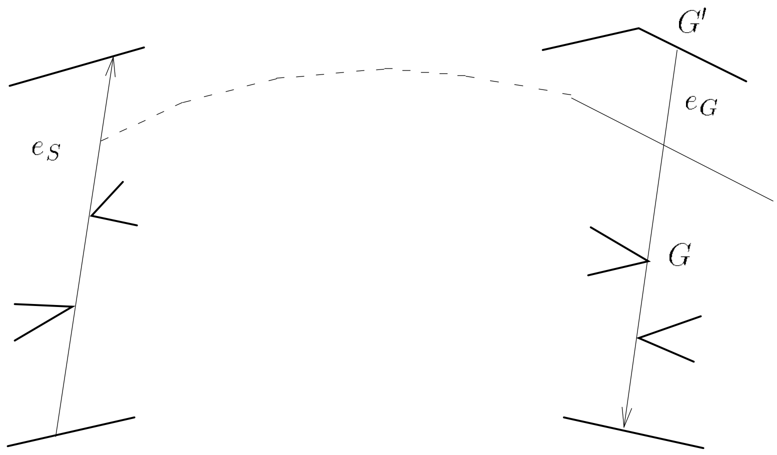



Figure 4.8: Goal is reached when we cross $e_G$.

twisted until it is counterclockwise-pinned. Thus there would be an inflection edge between $e_S$ and $e_G$. That shorter path will be found when intermediate answers are combined in the final dynamic programming stage.

We will be running our algorithm to find the minimum-link path between $e_S$ and $e_G$ many times, once for each spiral parameter value $i$ between 0 and $k$. For different values of $i$ the algorithm may find a different minimum-link path from $e_S$ to $e_G$. Large values of $i$ allow paths with few links but many spirals, i.e. large total turn. Small values of $i$ favor paths which may have many links but which have small total turn. We need to be careful in our minimum-link algorithm so that we are able to find this second kind of path. Our illumination steps must depend on the parameter $i$ which restricts the amount of spiralling. The final dynamic programming stage will choose which path, over all possible choices of $i$, will be incorporated in the complete path from the start to the goal.

The initial illumination step is accomplished by finding the set of edges that contribute to a set of triangles which cover the illuminated region. A subset of these visibility edges will become illumination edges in the next iteration.

We assume that an illumination step proceeds from the extension of a particular edge $\overline{uv}$ that was part of the boundary of the previously illuminated region. The visibility edges which contribute to the triangles illuminated from $\overline{uv}$ can have eight edge types depending on the position of their endpoints and whether they are

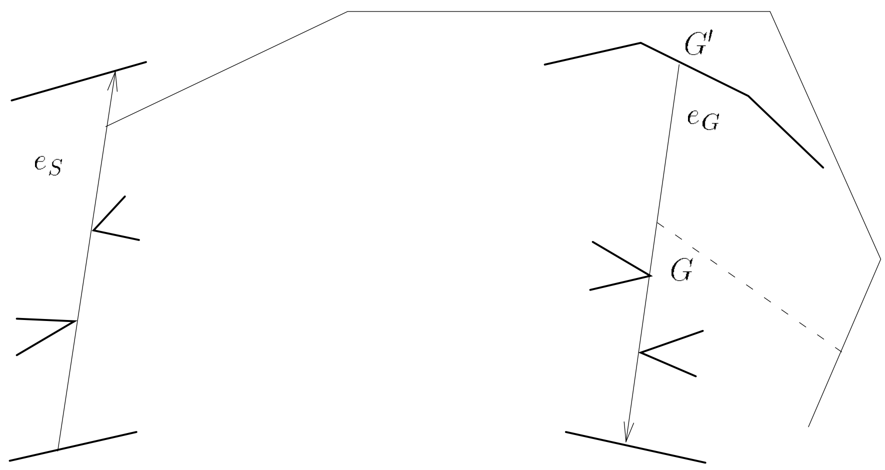



Figure 4.9: If $e_G$ is illuminated from the wrong direction, we stop.

pinned in a clockwise or counterclockwise direction. See Figure 4.10. We consider the types of all possible visibility graph edges which can cross the extension of $\overline{uv}$. Edges 1 and 2 in the diagram are pinned in the clockwise direction, and have both vertices on the illumined side. Edges 3 and 4 are pinned counterclockwise and also have both vertices on the illumined side. Other types of visibility graph edges also cross $\overline{vv'}$, but each will have at least one vertex on the old, formerly illumined side. As in [MRW92], such edges will not be necessary to compute the new visibility region.

Now label an edge an *illumination* edge when it satisfies the following three properties: (1) it is pinned in the correct direction, (2) the edge has the goal vertex $G$ on its correct side, and (3) the total turn of the path from $e_S$ up to and including the edge has less than $\Delta\theta + 2\pi i$ turn up to this point. In the algorithm, we maintain sums along edges as they become illumination edges to keep track of the total turn up to this step.

In the case we are considering, we are looking for right-turning paths. So, $G$ is on the correct side if it is to the right of the edge. Thus, for right turning paths we label some subset of edges of type 1 and 2 as illumination edges. We label the other edges with both vertices on the illumined side (type 1, 2, 3 or 4) as *blocking* edges, if they are either pinned in the wrong direction or have $G$ on the wrong side. The boundary of the set of points reachable from $\overline{vv'}$ in $r$ right-turning links will

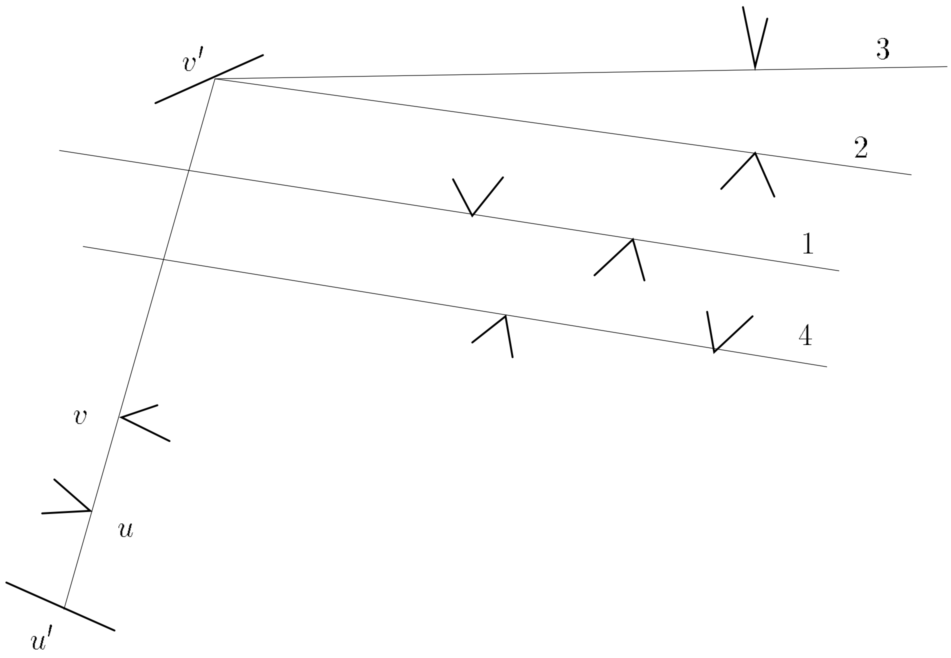



Figure 4.10: Visibility edges to compute right-turning visibility.

be made up edges of types 1, 2, 3, and 4.

In each step of the algorithm, as we illuminate the next region from an edge $\overline{uv}$ we must be able to identify all the relevant visibility and extension edges which cross $\overline{vv'}$. For the algorithm to be efficient, this identification must take time proportional to the number of relevant edges that cross $\overline{vv'}$ rather than $E$. This will be done efficiently by adapting the method used in [MRW92]. Then, the cost of finding the visibility regions is $O(E)$ when amortized over all illumination steps.

Once we have computed the illumination edges we would like to find the appropriate cell in the arrangement of these edges which contains the vertex $G$. This is same as the goal-directed idea of the original minimum-link path algorithm. We need to justify that keeping the illumination edges which form the boundary of this cell is sufficient also for finding a convex or convex-spiralling path with bounded total turn. We do so in the following lemma.

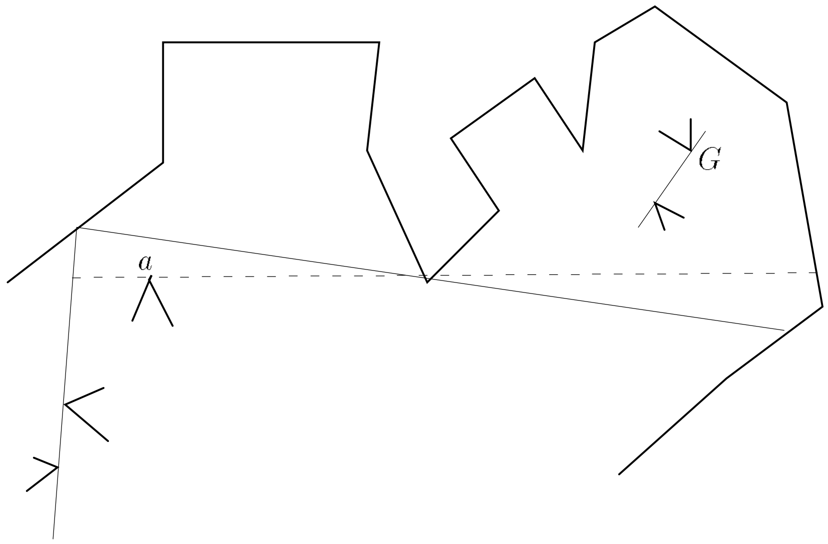



Figure 4.11: Goal in a blocked face.

**Lemma 4.9** *It suffices to use only illumination edges of the face of the arrangement that contains $G$.*

**Proof.** We must show that it is "safe" to stop illumination from the other edges. There are two main cases. In the first case, we argue that a path will not go through a blocked edge (if so we could shortcut). If there is a blocked edge on the boundary of $G$'s face then it must correspond to an extension of an inflection edge $\overline{ab}$ (otherwise, we could twist it to make it lie along a VG edge). This inflection edge is a witness to a path with shorter total turn and number of links from $e_S$ to $e_G$. This shorter path would lie along the inflection edge $\overline{ab}$. Thus this path will be found during the dynamic programming stage because we will tabulate the solution from $e_S$ to $\overline{ab}$ and from $\overline{ab}$ to $e_G$ at some point.

On the other hand, if there are *only* blocking edges around the face then there is no need to continue looking for a path from $\overline{vv'}$ to the goal inflection edge for this particular value of $i$. (See Figure 4.11.) This is evidence we will need to use an inflection edge to go from $e_S$ to $e_G$. That solution will be found in the dynamic programming stage.

In the second case, we show that it is safe to discard illumination edges that are not part of the boundary of the cell that contains $G$. If it were not safe

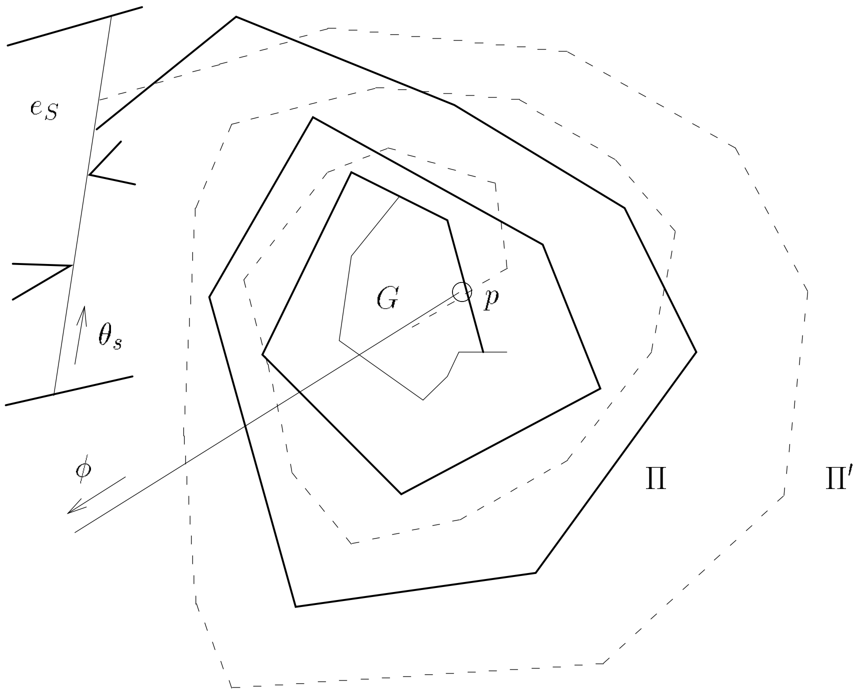



Figure 4.12: It suffices to use illumination edges to continue the path.

to do so it might be that a path with shorter total turns and links could "cut through" an illumination edge at a later stage. We show that if this were the case a shortcut would exist, so no such paths are possible. Thus it is safe to discard those illumination edges not on the boundary of the cell containing $G$. The tricky part is to consider the possibility of spiralling (something that was not a concern in [MRW92]). Without spiralling it is clear that when two paths have the same total turn and the same number of links, we can "forget" the illumination edges not on the face of the cell containing $G$.

When spiralling is allowed, we have to note that an optimal path cannot cut through an illumination edge at a later stage. We show that this cannot happen using a proof by contradiction. Consider a supposed optimal path that crosses an illumination edge. Call the current optimal path up to the illumination edge $\Pi$, and call the supposed new optimal path that crosses that edge $\Pi'$ (see the bold path and the dashed path in Figure 4.12.) Suppose that $\Pi'$ crosses the illumination edge at point $p$ and orientation $\phi$ and that $\theta_S$ is the orientation of the start inflection edge $e_S$. Let $\Delta\theta' = \theta_S - \phi$ be the appropriate difference in orientations between $e_S$ and this last link of $\Pi'$. The path $\Pi$ up to the illumination edge has total turn $\Delta\theta' + 2\pi j$ for some $j$. The path $\Pi'$ has total turn $\Delta\theta' + 2\pi j'$ for some $j'$.

If $j' < j$ then the illumination edge would not be blocking when a smaller value than $j$ spirals was used in the dynamic programming step. So, we would have found this short path already and tabulated it. If $j' > j$ then $\Pi'$ has too much turn and cannot be improved by crossing the illumination edge. If $j' = j$ then we can obtain a path with the same total turn as $\Pi'$ by using $\Pi$ up to the illumination edge and turning at point $p$. This path has fewer links than $\Pi'$, because of the way we proceeded with illumination steps. $\square$

The time bound of the algorithm presented in [MRW92] depends on a subtle charging scheme to *find* all of the appropriate types of new edges (type 1, 2, 3, 4 described above). It is important that the subgraph of the visibility graph searched to find these edges is connected, and that visibility graph edges that are used to find other edges are never used again. This charging scheme allows the cost of finding the new edges of visibility regions to be amortized over the whole algorithm and charged off to the $E$ visibility graph edges.

We state without elaboration that the same technique used in [MRW92] can be used to search for all edges of type 1, 2, 3 and 4. This technique searches a graph whose nodes correspond to a visibility graph with depth first search to find a connected subgraph which contains all the edges of type 1, 2, 3, and 4. We make a minor note that the proof of the connectivity of that subgraph never uses vertices not on the side of the illumined region. Thus, the graph is also connected for vertices on the illumined side only.

The running time of the algorithm for a fixed pair of inflection edges, a fixed amount of spiralling, and fixed number of links will be the same as the original algorithm, $O(E\alpha(n)\log^2 n)$ time.

Over all $O(E^2)$ inflection edges and all possible numbers of spirals $i$, $0 \le i \le n$, we will compute a minimum-link path between each pair that has at most $i$ spirals. We have at most four arc directions to compute between each pair, so we can complete our arc information in $O(E^3 n\alpha(n)\log^2 n)$ time.

### 4.2.3 Final Result

Combining the results of the previous two sections, we can compute the arc information in time $O(E\alpha(n)\log^2 n)$ for each appropriate pair of arcs, giving a total time bound of $O(E^3 n\alpha(n)\log^2 n)$ over all $O(E^2)$ pairs of inflection edges.

**Theorem 4.10** *In a simple polygon with holes, given a start point $s$, a goal point $t$, a bound on total turn $\theta$ and a bound $k$ on the number of links, the Total Turn and Links Problem can be solved in time $O(E^3\ n\ \alpha(n)\ log^2 n)$, where $E$ is the number of visibility graph edges among the obstacles.*

It is possible that a more efficient illumination scheme could compute the necessary arc information faster. Perhaps one could use the illumination steps from one edge to create a map of the number of links needed to reach every other vertex (for a fixed number of spirals $i$). This would be more efficient than recomputing the arc information for every pair of edges. This approach was taken in [MRW92] to compute a map from the source vertex to all other obstacle vertices in the scene. However, this result needs to use some fast algorithms to find many faces of the arrangements, and it is not clear if these algorithms extend to convex-spiralling paths. We would expect that if the illumination scheme could be improved, the $O(knE^2)$ running time of the dynamic programming stage would dominate the running time of the overall algorithm.

# Chapter 5

# Computing a Shortest $k$-Link Path in a Polygon[1]

A shortest $k$-link path is a path that is "good" with respect to link distance *and* Euclidean length. In this chapter, we give the first algorithmic results for the problem of finding a shortest polygonal path from $s$ to $t$ within a simple polygon $P$, subject to the restriction that the path have at most $k$ links or edges. Unlike previous work, we would like to consider the problem where there are no restrictions on the orientations of links. We give a full characterization of the structure of shortest $k$-link paths in a polygon with or without holes, and prove necessary local optimality conditions. Based on these structural results, we give an algorithm to compute a $k$-link path with length at most $(1+\epsilon)$ times the length of a shortest $k$-link path, for any error tolerance $\epsilon > 0$. Our algorithm runs in time $O(n^3 k^3 \log(Nk/\epsilon^{1/k}))$, where $N$ is the largest integer coordinate among the $n$ vertices of $P$. Note that the dependence on $\epsilon$ is *logarithmic* in $1/\epsilon$. Within the same time bound, the algorithm can also compute an $\epsilon$-optimal path under any single *combined* objective, $f(L, G)$, where $L$ and $G$ denote link distance and Euclidean length, and $f$ is an increasing function in $G$ for each $L$.

[1]This chapter contains joint work with Esther Arkin and Joseph S. B. Mitchell. An extended abstract of this material appeared in the proccedings of FOCS 92.

## 5.1 Overview

A minimum-link path can be arbitrarily longer (in Euclidean length) than a shortest length path. (Refer to Figure 5.1.) In his thesis, Suri [Sur87] posed the natural question: How can one find a shortest (in Euclidean length) path that uses at most $k$ links?

The difficulty of this problem is due to the non-uniqueness of $k$-link paths. In fact, there may be an infinite number of different $k$-link paths joining a given pair of points in a polygon. Even when $k$ equals the link distance between the pair of points, there may be a continuum of $k$-link paths. In contrast, the Euclidean shortest path, also know as the geodesic path, between two points in a polygon with holes is almost always unique. (It is not unique when one of the points lies on a bisector curve in the shortest path map with respect to the other point.)

Another main technical issue we must address is that shortest $k$-link paths need not lie on a visibility graph, or on any simple variant of the visibility graph. Shortest $k$-link paths may turn at points interior to the polygon whose coordinates are given by the roots of high-degree polynomials. Thus, we are unable to obtain a path that exactly minimizes the Euclidean length among all $k$-link paths. Instead, our algorithms produce a *provably good* approximation of a shortest $k$-link path. This approximation has length at most $(1+\epsilon)$ times the length of an optimal path. Note that by expressing paths as formulas in the theory of real closed fields, one can obtain, in *exponential time*, an exact solution for the decision problem: Does there exist a $k$-link path of length at most $d$?

In contrast to previous work, this problem of finding shortest $k$-link paths *without orientation restrictions* admits no finite path-preserving graph. Instead, we perform an analysis of the local and global structure of shortest $k$-link paths in simple polygons. We then use the global structure to decompose the problem into a set of "raceway" problems. Within each raceway, we further decompose the problem into $O(n^2)$ "elementary" problems, which ask us to find the shortest $i$-link paths between pairs of "flush" edges (defined below). We solve these elementary problems by a binary search whose tests consist of tracing locally optimal paths. We use dynamic programming to combine the solutions to the elementary problems into raceway solutions and to combine the raceway solutions into a global solution.

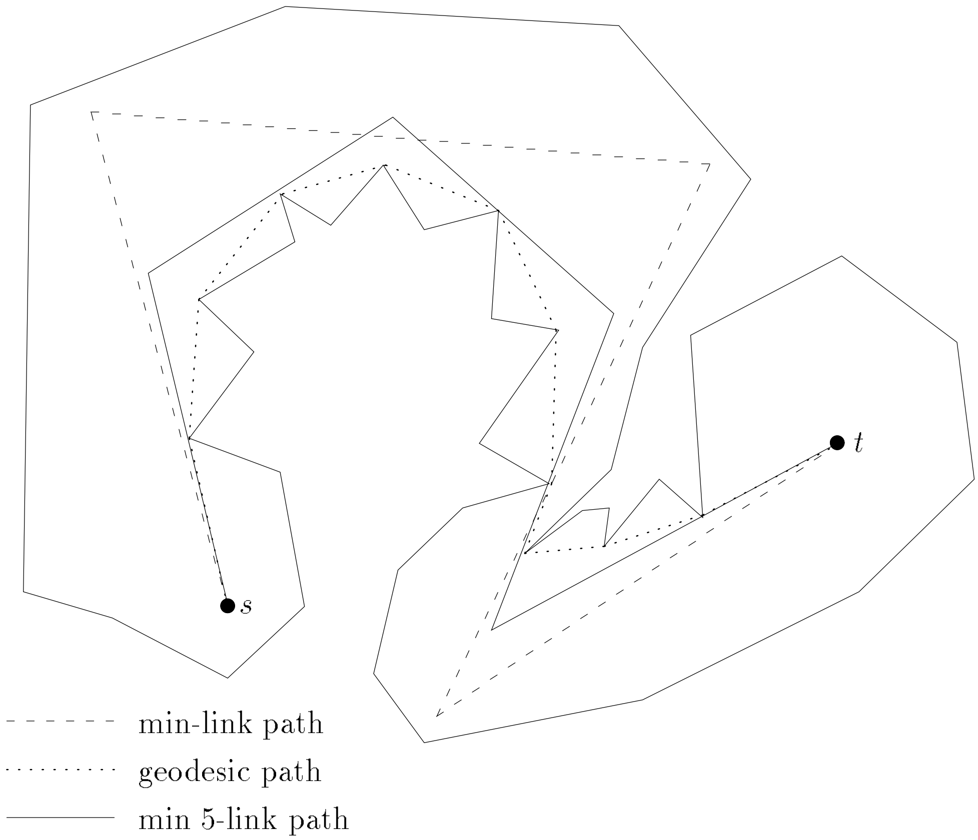



Figure 5.1: A minimum-link path, a shortest path, and a shortest 5-link path.

We begin by presenting an algorithm for a relaxed version of the problem, then move on to the more difficult case. In the relaxed version of this problem, our algorithm finds a $(2k-2)$-link path whose Euclidean length is no more than the length of the shortest $k$-link path. This algorithm does use a "path preserving" graph approach. The running time for this algorithm is $O(kE^2)$, where $E$ is the number of edges in the visibility graph. A variation of this approach yields an alternative method for producing a $k$-link path (not $(2k-2)$-link) within a $(1+\epsilon)$ factor of optimal length in a simple polygon without holes in time $O(k^3n^2/\epsilon)$.

## 5.2 A General Approximation Scheme

In this section we consider the problem of computing a short (Euclidean length) path with a restricted number of links in a polygon with holes. If we restrict ourselves to rectilinear paths within $P$ (or to paths whose links come from any fixed finite set of allowable directions), then our problem can be solved exactly by a fairly straightforward method. (See [dBvKNO90,dB91,dBvKN91,YLW91] for a discussion of rectilinear path problems and more sophisticated algorithms.) Thus, we would like to solve this problem without restrictions on orientations. The exact problem seems hard, and in fact we suspect that it is NP-hard. Nevertheless, in a simple polygon with holes we are able to show the following:

**Theorem 5.1** *Let $P$ be a polygon, possibly with holes, whose boundary consists of $n$ edges. For any $k$, $d_L(s,t) \leq k \leq |\pi_G(s,t)|$, one can compute an $s$–$t$ path in $P$ that has at most $(2k-2)$ links and has length at most that of a shortest $k$-link $s$–$t$ path, in time $O(kE^2)$, where $E$ is the number of visibility graph edges.*

To find this $(2k-2)$-link path we construct a discrete graph, based on an arrangement of line segments within $P$, that is guaranteed to contain the path. For each vertex $v$ of $P$ (and also for $v = s, t$), and for each vertex $u$ of $P$ that is visible from $v$, we extend a segment from $v$ in the direction of $u$, stopping when we enter the exterior of $P$ at some point $u'$. We say that $\overline{vu'}$ is the segment *anchored at $v$ in the direction of $u$* and that $\overline{uu'}$ is the *extension segment associated with $\overline{vu'}$*. This yields a set $S$ of $O(E)$ segments within $P$, where $E$ is the size of the visibility graph, $VG(P)$.

We now construct a graph $\mathcal{G}$ (see Figure 5.2) whose nodes correspond to the segments of $S$. We connect two nodes with an edge in $\mathcal{G}$ if the associated extension segments cross (at a point interior to $P$). The length of such an edge, joining nodes corresponding to segments $\sigma_1$ and $\sigma_2$ (anchored at vertices $v_1$ and $v_2$, $v_1 \neq v_2$), is defined to be the length of the path $v_1$ to $\sigma_1 \cap \sigma_2$ to $v_2$. We also connect a node $\sigma$ to node $\sigma'$ with a zero-length edge, when $\sigma$ and $\sigma'$ are anchored at a common vertex $v$.

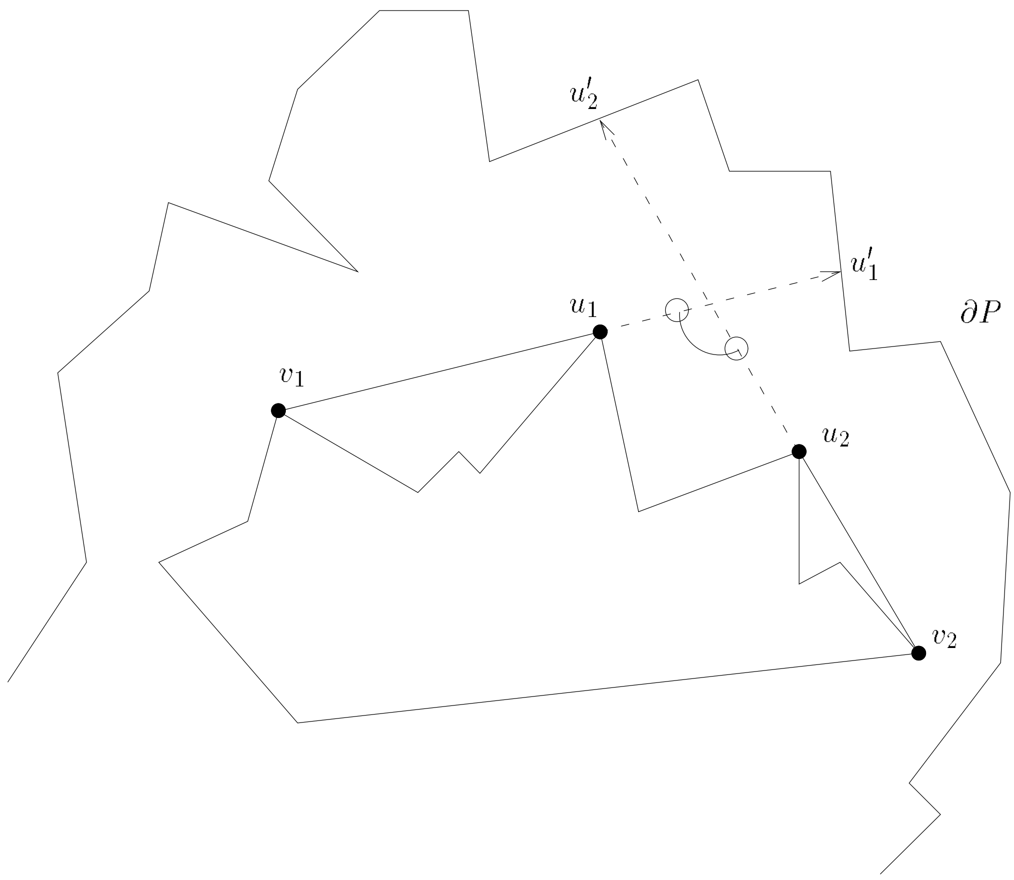



Figure 5.2: Definition of graph $\mathcal{G}$.

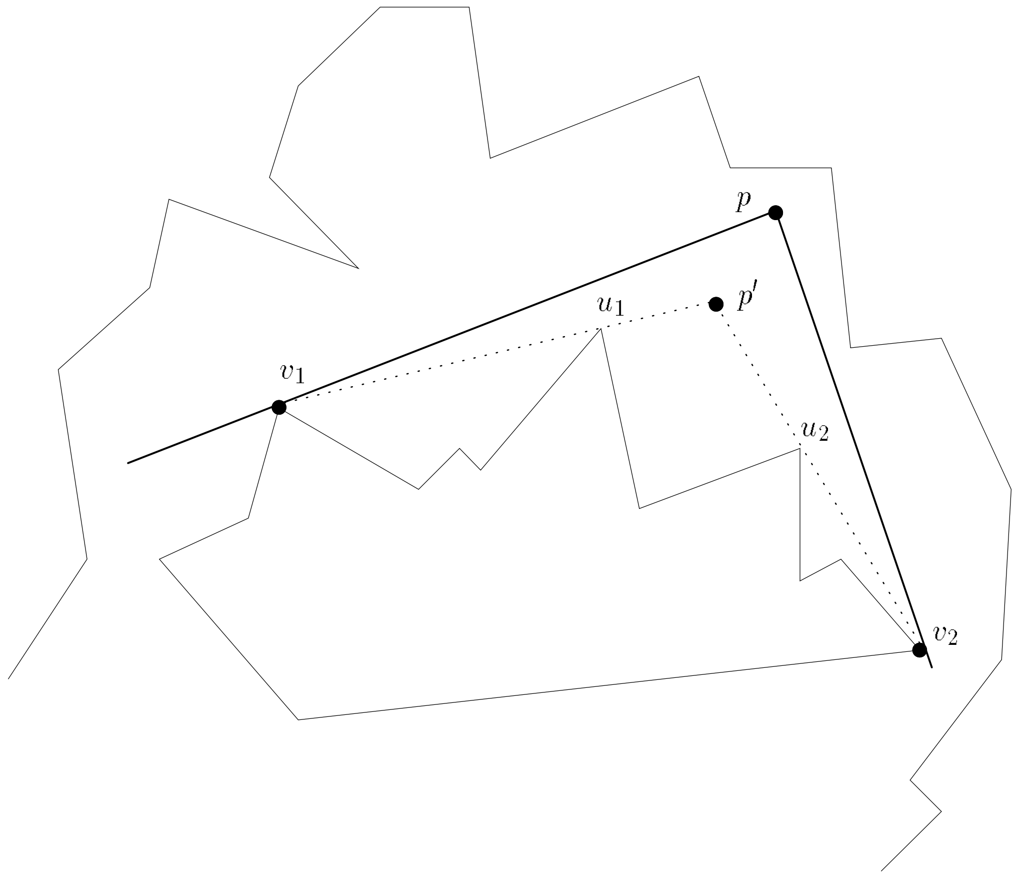



Figure 5.3: $\mathcal{G}$ is connected and provides a good approximation.

**Lemma 5.2** *The graph $\mathcal{G}$ is connected, and corresponding to any shortest $k$-link $s$–$t$ path of length $d$ in $P$ there is a path in $\mathcal{G}$ with at most $2k$ nodes, such that the corresponding segments constitute a $(2k-2)$-link $s$–$t$ path with length at most $d$.*

**Proof.** To see this, consider Figure 5.3. Suppose the dark path represents two of the links in a shortest $k$-link path, with an interior turn point $p$ between them. These links touch vertices $v_1$ and $v_2$. (See Lemma 5.5 for discussion of how to perturb a shortest $k$-link path so that each edge of the path in contact with at least one vertex of $P$.) By "breaking" each link on the vertex it touches and swinging each link down until it hits a vertex ($u_1$ and $u_2$ respectively), we have moved the path onto one that is represented in our graph $\mathcal{G}$. We do not need to break the first and last links adjacent to $s$ and $t$ respectively. Each of the $k-2$

links in between the first and last link is broken at most once, so we have at most doubled the number of links in between. Also, the length of the two new links is less than that of $\overline{v_1p} + \overline{pv_1}$. (This geometric fact is shown as part of Lemma 2.6. See also Figure 2.6). Thus, the length of the path is less than or equal to the length of the optimal shortest $k$-link path. □

The graph $\mathcal{G}$ has $O(E)$ nodes and $O(E^2)$ edges, and it can be constructed in time $O(E^2)$. By a method of Chazelle and Edelsbrunner [CE92] one can construct the intersection of $E$ segments in "output sensitive" time, that is in time proportional to the actual number of intersections. Their algorithm runs in time $O(ElogE+k)$, where $k$ is the number of intersection points. In the worst case $k$ can be $E^2$. We can then use dynamic programming to compute a shortest $(2k-2)$-node path in $\mathcal{G}$ from $s$ to $t$ in time $O(kE^2)$. (Refer to Lemma 2.2 for more details of this dynamic programming method).

An improvement on the number of links can be achieved in the following way. If we define the graph $\mathcal{G}$ slightly differently, taking special care at inflection edges, then we can find a $k+C$-link path, instead of a $(2k-2)$-link path. where $C$ is the number of non-inflection edges in an optimal $k$-link path. This approach is similar to the approach of Ghosh for finding a minimum-link path without length restrictions [Gho91].

We also note that while there is a unique homotopy type for paths from $s$ to $t$ in a simple polygon, there are an exponential number of possible threadings for an $s$ to $t$ path in a polygon with $n$ holes. In fact, this simple $(2k-2)$-link approximation that we derived for a polygon with holes may not give a path with the same homotopy as the true shortest $k$-link path.

We give a different algorithm that should be easier to implement for a simple polygon without holes. We show that it is possible to get within a constant factor of both the minimum-link distance and the Euclidean shortest path simultaneously.

**Theorem 5.3** *In a simple polygon $P$ one can compute an $s$-$t$ path which has at most twice the number of links that are in a minimum-link path from $s$ to $t$ and Euclidean length at most $\sqrt{2}$ times the length of the geodesic shortest path between $s$ and $t$.*

**Proof.** Consider the "greedy" minimum-link path from $s$ to $t$ computed by the algorithm given by Ghosh [Gho91]. The minimum-link path will use the inflection

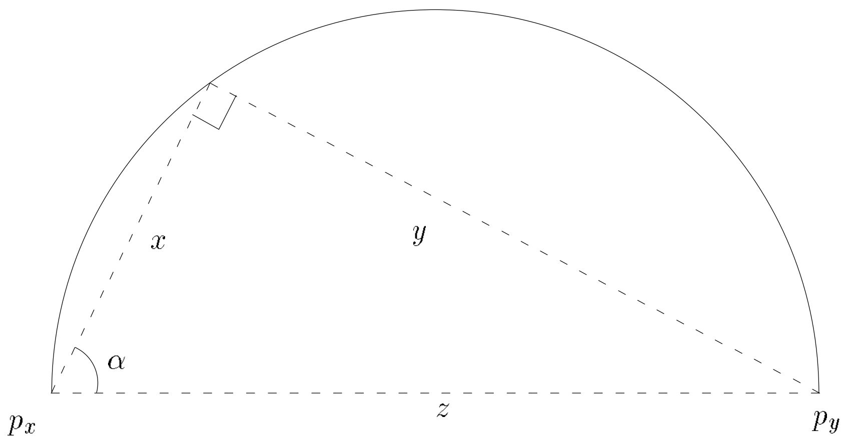



Figure 5.4: Sum of edges $x$ and $y$ of a right triangle is at most $\sqrt{2}z$.

edges of the geodesic path, and its non-inflection links will go through vertices of the geodesic path. Suppose this path has $L$ links. By adding at most one link between the consecutive links of this path we prove we obtain a path with at most twice the number of links ($2L$), and with Euclidean length at most $\sqrt{2}$ times the length of the geodesic shortest path between $s$ and $t$.

Consider two consecutive links $x$ and $y$ on the greedy minimum-link path from $s$ to $t$. If the links $x$ and $y$ formed a perfect right angle with each other then $x + y$ will be at most $\sqrt{2}z$ by the following argument. (Refer to Figure 5.4.) Let $\alpha$ be the angle between $x$ and $z$. We know that $x^2 + y^2 = z^2$ since this is a right triangle. We also know that $\sin\alpha = y/z$ and $\cos\alpha = x/z$. Also $x + y = z(x/z + y/z)$, so we know $x + y = z\sin\alpha + \cos\alpha$. The function $f(\alpha) = \sin\alpha + \cos\alpha$ is maximized when $df/d\alpha = \cos\alpha - \sin\alpha = 0$. This is attained for $\sin\alpha = \cos\alpha = \sqrt{2}/2$. In this case $\sin\alpha + \cos\alpha = \sqrt{2}$; thus $x + y$ is at most $\sqrt{2}z$. Since $z$ is a lower bound on the length of the geodesic path between $p_x$ and $p_y$, we know that $x + y$ is most $\sqrt{2}$ times the length of that geodesic path as well.

If the angle formed between the links $x$ and $y$ is obtuse then it is also within $\sqrt{2}$ of the geodesic shortest path by the following argument. Consider Figure 5.5. We know that $x$ and $y$ must be within the circular arc which is the locus of points where $x$ and $y$ would form an exact right angle with each other. Let us extend $y$ until it hits the arc giving a segment with length $y'$. Let $x'$ be the length of the

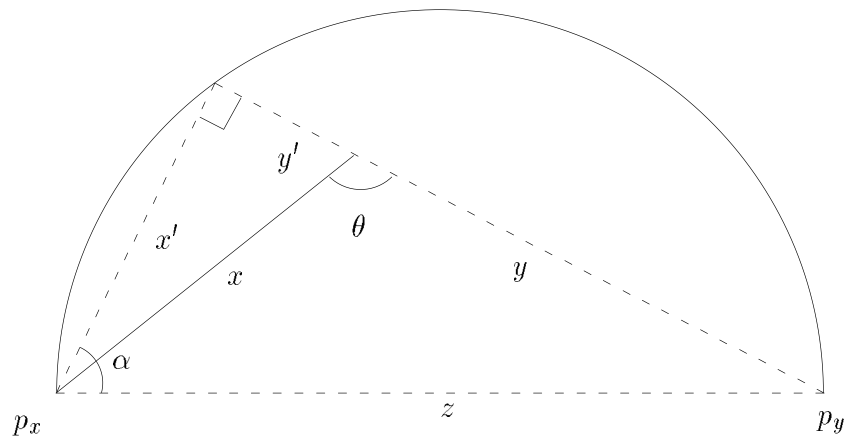



Figure 5.5: Sum of edges $x$ and $y$ of an obtuse triangle is at most $\sqrt{2}z$.

segment from that arc point to $p_x$. By the triangle inequality, $x \leq x' + y'$ implies $x + y \leq x' + y' + y$. By our previous observation for right triangles we know that $x' + y' + y \leq \sqrt{2}z$. Thus, for an obtuse angle $x + y \leq \sqrt{2}z$.

So, we know if two consecutive links $x$ and $y$ form a right angle or an obtuse angle with each other, they are at most $\sqrt{2}$ times the length of the geodesic path from $p_x$ to $p_y$. If two consecutive links $x$ and $y$ form an acute angle then we must add an extra link between them in order to make the same guarantee. (See Figure 5.6.) If $x$ and $y$ form an acute angle we add a middle link $m$ perpendicular to $x$ so that $m$ is tangent to the geodesic path between $p_x$ and $p_y$. Let $t$ be the point of tangency. (If $m$ happens to be flush with the geodesic path choose a representative tangency point $t$. If the edge from $p_x$ to $p_y$ is on the geodesic path we can then let $t$ be $p_y$). Let $x'$ be the part of $x$ before $m$, and $y'$ the part of $y$ after $m$. Let $m_x$ be the length of $m$ between $x$ and $t$ (similarly let $m_y$ be the length between $t$ and $y$). We have chosen $m$ so that the angle $\theta_1$ between $x$ and $m$ is exactly a right angle. The length of the geodesic path from $p_x$ to $t$ is bounded below by the length $z_1$ from $p_x$ to $t$. By our observation for right triangles we know that $x' + m_x \leq \sqrt{2}z_1$. Similarly, the length of the geodesic path from $t$ to $p_y$ is bounded from below by the length $z_2$ from $t$ to $p_y$. The angle $\theta_2$ between $m$ and $y$ is obtuse, since it is equal to $\theta + \pi/2$. Thus, by our observation for obtuse triangles we know that $m_y + y' \leq \sqrt{2}z_2$.

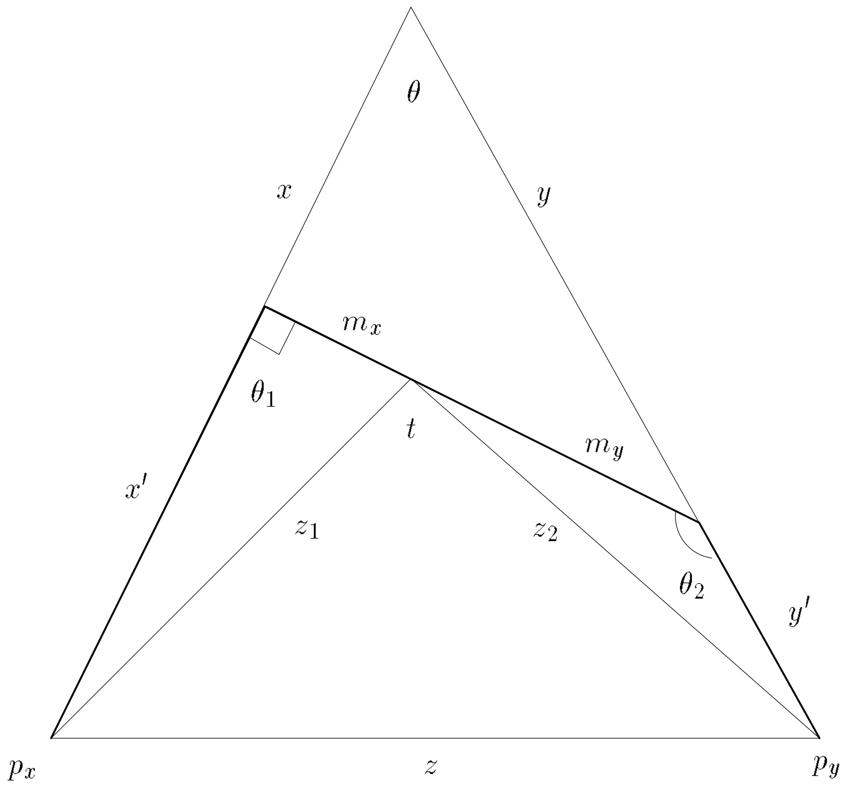



Figure 5.6: Adding a shortcut between an acute turn.

Combining these, we know that by adding the (perpendicular) middle link $m$, the path $x' + m + y'$ we obtain a path at most $\sqrt{2}$ times the length of the geodesic path from $p_x$ to $p_y$.

We need to add most $L$ such links to the original greedy minimum-link path (in the case that every pair of angles between consecutive links is acute). After adding all such links we have a path with at most $2L$ links with length within a factor of $\sqrt{2}$ of the geodesic optimal path between $s$ and $t$. $\square$

This same approximation *cannot* be made in a polygon with holes. In fact, in a polygon with holes there is no way to get within a constant factor of both the minimum-link distance and the geodesic (Euclidean) shortest path simultaneously. To see this consider Figure 5.7.

In this example, the middle path joining $s$ and $t$ between the obstacles can be made so that it has a short Euclidean distance but it has $\Omega(n)$ links. The upper or

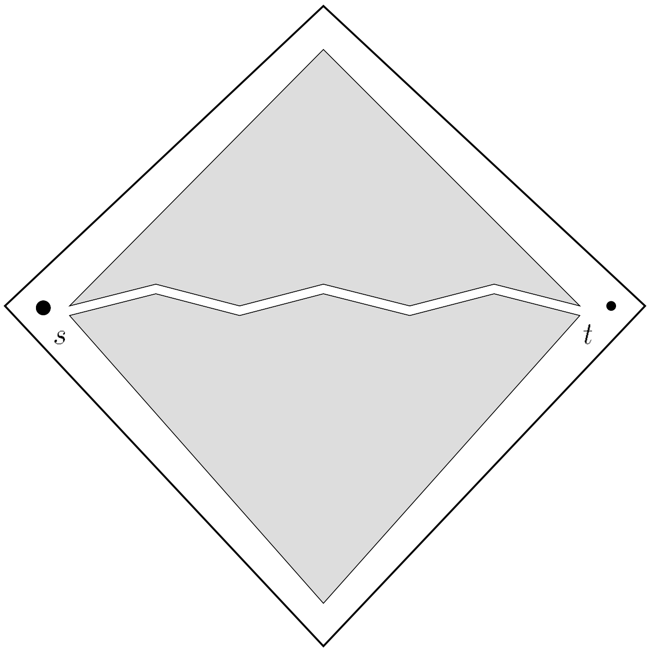



Figure 5.7: Polygon with two holes, cannot approximate links/length at same time.

lower paths around the obstacle have just two links but can be made so that they have Euclidean length arbitrarily longer than the Euclidean length of the middle path. Thus the link distance of the upper or lower paths is not within a constant factor of the link distance of the middle path, and the Euclidean distance of the middle path is not within a constant factor of the upper or lower paths. Since these are the only three possible paths between $s$ and $t$, in general one cannot find an approximate path between $s$ and $t$ in a polygon with holes that has link length and Euclidean length both within a constant factor of both the minimum-link distance and the geodesic length.

In a simple polygon without holes, where there is just one possible homotopy type for an optimal path, we can derive a closer approximation than the one given for a polygon with holes in Theorem 5.1. In fact we claim that in a simple polygon $P$ (without holes), whose boundary consists of $n$ edges, for any $k$, $d_L(s,t) \leq k \leq |\pi_G(s,t)|$, one can compute, in time $O(k^3 n^2/\epsilon)$, an $s$–$t$ path in $P$ that has at most $k$ links and has length at most $(1+\epsilon)$ times that of a shortest $k$-link $s$–$t$ path.

We outline a proof of this claim. First, we will use the fact that the Euclidean metric can be approximated to accuracy $\epsilon$ by a fixed-orientation metric of $K = O(1/\sqrt{\epsilon})$ equally-spaced directions (see Lemma 2.5). For each vertex $v \in \pi_G(s,t)$ and each of the $K$ directions, construct a $k$-link "greedy" path from $v$ to $t$ where the first link of the path uses the given fixed orientation. This can be accomplished by modifying Suri's greedy minimum-link path algorithm [Sur86]. Let $S$ be the set of segments on the paths constructed. We claim that the resulting arrangement of these $nkK$ segments is sufficiently "rich" to contain an approximating $k$-link $s$–$t$ path. A shortest $k$-link path can be perturbed onto the arrangement by a sequence of "twists", in which each link is rotated slightly (by at most angle $2\pi/K$) onto a segment of the arrangement, *while maintaining the connectivity of the $k$ links.* A proof of correctness uses induction on the number of links. Assume the first $i$ links have been twisted onto the arrangement. When the $i+1$st link is twisted (say, clockwise), it cannot disconnect the path, since the path can become disconnected only if it is rotated past the endpoint of the $i$th link. However, there is a segment attached to the endpoint of the $i$th link, leaning against the geodesic path. So the $i$th link is either a $K$-orientation segment (in which case the next greedy edge must exist among the set $S$), or it is one segment in a greedy extension path out of some other $K$-orientation segment. In either case, we know that the next segment is in the set $S$. To complete the proof of correctness of this approach we must appeal to some structural lemmas shown in Section 5.3 as well as lemmas similar to the Trapping and Dilation Lemmas in Section 5.4. However, this approach results in an algorithm that has a worse dependence on $\epsilon$ than the algorithm we will present next. We have sketched this approach and proof because the algorithm may be easier to implement.

In Section 5.4 we will show that in a *simple* polygon $P$, the dependence of the running time on $\epsilon$ can be made *logarithmic* in $1/\epsilon$, at a cost of a factor of $n$ and dependence on the term $\log N$. First we must make a careful analysis of the structure of an optimal path.

## 5.3 Characterizing Shortest $k$-Link Paths

Let $\pi^*$ denote a shortest $k$-link path from $s$ to $t$, and let $d^*$ denote the Euclidean length of $\pi^*$. Our goal now is to give a characterization of the structure of $\pi^*$. Let

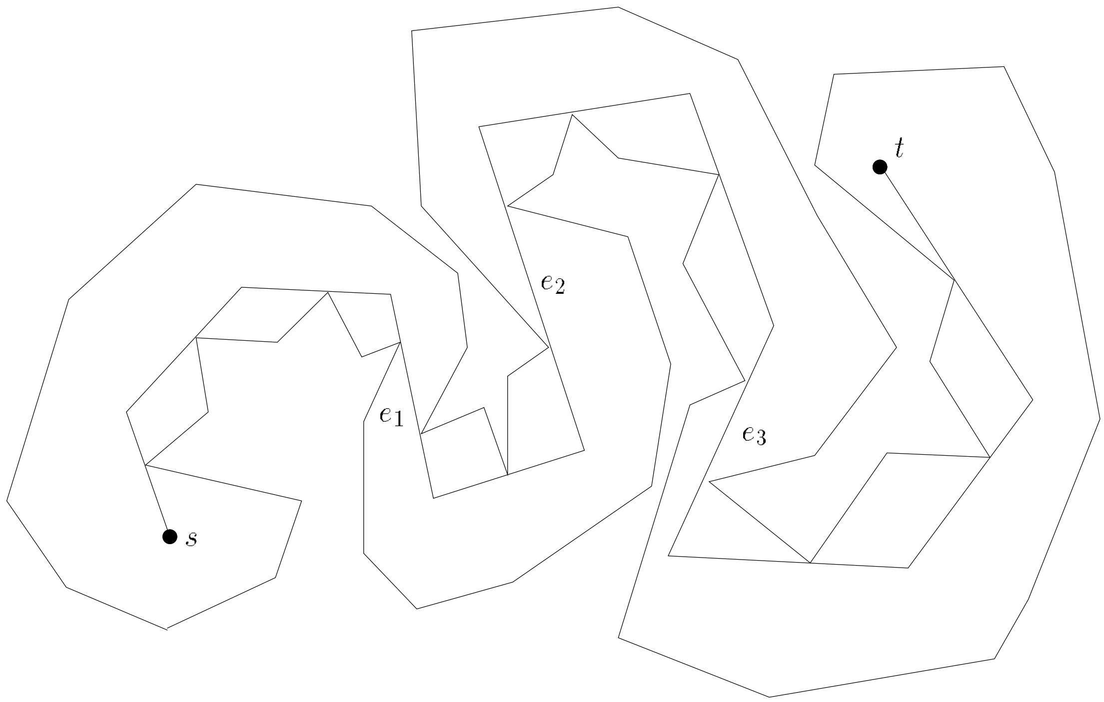



Figure 5.8: The inflection edges of $\pi^*(s,t)$.

the inflection edges of $\pi_G(s,t)$ be denoted $e_1, \ldots, e_I$; let $e_0$ and $e_{I+1}$ denote the first and last links of $\pi_G(s,t)$. Note that the edges $e_j$ partition $\pi_G(s,t)$ into subchains that are purely left-turning or purely right-turning. We call these subchains *convex chains*, although it should be understood that they can spiral, etc. (See Figure 5.8.)

We will show that $\pi^*$ will follow the inflection edges of $\pi_G(s,t)$. Later, this will allow us to decompose the problem of computing $\pi^*$ into the problem of computing short paths between inflection edges.

**Lemma 5.4** *Let $\pi^*$ be a shortest $k$-link path in a simple polygon $P$. Each inflection edge $e_j$ in the geodesic path $\pi_G(s,t)$ must be a subset of an edge of $\pi^*$, and each inflection edge of $\pi^*$ must contain an inflection edge of $\pi_G(s,t)$.*

**Proof.** Suppose not. The inflection edge $e_j$ can be extended in two directions to the boundary of $P$, giving a segment $e_j'$. The extension of the segment $e_j'$ partitions $P$ into four subpolygons: one containing $s$, a "bay" $B_j$ adjacent to this, a "bay" $B_j'$ adjacent to $B_j$ (with $B_j \cap B_j' = e_j$), and a polygon adjacent to $B_j'$, containing

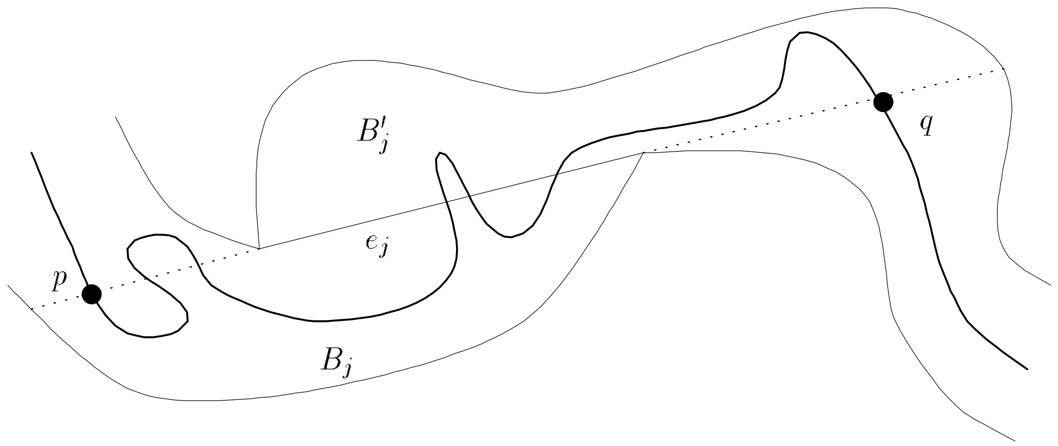



Figure 5.9: Shortest $k$-link path will use inflection edges.

$t$. (See Figure 5.9). The path $\pi^*$ must have at least one vertex in each of the two bays $B_j$ and $B'_j$. If $p$ is the first entry point of the path into $B_j$ and $q$ is the last exit point of the path from $B'_j$ then we can shortcut $\pi^*$ from $p$ to $q$, without increasing its link length or Euclidean length. This would imply that $\pi^*$ was not a shortest $k$-link path. □

*Remark.* This lemma also holds for polygons with holes, provided that we define $\pi_G(s,t)$ to be the shortest path homotopically equivalent to $\pi^*$.

**Lemma 5.5** *Let $\pi^*$ be a shortest $k$-link path in a simple polygon $P$. Each edge $e$ of $\pi^*$ can be perturbed to lie on at least one vertex of $P$.*

**Proof.** We can use a perturbation argument. If some link edge $e$ does not lie on a vertex we can perturb it by moving it parallel to itself until it comes in contact with at least one vertex of $P$. By using this new link parallel to the old one as a shortcut we have a new path that has the same or fewer links and the same or less Euclidean length as $\pi^*$. □

The bend points of path $\pi^*$ are either *interior* bend points (interior to $P$) or *boundary* ("bash") bend points (lying on some boundary edge of $P$). We call an edge of $P$ that contains a bash point a *bash wall* of $P$.

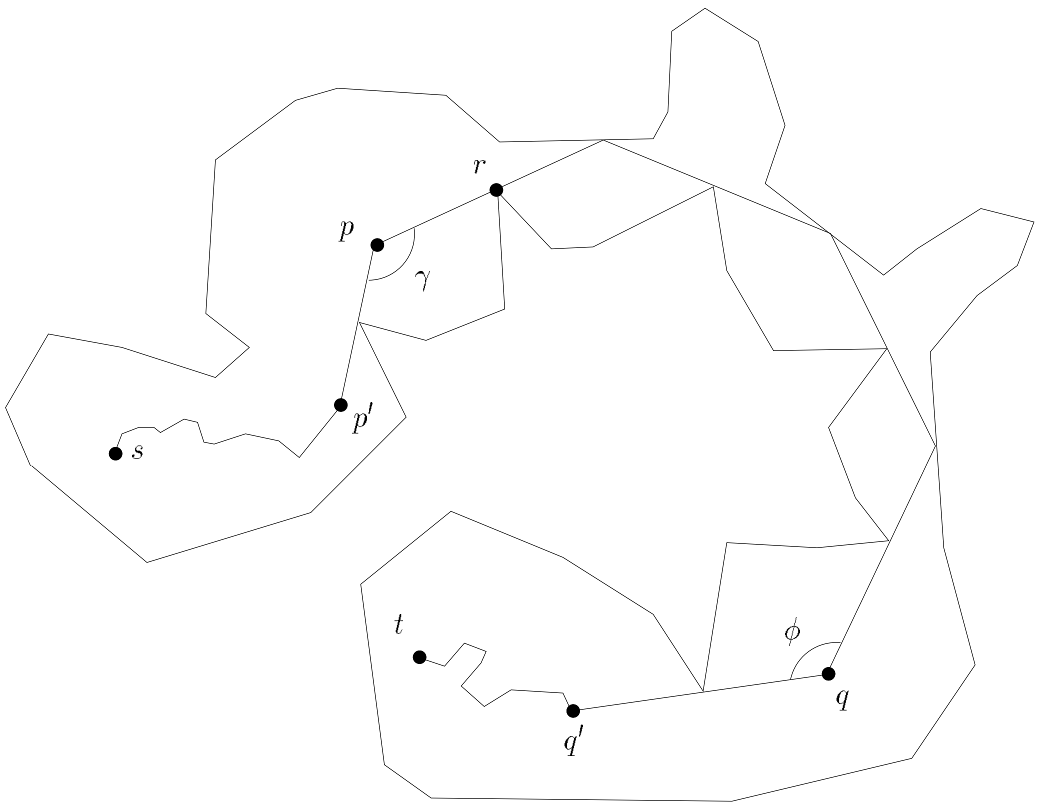



Figure 5.10: A balanced chain $\pi'$.

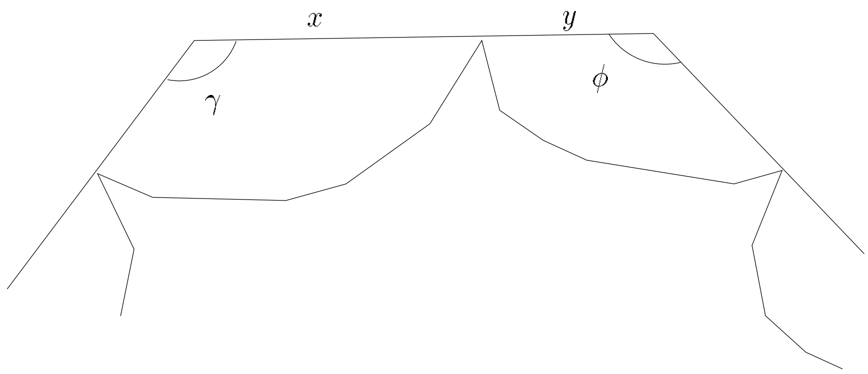



Figure 5.11: A balanced edge.

Let $\pi$ be a polygonal $s$–$t$ path and let $\pi'$ be a subpath of $\pi$ joining two consecutive interior bend points, $p$ and $q$, of $\pi$. Refer to Figure 5.10. Thus, the bend points of $\pi'$, if any, are all bash points. Let $p'$ be the predecessor of $p$ along $\pi$ and let $q'$ be the successor of $q$ along $\pi$. Assume that all edges of $\pi'$ are rocking. An edge of $\pi^*$ that is not inflection nor flush is called *rocking* if it is in contact with a (single) vertex of $P$ (a *rocking vertex*) (see Figure 5.14). We say that $\pi'$ is a *balanced chain with respect to* $\pi$ if the path from $p'$ to $q'$ via $\pi'$ is shortest among paths with the same number of links. In the special case that $\pi'$ is a balanced chain consisting of the single edge $\overline{pq}$, we say that $\overline{pq}$ is a *balanced edge* (Figure 5.11).

We can characterize balanced chains as follows: The first link of $\pi'$ rests on a rocking point $r$; if we perturb it by a small rotation about $r$, while adjusting all of the other links of $\pi'$ accordingly (so that $\pi'$ remains connected, with all of its bend points being bash points), then the resulting perturbed path from $p'$ to $q'$ must be longer than the original. By somewhat involved calculus, we can quantify this statement and derive a precise characterization of balanced chains. (We will prove this formula in its full generality in Section 5.4.5.)

**Lemma 5.6 (Balanced Chains)** *In order for a chain $\pi'$ to be balanced with respect to a polygonal path $\pi$, the angle of turn $\phi$ at its endpoint $q$ must be in a unique relationship with its angle of turn $\gamma$ at $p$: $\phi = F(\gamma)$, where the function $F$ depends on $p'$, $q'$, the rocking points of $\pi'$, and the edges of $P$ containing the bash points of*

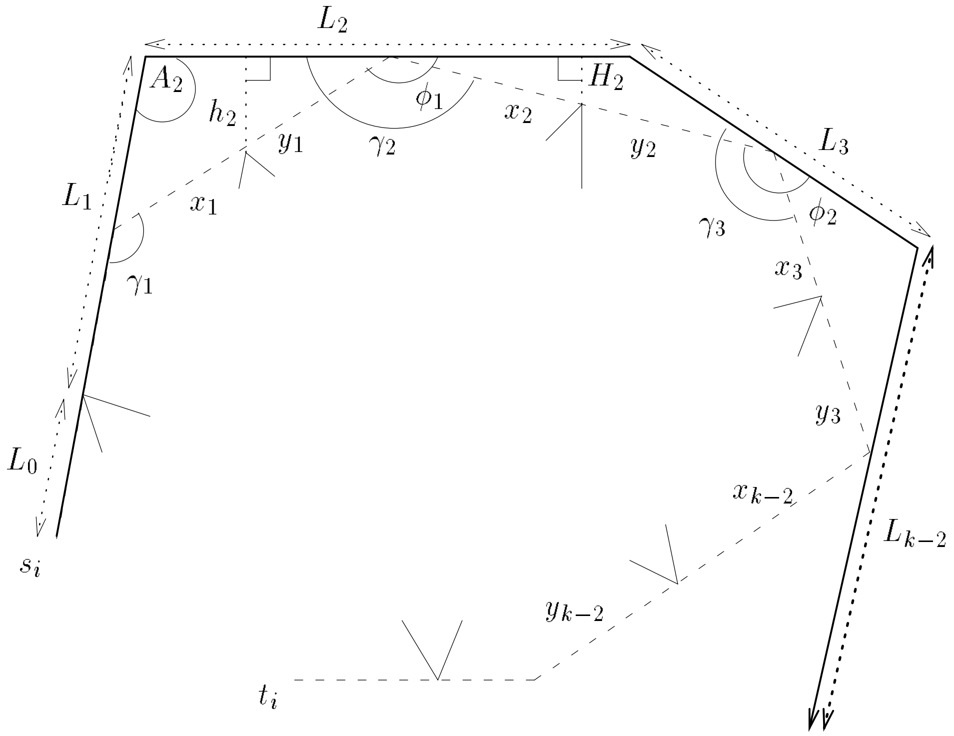



Figure 5.12: Balanced edges with bashes.

*$\pi'$. For the case of balanced edges, this relationship is expressed by the relationship*

$$x \cot \frac{\gamma}{2} = y \cot \frac{\phi}{2},$$

*where $x$ and $y$ are defined in Figure 5.11. More generally, the expression*

$$\sum_{i=1}^{k-2} \left[ \left[ y_i \cot \frac{\phi_i}{2} - x_i \cot \frac{\gamma_i}{2} \right] \prod_{j=2}^{i} \frac{h_j \csc^2(\gamma_{j-1} - A_j)}{H_j \csc^2 \gamma_j} \right]$$

*must equal zero, where the various constants are defined in Figure 5.12.*

For paths within a simple polygon $P$, we say that an edge in a polygonal $s$–$t$ path is *flush* if it contains a non-inflection edge of $\pi_G(s,t)$. (A similar definition applies to polygons with holes, provided we use a homotopically equivalent geodesic path.) The following uniqueness result will be a consequence of our various results on local optimality:

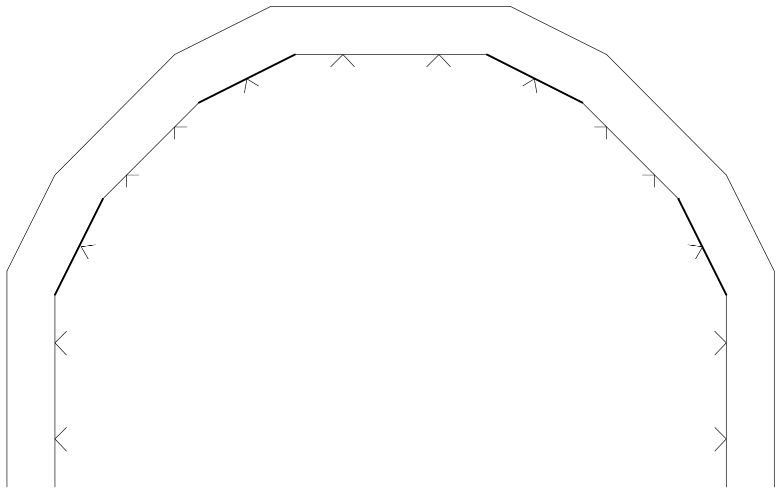


Figure 5.13: Exponential number of shortest k-link paths.

**Lemma 5.7** *Let $a$ and $b$ be consecutive flush edges along a shortest $k$-link path, $\pi^*$, and let $i$ denote the number of links along $\pi^*$ between $a$ and $b$. Then, the subpath of $\pi^*$ joining $a$ and $b$ is the unique shortest $i$-link path from $a$ to $b$.*

It is not the case that shortest $k$-link $s$–$t$ paths in a simple polygon are unique; indeed, we can give a construction to show that there can be an exponential number of equivalent (same Euclidean length) $k$-link paths. This example also demonstrates how flush edges may be part of the shortest $k$-link path.

To see this, consider the example given in Figure 5.13. The long edges are flush edges or extensions of flush edges within the raceway. Since they are very long straight connections, they would be used in a path having $k$ links (for large enough $k$) to keep the total number of links small. The short dark edges are possible small shortcuts or "toggles" where allowing an extra link would allow a small shortcut between two flush edge extensions. If the number of extra links allowed is fixed (say half the possible toggles) there are an exponential number of choices to toggle between the flush edges. (The figure can be made precisely symmetric so that any toggle shortcut will decrease the Euclidean length by exactly the same amount.) Thus, all of the possible combinations of a fixed number of shortcuts will give paths with equivalent link and Euclidean length. Thus there are an exponential number of equivalent shortest $k$-link paths in this polygon.

We will show (in Sections 5.4.1– 5.4.5) that in order for $\pi^*$ to be optimal, it must be locally optimal in the following sense:

**Lemma 5.8 (Local Optimality)** *If one perturbs any single edge $e$ of a shortest $k$-link path, $\pi^*$, the result is either to disconnect $\pi^*$ or to increase its length. In particular, $e$ must be in contact with at least one vertex of $P$ and $e$ must fall into one of the following cases:*

- *$e$ is an inflection edge of $\pi^*$, containing an inflection edge of $\pi_G(s,t)$; or*
- *$e$ is a flush edge, containing $\overline{uv} \subset \pi_G(s,t)$, and the balanced position of $e$ with respect to rocking point $u$ is clockwise from $e$, while the balanced position of $e$ with respect to rocking point $v$ is counterclockwise from $e$; or*
- *$e$ is a rocking edge with both of its endpoints being bash points; or*
- *$e$ is a rocking edge with both of its endpoints being interior bends, such that $e$ is balanced; or*
- *$e$ is a rocking edge with exactly one of its endpoints being a bash point — the balanced position of $e$ with respect to its rocking point is either clockwise or counterclockwise from $e$ depending on whether the bash endpoint is on $e$'s left or right end, respectively.*

Based on the above lemmas, we can now outline the general structure of an optimal path $\pi^*$. Path $\pi^*$ is partitioned by its inflection edges into pieces, $\pi_0^*, \pi_1^*, \ldots, \pi_{I+1}^*$, with piece $\pi_j^*$ joining inflection edge $e_j$ to $e_{j+1}$. Path $\pi_j^*$ is further partitioned according to the flush edges of $\pi^*$ into *elementary paths* joining pairs of consecutive flush or inflection edges. Each elementary path joining flush/inflection edge $a$ to flush/inflection edge $b$ has the following structure: From $a$, $\pi^*$ is "greedy" (following the bash points of $\pi_L(s,t)$) for some portion, until a *first interior bend point* $p_1$ (which may, in fact, occur on the first link of the elementary path — i.e., the greedy path may be empty). The path $\pi^*$ then consists alternately of chains of interior bend points (i.e., chains of balanced rocking edges) and balanced chains of bash points. See Figure 5.14 for an example of the structure of an optimal path.

The first interior bend point, $p_1$, will play a special role in our search algorithm. We will perform a binary search to identify $p_1$ approximately. In each step of the binary search, we will trace a path using the local optimality condition and check

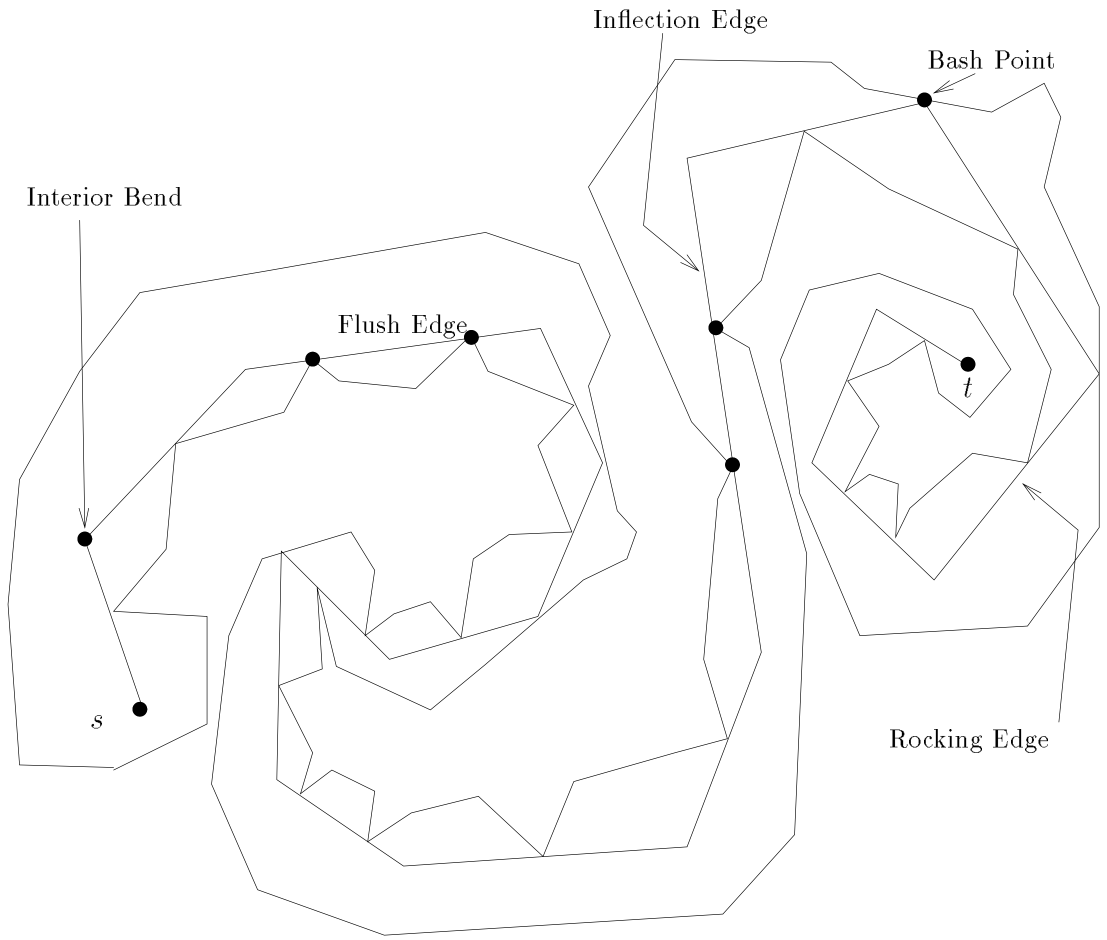



Figure 5.14: Structure of an optimal path.

if the path can hit our desired target in the required number of links. After each step, we narrow the possible positions of $p_1$ by half. Since the position of $p_1$ may be the root of a high degree polynomial we stop the search when the range is small enough. Note that not all points along $\pi_L(s,t)$ are possible first bend points for $\pi^*$. A first bend point $p_1 \in \pi_L(s,t)$ is said to be *valid* if there exists a shortest $i$-link path from $s$ to some $t'$ that uses $p_1$ as the first bend point. We can characterize the valid first bend points in the following way: let $e = \overline{qz}$ be an edge of $\pi_L(s,t)$, with rocking point $r$. Let $p_e$ be the point on $\overline{rz}$ (if it exists) such that $\overline{qp_e}$ is balanced with respect to the three-link path: predecessor of $e$, $\overline{qp_e}$, and the ray from $p_e$ through its (left) tangency point, $r_2 \in \pi_G(s,t)$. Refer to Figure 5.15.

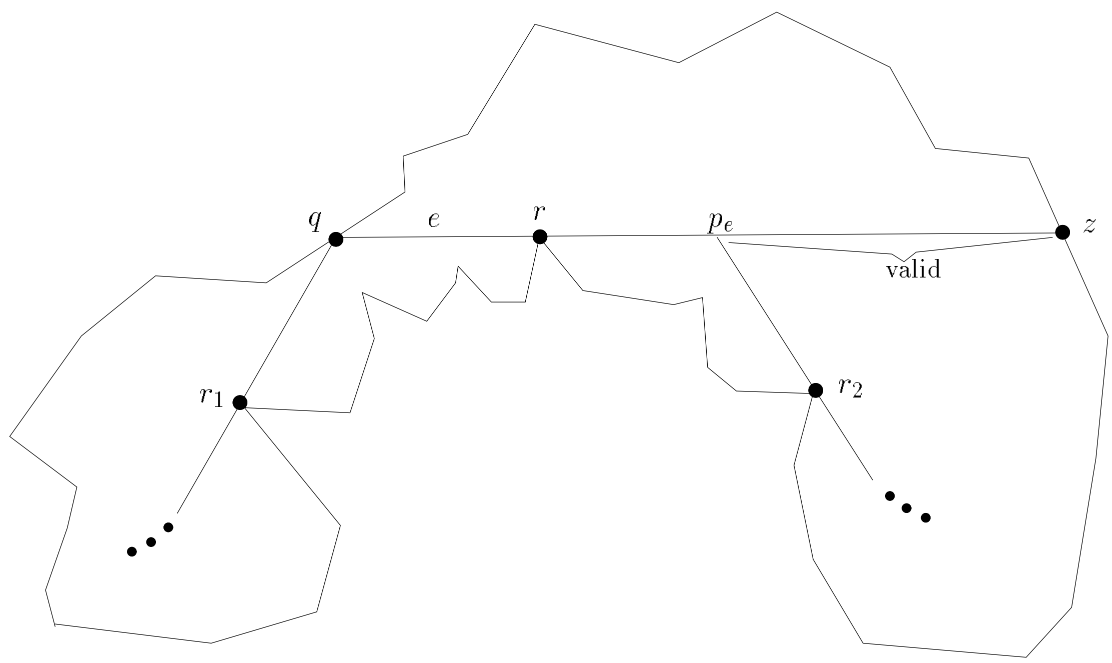



Figure 5.15: Valid first bend points along edge $e$.

**Lemma 5.9** *The valid first bend points for $e \in \pi_L(s,t)$ are precisely the points on the (possibly empty) segment $\overline{p_e z}$.*

**Proof.** Suppose we follow the greedy path for a few links and then use the first local optimality condition to find the next link. We assume that the last bash point $b$ is interior, and look for the next interior turn and uniquely defined next tangency point. Call that interior turn point $p_e$. If potential turn points were allowed to be before $p_e$, then the path would be able to twist counterclockwise to achieve a local optimum. Thus the points before $p_e$ cannot be part of a continuation of an optimal path. We have chosen $p_e$ so that for any point $x$ along $\overline{p_e z}$, the three links $\overline{r_1 q}$, $\overline{qx}$, $\overline{xr_2}$ will be at local optimality, because the middle link will either exactly satisfy the balance condition or the middle link will want to rock clockwise to advance further along the path but is prevented by the bash point $q$. Also, potential turn points before $p_e$ would not have these three links at local optimality and hence the middle link would want to twist counterclockwise to become optimal (i.e. $b$ would not have been the bash point).

If interior turns before $p_e$ were allowed, the path could be locally improved by rocking edge $e$. By disallowing invalid turns we know that we will be tracing only locally optimal paths. Preprocessing of the greedy minimum-link path to find the valid ranges of first bend points can be done in $O(n)$ time. □

We have now completed the characterization of a shortest $k$-link path. We are ready to describe an algorithm to approximate this path.

## 5.4 Computing Shortest $k$-Link Paths in Simple Polygons

The inflection edges of the unique geodesic path, $\pi_G(s,t)$, permit us to decompose the full problem into a sequence of "raceway" problems, where we must find (nearly) shortest $i$-link paths (for many values of $i$) between two consecutive inflection edges, while staying within a relevant subset of $P$. Within a raceway, though, there still may be an exponential number of shortest $i$-link paths; thus, we further decompose the problem into "elementary" problems of finding shortest $i$-link paths between pairs of flush edges (edges that contain an edge of $\pi_G(s,t)$). Each of these problems is solved by a binary search for the first bend point in an optimal path: each test of the search involves tracing a path forward through the raceway, applying the local optimality conditions of Section 5.4.5 at each step (several lemmas are needed to justify this). The justification of the search itself is based on a key "Monotonicity Lemma" (see Lemma 5.15) which shows that paths traced forwards in this way behave "nicely" with respect to changes in the first bend point. The search stops when every edge of the traced path is known to lie within a very tiny range of angles, of size at most $\delta(\epsilon)$. This property is guaranteed if we make the interval of first bend angles small enough — less than $\delta(\epsilon)/D^k$, where $D$ is a "dilation" factor that measures the maximum amount by which a small interval of directions trapping link $i$ can get larger for link $i+1$. We will obtain a bound on the dilation $D$ by bounding the change in the $i$th turn angle with respect to the change of the first angle (the $i$th angle is determined by the first angle using the local optimality condition). Our bounds and analysis assume that the vertices of $P$ are on an integer grid of size at most $N$-by-$N$; this allows us to write lower bounds on angles between triples of vertices (refer to Section 2.3.3 for more detail about geometry on a grid). Finally we assemble all of the data from the binary

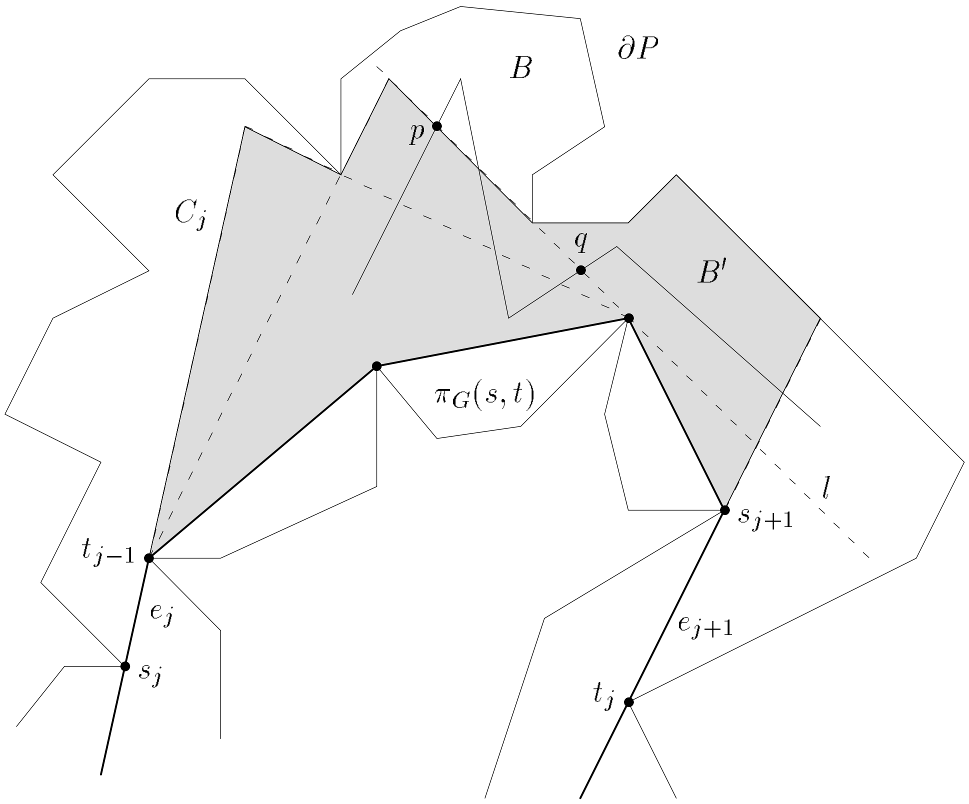



Figure 5.16: The raceway $R_j$ between inflection edges $e_j$ and $e_{j+1}$.

searches, combining it via dynamic programming recurrences to obtain the final result: an $\epsilon$-optimal $k$-link path from $s$ to $t$.

### 5.4.1 Decomposing the Problem

Using Lemma 5.4, we decompose our problem into a sequence of "raceways" between extensions of inflection edges. Raceway $R_j$ is the portion of $P$ that lies between inflection edges $e_j$ and $e_{j+1}$. Refer to Figure 5.16. We further trim from $R_j$ all of the pockets of $P$ that are on the convex side of the subpath $\pi_G(s_j, t_j)$. By an argument similar to that of Lemma 5.4, we can say that the shortest $k$-link path will never go into a bay below the convex chain $\pi_j$ which is between of inflection edges $e_j$ and $e_{j+1}$. Thus we can replace the portion of the boundary of $P$ between $t_{j-1}$ and $s_{j+1}$ by $\pi_j$.

We can also replace the portion of the boundary of $P$ between $s_j$ and $t_j$ (going clockwise) by the *complete visibility chain*, $C_j$, which is the boundary of all points $p$ such that $p$ sees both of its tangent points with $\pi_j$. ($C_j$ can be found in linear time [Gho91].) We show that $\pi^*$ must lie within this raceway $R_j$, which is bounded by $C_j$, the inflection edges $e_j$ and $e_{j+1}$, and the path $\pi_j$. The proof of this lemma again is very similar to the inflection edge lemma, Lemma 5.4. (A similar lemma was also given by Ghosh [Gho91].)

**Lemma 5.10** *A shortest $k$-link path will lie inside the raceway $R_j$.*

**Proof.** Refer to Figure 5.16. Suppose $\pi^*$ goes into some bay $B$ that is not part of the complete visibility region. This implies that $\pi^*$ crosses some edge of the complete visibility chain $C_j$. Let $l$ be the extension of that edge in both directions. We will assume that $l$ is clockwise-pinned. (A similar argument can be made for the case when $l$ is counterclockwise-pinned.) It is clear that the line $l$ splits the polygon into four pieces. Label as $B'$ the region that is not $B$, and that does not contain $s_j$ or $t_j$. It is clear that $\pi^*$ must enter $B'$ to reach $e_{j+1}$. Let $p$ be the first point where $\pi^*$ crosses $l$ to enter the bay $B$ and let $q$ be the last point where $\pi^*$ crosses $l$ to enter bay $B'$. By shortcutting along $\overline{pq}$ one would obtain a shorter path with the same number links, thus $\pi^*$ could not have been optimal. □

We note one important difference between this lemma and the lemma for inflection edges (Lemma 5.4). The inflection edges separate the start point from the goal point and hence *must* be followed by an optimal path. The edges of the complete visibility chain $C_j$ will not always separate the start from the goal and hence the edges of the chain may or may not be used in a shortest $k$-link path.

Using Lemma 5.10 we can now decompose our problem into finding shortest $k_j$-link paths between $s_j$ and $t_j$ in a raceway $R_j$, where $k_j$ ranges from $d_L(s_j, t_j)$ up to $|\pi_G(s_j, t_j)|$. We will then be able to use a dynamic programming approach to join raceway paths and find a shortest $k$-link path from $s$ to $t$.

### 5.4.2 Shortest $k$-link Path in a Raceway

Let us fix our attention on one such raceway, $R$, as described in the previous section. We would like to find shortest $i$-link paths within $R$, for a variety of values $i$ (at most $k$ such values). As stated in Section 5.3, a shortest $i$-link path may have many flush edges along it, but between each pair of consecutive flush edges, we will show the path is an alternating sequence of chains of bash points (greedy paths) and interior-bend paths.

We must introduce $O(n)$ additional special edges that further decompose the raceway problem. We call these edges *leaning* edges, and they are defined to be edges $\overline{uv}$ of $VG(P)$ such that one endpoint $u$ is a vertex of $\pi_G(s,t)$, while $v$ is a vertex of $P$ such that $\overline{uv}$ is an edge of *right tangency* from $v$, with respect to the chain $\pi_G(s,t)$ within $R$. We will show later that leaning edges can "split" our "wedge" of traced paths, causing the angular ranges of each link to amplify in an unbounded fashion, (preventing us from obtaining a bounded "dilation" factor). Thus, we will need to further subdivide the raceway problem according to the $O(n)$ leaning edges. We are able to show that the resulting elementary problems from $a$ to $b$ (where $a$ and $b$ are now either inflection, flush, or leaning edges) permit a provably good approximation to the optimal path.

**Lemma 5.11** *There can be at most $O(n)$ leaning edges in our path.*

**Proof.** A leaning edge actually corresponds to an edge of the complete visibility chain in the raceway. Refer to Figures 5.16 and 5.27. By Ghosh's results [Gho91], the complete visibility chain is constructed using the vertices of the inner illuminating convex chain. This is because the extension edges of the complete visibility chain must be tangent to the inner convex polygon. Thus, each vertex can contribute to at most two leaning edges, and therefore there are at most $O(n)$ leaning edges on an optimal path. □

We define the *combinatorial type* of a path to be the sequence of labels, one per bend point, listing the bash wall for each bashing bend point, or noting that a bend point is interior. (Note that we do *not* include the set of rocking points in this definition of combinatorial type.) Leaning edges correspond, then, to edges of $VG(P)$ that can account for a change in the combinatorial type of our traced paths.

### 5.4.3 No Bash Points Case ($\mathrm{LOPT}_1$)

We first derive the local optimality condition for the case when our path consists of interior turns only. Within a particular raceway we know the first and last inflection edges. For the moment let us assume that we are trying to fit one link between the

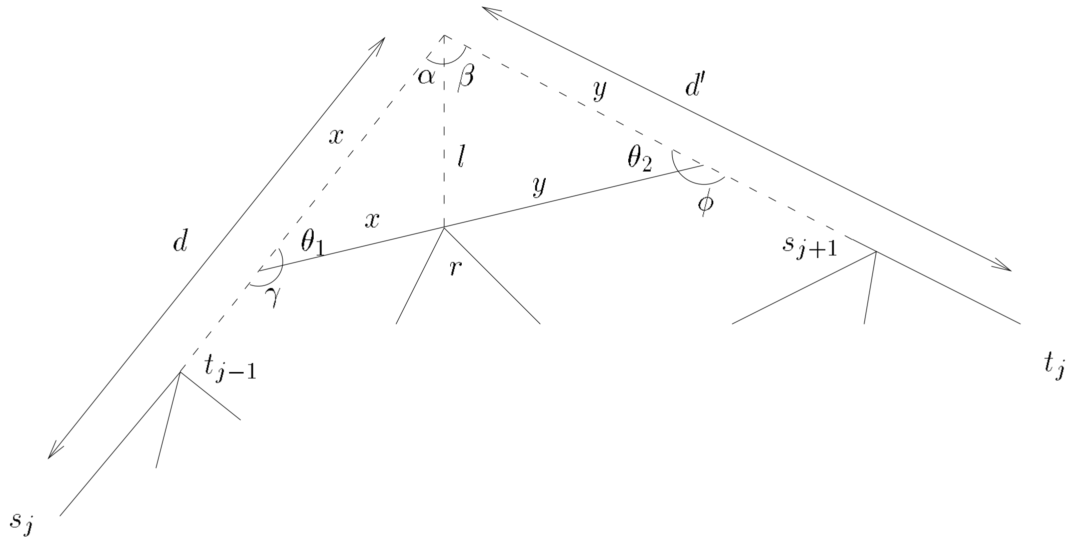



Figure 5.17: Local optimality of an interior turn.

two inflection edges, and we want this one link to balance on a particular rocking point. Thus, we know $s_j$, $t_{j-1}$, $s_{j+1}$, $t_j$, and a rocking point $r$. (See Figure 5.17.)

We would like to write the length from $s_j$ to $t_j$ as a function of $\theta_1$.

$$l(\theta_1) = (d - x) + (d' - y) + \frac{l \sin\alpha}{\sin\theta_1} + \frac{l \sin\beta}{\sin\theta_2}.$$

Since $x = l\sin(\alpha + \theta_1)/\sin\theta_1$, and $y = l\sin(\alpha + \theta_1)/\sin\theta_2$,

$$l(\theta_1) = d + d' - \frac{l\sin(\alpha + \theta_1)}{\sin\theta_1} - \frac{l\sin(\alpha + \theta_1)}{\sin\theta_2} + \frac{l\sin\alpha}{\sin\theta_1} + \frac{l\sin\beta}{\sin\theta_2}.$$

Since $\theta_2 = \pi - \theta_1 - \alpha - \beta$,

$$l(\theta_1) = d + d' + \frac{h(\sin\alpha - \sin(\alpha + \theta_1))}{\sin\theta_1} + \frac{h(\sin\beta - \sin(\alpha + \theta_1))}{\sin(\theta_1 + \alpha + \beta)}.$$

We differentiate and set $l'(\theta_1) = 0$ to find the value of $\theta_1$ that will give the shortest 3-link path:

$$\begin{aligned}
0 &= \frac{\sin\theta_1(-\cos(\alpha + \theta_1)) - (\sin\alpha - \sin(\alpha + \theta_1))\cos\theta_1}{\sin^2\theta_1} + \\
&\quad \frac{\sin(\theta_1 + \alpha + \beta)(-cos(\theta_1 + \alpha)) - (\sin\beta - \sin(\theta_1 + \alpha))cos(\theta_1 + \alpha + \beta)}{\sin^2(\theta_1 + \alpha + \beta)} \\
0 &= \frac{\sin(\alpha + \theta_1 - \theta_1) - \sin\alpha\cos\theta_1}{\sin^2\theta_1} - \frac{\sin\beta + \sin\beta\cos(\theta_1 + \alpha + \beta)}{\sin^2(\theta_1 + \alpha + \beta)}
\end{aligned}$$

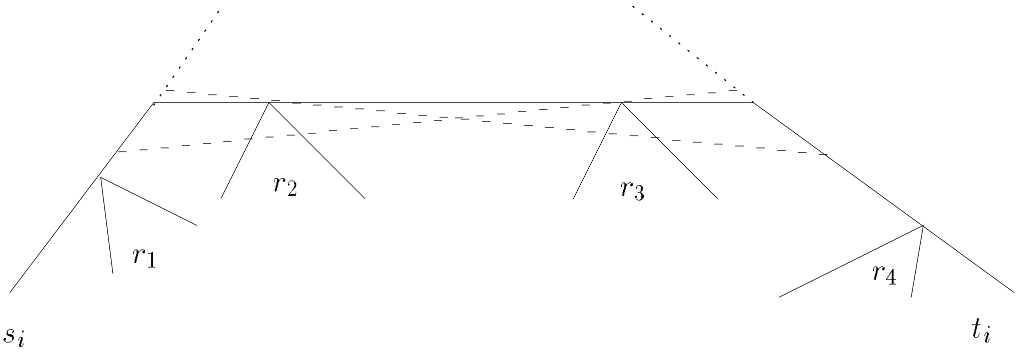



Figure 5.18: Flush edges may be locally optimal

From this we obtain the local optimality condition

$$\frac{\sin\alpha}{(1+\cos\theta_1)} = \frac{\sin\beta}{(1+\cos\theta_2)}.$$

Using trigonometric identities, this formula can also be written as

$$y\cot\frac{\phi}{2} = x\cot\frac{\gamma}{2}.$$

In our search procedure, we will want to use this formula in reverse. We will know our initial inflection edge, $s_j$, $t_{j-1}$, our current choice of a turn point, which implies an initial turn angle $\gamma$ and the first rocking point $r$. We will be testing a new rocking point $s_{j+1}$. So will we need to express $\phi$ as a function of the these values we know. In order for our search procedure to work we will later show in Section 5.4.6 that $\phi$ will be uniquely determined by $s_j$, $t_{j-1}$, $r$, $\gamma$, and $s_{j+1}$.

Later, we will also show that a monotonicity property holds (see Lemma 5.15). We would like to show that as we move the first bend point up $\overline{s_j t_{j-1}}$ the locally optimal path we will trace will reach "further along" the raceway. This monotonicity property allows us to do binary search for the first bend point.

### 5.4.4 Flush edges

We can now understand why flush edges may be locally optimal. (See Figure 5.18.) In this case the middle link would "rock" clockwise if it were balancing on $r_2$ but it is prevented from coming to rest at its balancing point by the point $r_3$ on the geodesic path. Similarly, the middle link would like to "rock" counterclockwise if it were on $r_3$ but is prevented by point $r_2$. Thus in this case the *flush* edge passing through $r_2$ and $r_3$ is at local optimality because it is at rest and no local improvement can be made.

Such flush edges can occur in a shortest $k$-link path from $s$ to $t$, for example in a long skinny corridor. If the number of flush edges is large enough this degenerates to the case where the path is all the flush edges, i.e. the geodesic path. We take flush edges into account by adding them to our dynamic programming algorithm.

### 5.4.5 Bash Points Case ($\mathrm{LOPT}_2$)

We can derive a similar optimality condition for the case when the locally optimal path must hit the boundary of the bounding polygon. As we attempt to trace out a locally optimal path we may want to make a turn that is prevented because the boundary is in the way. In this case we clip the path at the boundary, take the first greedy link from that point on the boundary to find the next rocking point, and use a "1-bash point" local optimality criterion to find the next optimal interior turn. This turn may also be prevented by the boundary, in which case we must continue to trace out a path using a multiple bash-point local optimality criteria.

In this case the length of the path with bashes is $l_1 + l_2 + \cdots + l_{k-2} - L_2 - L_3 - \cdots - L_{k-2}$, where $l_1$ is the length of a path from $s_i$ along $L_1$, the first rocking link and then *following the path along* $L_2$ (refer to Figure 5.12). Similarly, $l_2$ is the length of the path starting along $L_2$, taking the middle link and then travelling along $L_3$. Thus by adding these path lengths together and subtracting away the constant parts we can find an expression for the rocking edges.

If we perturb the path slightly by rotating the first (non-fixed) link counterclockwise by an angle $\delta_1$, we can express the change in the length of the path as a function of $\delta_1$.

$$
\begin{aligned}
\frac{dl}{\delta_1} &= \frac{dl_1}{\delta_1} + \frac{dl_2}{\delta_2} + \cdots + \frac{dl_{k-2}}{\delta_{k-2}} \\
\frac{dl}{\delta_1} &= (y_1 \cot\frac{\phi_1}{2} - x_1 \cot\frac{\gamma_1}{2}) + \frac{\delta_1}{\delta_2}(y_2 \cot\frac{\phi_2}{2} - x_2 \cot\frac{\gamma_2}{2}) + \cdots \\
&\quad + \frac{\delta_{k-1}}{\delta_1}(y_{k-2} \cot\frac{\phi_{k-2}}{2} - x_{k-2} \cot\frac{\gamma_{k-2}}{2})
\end{aligned}
$$

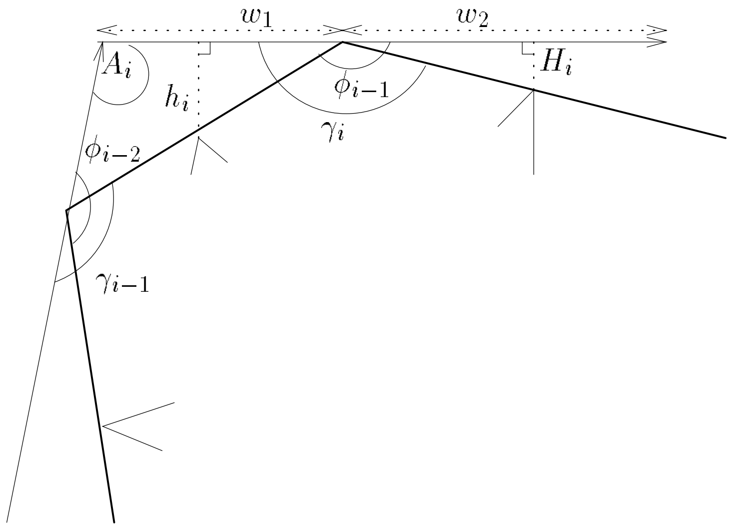



Figure 5.19: How $\gamma_i$ changes as $\gamma_{i-1}$ changes.

The path is locally optimal when $dl/\delta_1 = 0$. Since $\delta_{i+1}/\delta_i = d\gamma_{i+1}/d\gamma_i$, this means:

$$\begin{aligned}
\frac{dl}{d\gamma_1} &= \frac{dl_1}{d\gamma_1} + \frac{dl_2}{d\gamma_2} \cdot \frac{d\gamma_2}{d\gamma_1} + \cdots + \left(\frac{dl_{k-2}}{d\gamma_{k-2}} \frac{d\gamma_{k-2}}{d\gamma_{k-3}} \cdots \frac{d\gamma_2}{d\gamma_1}\right) \\
&= \left(y_1 \cot \frac{\phi_1}{2} - x_1 \cot \frac{\gamma_1}{2}\right) + \cdots \\
&\quad + \left(y_{k-2} \cot \frac{\phi_{k-2}}{2} - x_{k-2} \cot \frac{\gamma_{k-2}}{2}\right) \frac{d\gamma_{k-2}}{d\gamma_{k-3}} \cdots \frac{d\gamma_2}{d\gamma_1}
\end{aligned}$$

We would like a formula for $d\gamma_i/d\gamma_{i-1}$. We can derive one using implicit derivation. Refer to Figure 5.19 to see pictures of the constants used here. Note that the angles $A_i$ and the lengths $w_1$, $w_2$, $h_i$ and $H_i$ are fixed, given the specification of polygon $P$. Also, note that $\phi_{i-1} = \pi - \gamma_i + A_i$. Further, note that $w_1 = h_i \cot(\gamma_{i-1} - A_i)$ and $w_2 = H_i \cot(\pi - \gamma_i) = -h_2 \cot \gamma_i$.

Let $W = w_1 + w_2$. (Note that $W$ is fixed). Then we can use implicit derivation as follows:

$$W + H_i \cot \gamma_i = h_i \cot(\gamma_{i-1} - A_i)$$

Taking the derivative of both sides with respect to $\gamma_{i-1}$ we obtain

$$-H_i \csc^2 \gamma_i \frac{d\gamma_i}{d\gamma_{i-1}} = -h_i \csc^2(\gamma_{i-1} - A_i).$$

Thus, we see that for $i = 2, 3, \cdots, k-2$,

$$\frac{d\gamma_i}{d\gamma_{i-1}} = \frac{h_i \csc^2(\gamma_{i-1} - A_i)}{H_i \csc^2 \gamma_i}.$$

Combining these we have shown the Balanced Chain Lemma 5.8, which says that the path is locally optimal when

$$\sum_{i=1}^{k-2} \left[ \left[ y_i \cot \frac{\phi_i}{2} - x_i \cot \frac{\gamma_i}{2} \right] \prod_{j=2}^{i} \frac{h_j \csc^2(\gamma_{j-1} - A_j)}{H_j \csc^2 \gamma_j} \right] = 0.$$

### 5.4.6 Uniqueness

In this section we will prove two facts about the above Local Optimality formula. First, we will show the formula above has a unique solution, so we can to invert it. We will want to use the formula above as a method of tracing a path for a particular initial choice for the turn angle. Given an initial turn angle we will want determine the next rocking point that obeys the local optimality condition.

We prove that for a particular rocking point $r$ the formula is uniquely invertible. Second, we show that there is a unique rocking point along the inner (convex) geodesic path of the convex raceway such that the formula traces a path *tangent* to the raceway. This will allow us to *test* the vertices of $\pi_G$ to find the next rocking point that obeys local optimality.

Suppose we want to know $x$ and $\theta$, if $x \cot(\theta/2) = c_1$, where $c_1$ is a fixed constant. (See Figure 5.20.) We know the rocking point $r$. Assume it has coordinates $(x_0, y_0)$. Using straightforward algebra and trigonometry we will show that the solution for $x$ and $\theta$ is unique.

**Lemma 5.12** *The formula $x \cot(\theta/2) = c_1$ can be solved uniquely for a fixed rocking point $r$ (refer to Figure 5.20).*

**Proof.** From the geometry of the problem, we know

$$\begin{aligned} x \cot(\theta/2) &= x \frac{(1 + \cos \theta)}{\sin \theta} \\ &= x \left( \frac{1}{\sin \theta} + \frac{\cos \theta}{\sin \theta} \right) \\ &= x \left( \frac{\sqrt{(x - x_0)^2 + {y_0}^2}}{y_0} + \frac{(x - x_0)}{y_0} \right) \end{aligned}$$

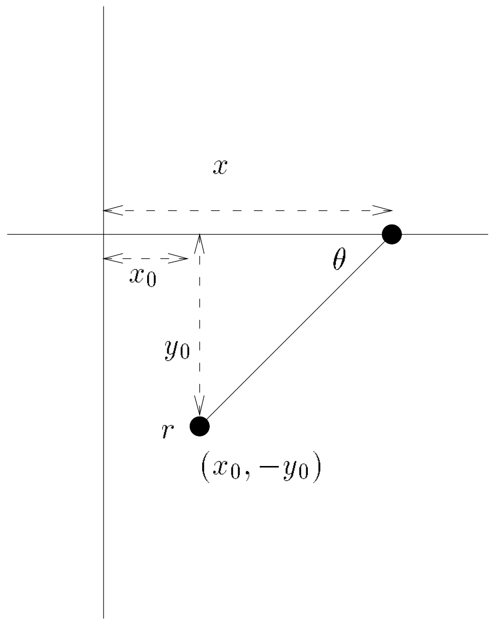



Figure 5.20: Uniqueness of the solution for a fixed rocking point.

$$
\begin{aligned}
x\left(\sqrt{(x-x_0)^2+{y_0}^2}+(x-x_0)\right) &= c_1y_0 \\
x\sqrt{(x-x_0)^2+{y_0}^2}+x^2-x_0x &= c_1y_0 \\
x\sqrt{(x-x_0)^2+{y_0}^2} &= -x^2+x_0x+c_1y_0 \\
x^2\left((x-x_0)^2+{y_0}^2\right) &= \left(-x^2+x_0x+c_1y_0\right)^2 \\
x^2\left(x^2-2xx_0-{x_0}^2+{y_0}^2\right) &= \left(-x^2+x_0x+c_1y_0\right)\left(-x^2+x_0x+c_1y_0\right) \\
&= x^4-x_0x^3-c_1y_0x^2-x_0x^3+{x_0}^2x^2 \\
&\quad +c_1x_0y_0x-c_1y_0x^2+c_1x_0y_0x+{c_1}^2{y_0}^2 \\
x^4-2x_0x^3+{x_0}^2x^2+{y_0}^2x^2 &= x^4-2x_0x^3-2c_1y_0x^2 \\
&\quad +x_0x^2+2c_1x_0y_0x+{c_1}^2{y_0}^2.
\end{aligned}
$$

By cancelling and regrouping terms we find

$$
\begin{aligned}
-2c_1y_0x^2-{y_0}^2x^2+2c_1x_0y_0x+{c_1}^2{y_0}^2 &= 0 \\
2c_1x^2+y_0x^2-2c_1x_0x-{c_1}^2y_0 &= 0 \\
(2c_1+y_0)x^2-(2c_1x_0)x-{c_1}^2y_0 &= 0,
\end{aligned}
$$

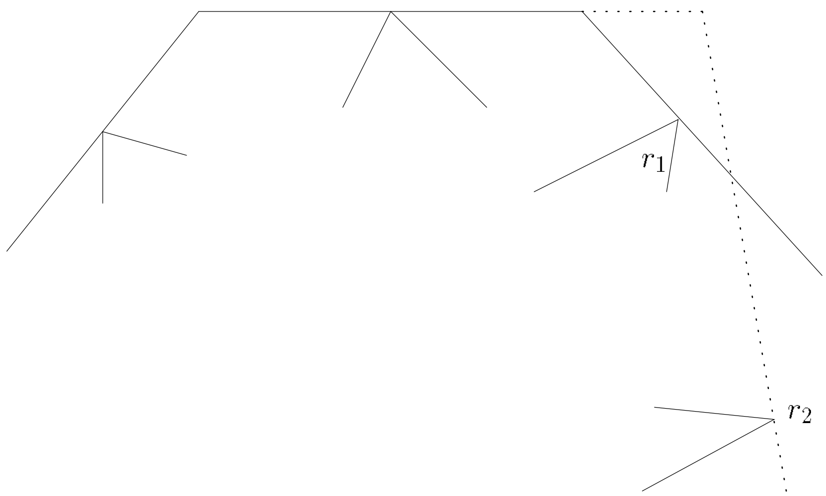



Figure 5.21: Can locally optimal paths cross?

and by applying the quadratic formula we find that

$$\begin{aligned} x &= \frac{2c_1 x_0 + \sqrt{4{c_1}^2 {x_0}^2 + 4(2c_1 + y_0)({c_1}^2 y_0)}}{2(2c_1 + y_0)} \\ &= \frac{x_0 + \sqrt{{x_0}^2 + y_0^2 + 2c_1 y_0}}{2 + y_0/c_1} \end{aligned}$$

Note that the discriminant is always greater than zero so a solution always exists. Also, we choose only the positive solution since the origin $(0,0)$ represents the rocking point the middle link is leaning on, so negative values of $x$ do not make sense. $\square$

When we solve for $x$ and $\theta$ for a particular rocking point, we also check if the solution is *tangent* to the geodesic path at this point. We know there is a unique solution for a particular rocking point, but now we must also show that there is a unique rocking point when the solution will become tangent to the inner geodesic path. We want to show that the solutions will not "cross" as in Figure 5.21.

**Lemma 5.13** *There is a unique rocking point $r$ such that the (unique) solution to the formula $x\cot(\theta/2) = c_1$ for rocking point $r$ will be tangent to the inner geodesic path.*

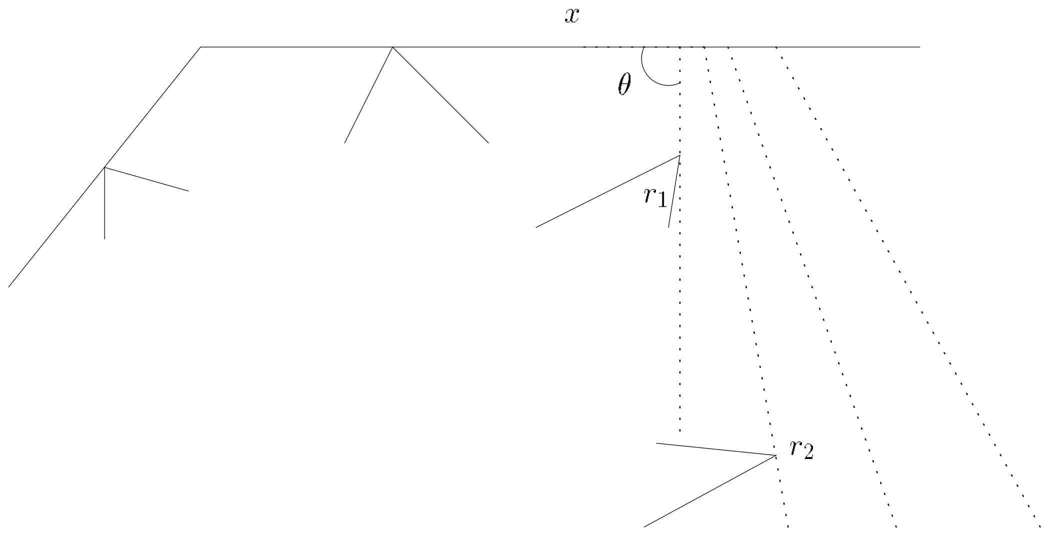



Figure 5.22: Locally optimal paths “fan-out”, so only one can be tangent.

**Proof.** The fact that the solutions will not cross comes from a straightforward examination of the local optimality condition. We know that $x\cot(\theta/2) = c_1$ where $c_1$ is a constant that we can determine based on path we have traced so far. Let us examine how $\theta$ varies as $x$ increases (refer to Figure 5.22). As $x$ increases, $\cot(\theta/2)$ must decrease. Cotangent is a monotonically decreasing function on the interval 0 to $2\pi$. Thus, if $\cot(\theta/2)$ is decreasing, $\theta$ must be increasing and the locally optimal path must “fan-out” as illustrated. Thus, there will be at most one rocking point such that the locally optimal path is tangent to the inner geodesic path at that point. $\square$

Using these two lemmas allows us to uniquely “trace” paths for a particular initial turn angle choice.

### 5.4.7 Tracing Paths

Given a *valid* first bend point, $p_1$, we can uniquely trace out a shortest path consistent with $p_1$ from flush edge $a$ to flush edge $b$, using the local optimality conditions. Let $\pi(\gamma, i)$ denote the resulting path, where $\gamma$ is the turn angle associated with point $p_1$.

The crucial lemma which allows us to do tracing is the following (note that this lemma applies only to elementary paths between flush edges):

**Lemma 5.14 (Tracing Lemma)** *Given a first bend point $p_1$, the next link of $\pi(\gamma, i)$ is uniquely determined (it is simply obtained by "leaning" an edge through $p_1$ against path $\pi_G(s,t)$). Given a subpath of $\pi(\gamma, i)$ traced some number of links beyond $p_1$, the next link of $\pi(\gamma, i)$ is* uniquely *determined by the local optimality conditions and the subpath so far. The next rocking point can therefore be determined by advancing along $\pi_G(s,t)$, one vertex at a time, starting from the last rocking point of the current subpath.*

**Proof.**

This lemma follows from Lemma 5.12 and Lemma 5.13 above which imply that we can trace paths uniquely. Recall that an elementary path may consist of a greedy link, an interior turn, bashes, another interior turn, more bashes, etc. In order to trace a path, we first choose an initial an interior turn from our range of valid first turn points, and then use LOPT1 to determine where we should bend next. At any stage, we may have a long sequence of bash points as the last several turns in the current subpath. Preceding the sequence of bash points is an interior bend point (e.g., our first choice of an interior turn point $p_1$); thus, we are able to use the balanced-chain condition to determine where the next interior bend should be (if it is feasible). We will call this procedure *tracing* a path.

In order to apply the local optimality formula we need to know the next rocking point. We advance along $\pi_G(s,t)$ to find the next rocking point, *assuming that the next bend point is an interior turn.* If the new bend point is feasible, we find the tangency point between the bend point and the inner part of the convex chain ("go greedy from the bend point") and use the previous link and this angle to start tracing using LOPT1 again.

If we solve for the unique next "interior" bend point and find that it is *not* reachable while staying interior to $P$, then the traced path should have a bash point on the end of the current link, so we "bash" and continue. If the path bashes into the outer polygon we will use LOPT2 to determine the next interior turn (this may also bash, in which case we must continue using LOPT2 until we get to a point where we make another interior turn). When the interior turn for LOPT2 is feasible we "go greedy from the bend point" and use the last link and new angle to start tracing LOPT1 again. Thus we can forget the bashes we made so far.

Why is "bashing" the best thing to do? Suppose you did not bash, and some internal point along the segment was an interior turn point. The link continuing from this interior turn point would rest against the geodesic path. The three links up to this point (previous link, segment, new link) are not locally optimal, because there was no valid interior turn point. Thus the middle segment would not be locally balanced and it would want to "rock".

We know that $\pi(\gamma, i)$ will be locally optimal; in fact, it will give exactly the shortest $i$-link path to points beyond the $i$th rocking point (for the given first bend point $p_1$). This follows from using the local optimality criteria to trace the path, choosing $p_1$ from the valid set of candidate first interior bend points, and our uniqueness results. □

### 5.4.8 The Main Search Algorithm

Let $a$ and $b$ each be a flush, leaning, or inflection edge, where $b$ follows $a$ along $\pi_G(s,t)$. Let $s_a$ be the first vertex of $a$ on $\pi_G(s,t)$ and $r_a$ the succeeding one. Let $r_b$ be the first vertex of $b$ on $\pi_G(s,t)$ and $t_b$ the succeeding one. We now describe the main algorithm $\mathit{Path}(a,b,i)$ to find an $\epsilon$-optimal approximation to the shortest $i$-link path from $a$ and $b$ that does not use an intermediate flush edge. It uses the subprocedure given in the previous section that traces paths according to the local optimality conditions.

Let $\Gamma_0$ be the set $\bigcup_{e\in\pi_L}\Gamma_e$, where $\Gamma_e$ is the subinterval $[\gamma_e, \gamma'_e]$ of valid angles $\gamma$ that correspond to valid first interior bend points $p_1 \in e$. Let $\pi(\gamma, i)$ for $\gamma \in \Gamma_0$ be the resulting traced path that goes at most $i$ links, or until the first rocking point that is beyond $r_b$ on $\pi_G(s,t)$.

A point $p_1$ of $\Gamma_0$ is identified with the angle $\gamma$ between $\pi_L(a,b)$ and the ray from $p_1$ that is tangent to $\pi_G(s,t)$. Note that the total length of $\Gamma_0$, $|\Gamma_0|$, is bounded by $\sum_{e\in\pi_L}|\Gamma_e| = \sum_e(\gamma'_e - \gamma_e) \le \pi \cdot d_L(a,b)$.

During this procedure we will tabulate $g_i(a,b)$, an $\epsilon$-optimal approximation to the shortest $i$-link elementary path from $a$ and $b$ (where $a$ and $b$ are flush or leaning edges in the same raceway).

*Path(a, b, i):*

0. *(Create initial interval.)* $\Gamma \leftarrow \Gamma_0$

1. *(Check interval size.)* If $|\Gamma| \leq \delta_0 = \delta(\epsilon)/D^k$, go to Step 3.

2. *(Trace locally optimal path.)*

   Otherwise, $\gamma \leftarrow$ midpoint of $\Gamma$ and $\Pi \leftarrow \pi(\gamma, i)$.
   Let $r = i$th rocking point of $\Pi$.

3. *(Choose new, smaller interval.)*

   If $\Pi$ has fewer than $i$ links (i.e. we overshot $b$),
   then $\Gamma \leftarrow$ left half of $\Gamma$ and go to Step 1.
   Otherwise, if $\Pi$ has $i$ links, and $r = r_b$, then if $p$ is to the left of $\overline{t_b r_b}$
   then $\Gamma \leftarrow$ right half of $\Gamma$ and go to Step 1,
   else if $p$ is to the right of $\overline{t_b r_b}$ (which can only happen if $b$ is a leaning edge)
   then $\Gamma \leftarrow$ left half of $\Gamma$ and go to Step 1.
   Otherwise, if $r$ precedes $r_b$ along $\pi_L(a, b)$,
   then $\Gamma \leftarrow$ right half of $\Gamma$ and go to Step 1.

4. *(Verify path validity.)*

   If the *combinatorial type* of $\pi(\gamma, i)$ for $\gamma \in \Gamma = [\gamma_{\min}, \gamma_{\max}]$ is constant, then $g_i(a, b)$ is assigned the length of $\pi(\gamma_{\max}, i)$.
   Otherwise $g_i(a, b) \leftarrow \infty$, since the $\epsilon$-optimal path from $a$ to $b$ that we will produce will pass through some leaning edge $c$, and so will be found as $g_{i'}(a, c) + g_{i-i'-1}(c, b)$ for some $i'$.

### 5.4.9 Correctness of the Search

Correctness of the binary search procedure described above is based on the following key lemma, which is used to justify discarding half of the interval $\Gamma$ in Step 2:

**Lemma 5.15 (Monotonicity)** *Consider a particular raceway problem. Let $C$ denote the corresponding complete visibility profile, and let $\Gamma_0$ denote the set of valid first turn angles. For $1 < i \leq k$, let $z_i(\gamma)$ denote the point on $C$ obtained by extending the ith link of $\pi(\gamma, k)$ until it exits the raceway. Let $\pi(\gamma, i)$ be the subpath of $\pi(\gamma, k)$ up to and including link $i$. Then, if $\gamma, \gamma' \in \Gamma_0$, with $\gamma < \gamma'$, then $z_i(\gamma)$ precedes $z_i(\gamma')$ along the chain $C$.*

**Proof.** The proof is by contradiction, and uses an inductive argument based on the number of links in the two paths.

Note that $\pi(\gamma, i)$ gives an (exact, unique) shortest $i$-link path (that uses no flush or leaning edge) to any point of $\pi(\gamma, i)$ that lies on the $i$th link of $\pi(\gamma, i)$, *after* the $i$th rocking point of the path. This is a direct result of the way we trace a path following the local optimality conditions.

For this lemma we will use the geometric observation about intersections of cones proven in Lemma 2.6. With this geometric observation we can prove the desired monotonicity lemma.

Suppose $i = 2$. Then obviously $z_2(\gamma)$ precedes $z_2(\gamma')$, since we lean the first edge along the inner geodesic path in a greedy manner.

We can now use the induction hypothesis. Suppose we have shown that $z_i(\gamma)$ precedes $z_i(\gamma')$ along the chain $C$. We want to show that $z_{i+1}(\gamma) < z_{i+1}(\gamma')$. Assume that this is *not* true, that is, assume that $z_{i+1}(\gamma') < z_{i+1}(\gamma)$. There are two cases, and in either case we will get a contradiction.

First, suppose the order along $C$ is: $z_i(\gamma) < z_{i+1}(\gamma') < z_{i+1}(\gamma) < z_i(\gamma')$. See Figure 5.23. We immediately get a contradiction, since this implies $z_{i+1}(\gamma') < z_i(\gamma')$, which is false since link $i + 1$ will advance past link $i$. For the same reason, the order along $C$ cannot be $z_i(\gamma) < z_{i+1}(\gamma') < z_i(\gamma') < z_{i+1}(\gamma)$.

Second, suppose if the order along the chain $C$ is $z_i(\gamma) < z_i(\gamma') < z_{i+1}(\gamma') < z_{i+1}(\gamma)$, we also get a contradiction. First note that since $z_{i+1}(\gamma')$ is one link away from $\pi(\gamma, i)$, $z_i(\gamma')$ and $z_{i+1}(\gamma')$ are also one link away from $\pi(\gamma, i)$. (So the figure we draw is the generic picture.) See Figure 5.24 for an illustration of this case.

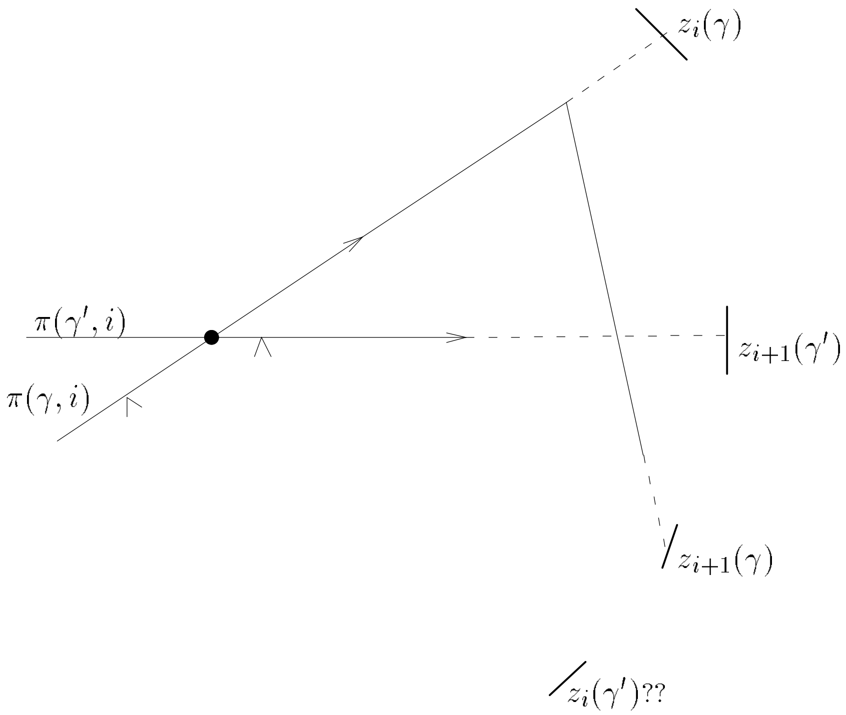



Figure 5.23: Trivial monotonicity contradiction case.

Refer to Figure 5.24 for a picture of constants defined in this section.

- $p_1$ is the intersection point between the $i$th links of $\pi(\gamma, i)$ and $\pi(\gamma', i)$ We know this intersection point must exist or we contradict the induction hypothesis.

- $p_2$ is the intersection point of the $(i+1)$st links of $\pi(\gamma, i)$ and $\pi(\gamma', i)$.

- $K_1$ is the cone with origin $p_1$ between the left ray from $p_1$ to $z_i(\gamma)$ and the right ray from $p_1$ to $z_i(\gamma')$.

- $K_2$ is the cone with origin $p_2$ between the left ray which is the negative of the ray from $p_2$ to $z_{i+1}(\gamma')$ and the right ray which is the negative of the ray from $p_2$ to $z_{i+1}(\gamma)$.

- $a$, $b$, $c$ and $d$ are defined as in the geometric observation for cones (see Lemma 2.6) using cones $K_1$ and $K_2$.

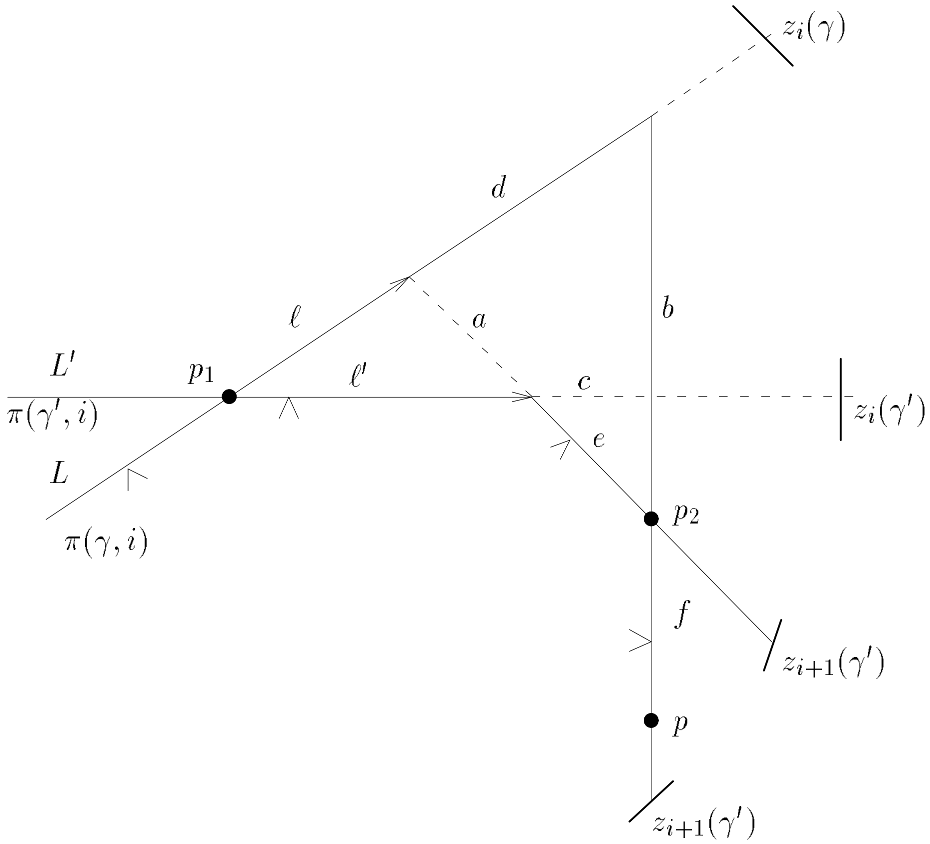



Figure 5.24: Monotonicity Lemma, harder case.

- $l$ is the length along the $i$th link of $\pi(\gamma, i)$ between $p_1$ and the intersection point with the left ray of $K_2$.
- $l'$ is the length along the $i$th link of $\pi(\gamma', i+1)$ after $p_1$.
- $p$ is a point just beyond the $(i+1)$st rocking point of $\pi(\gamma, i+1)$, along the extension of the $(i+1)$st link.
- $f$ is the distance from the intersection of the right ray of $K_1$ and the right ray of $K_2$ to this point $p$.
- $L$ is the length of $\pi(\gamma, i)$ up to the point $p_1$, and
- $L'$ is the length of $\pi(\gamma', i)$ up to the same point.

We know that $\pi(\gamma, i+1)$ is the shortest $(i+1)$-link path to any point after the $i+1$st rocking point. This follows from the way we traced the path using the local optimality conditions to choose our next interior bend point. As a consequence of local optimality we know that using $\pi(\gamma', i)$, extending the $i$th link and then using the portion of the $i+1$st link of $\pi(\gamma, i+1)$ will not be a locally optimal path. The $i$th link of such a path would not satisfy the local optimality condition. Thus, we know that $(L + l + d + b + f) \le (L' + l' + c + f)$.

Similarly we know that $\pi(\gamma', i+1)$ is the shortest $(i+1)$-link path to any point after its $i+1$st rocking point. By the same reasoning as above, using $\pi(\gamma, i)$ and then using an extension of the $i+1$st link of $\pi(\gamma', i+1)$ will not satisfy the local optimality conditions. Thus, we also know that $(L + l' + e) \le (L + l + a + e)$.

Combining these, we get $(l + d + b - c) \le l' \le (l + a)$, which implies $(d + b) \le (a + c)$, which contradicts the geometric observation made above.

Thus, $z_i(\gamma)$ must precede $z_i(\gamma')$ along the chain $C$, since we get a contradiction if we assume otherwise. $\square$

The main consequence of this lemma is that as we adjust the first turn point monotonically, the locally optimal path that is traced using this first turn point will also "advance" monotonically along the raceway. Thus, we know that if we decrease $\gamma$ we will not go as far along in the raceway, whereas if we increase $\gamma$ we will go farther along in the raceway.

This lemma validates the binary search approach to locate the optimal first bend point. If we test a first bend point and find that it "overshoots" our goal we can throw away bend point choices after the test point on the current interval $\Gamma$, since they will also overshoot the goal. Similarly if we "undershoot" the goal we can throw away the interval before the test point since these points will also undershoot. If at each stage we throw away roughly half the possible first bend points we will quickly narrow in on a small interval of candidate first bend points. Next, we will show that when this interval is made small enough any of the first bend points in the interval will trace a path that is very close to an optimal $k$-link path.

### 5.4.10 Bounding the Path Length Error

Here we will analyze how to bound the size $|\Gamma|$ in order to guarantee a path quality factor of $(1+\epsilon)$. First, we will show that if an optimal path is "trapped" in small wedges with common combinatorial type then the length of a path in these wedges will not be very far from optimal. Next, we will show that the dilation factor of our optimality condition is bounded. Thus, we can trap our paths in arbitrarily small wedges when we use the binary search procedure until the initial interval small enough. Combining these we obtain a condition for stopping the binary-search procedure because we have found a $(1+\epsilon)$-optimal path.

**Lemma 5.16 (Trapping Lemma)** *For a given $a$ and $b$, and for a given $k$, if $\Gamma = [\gamma_{\min}, \gamma_{\max}]$ is such that the paths $\pi(\gamma, k)$ all have common combinatorial type, for $\gamma \in \Gamma$, and the angles between the ith edge of $\pi(\gamma_1, k)$ and the ith edge of $\pi(\gamma_2, k)$ are each bounded above by $\delta$, then the Euclidean length of $\pi(\gamma, k)$ is at most $(1 + O(\delta N^2))$ times the length of a shortest $k$-link path joining $a$ and $b$.*

**Proof.** We consider two separate cases. First we consider the case where neither of the paths given by tracing $\gamma_{\min}$ or $\gamma_{\max}$ hits any bash walls and where the rocking points are the same. We call this the "No Bash point" case. See Figure 5.25.

We would like to find a lower bound on $|puq|/|pvq|$. To do this we need a lower bound on the size of $\pi - \theta_1$. Since $p$ and $q$ are vertices of the grid and the diamond region (the intersection of the two butterfly wedges) is contained in the polygon $P$, this region is also contained in the same $N$-by-$N$ grid. By Corollary 2.10, if the angle $\pi - \theta_1$ is smaller than $\theta_{\min}$, then the cone in $pvq$ can contain only collinear grid points. Since $p$ and $q$ lie on the interior convex chain, $\pi_G(s,t)$, the next point $r$ on $\pi_G(s,t)$ must lie in the cone $pvq$. However, this results in a contradiction

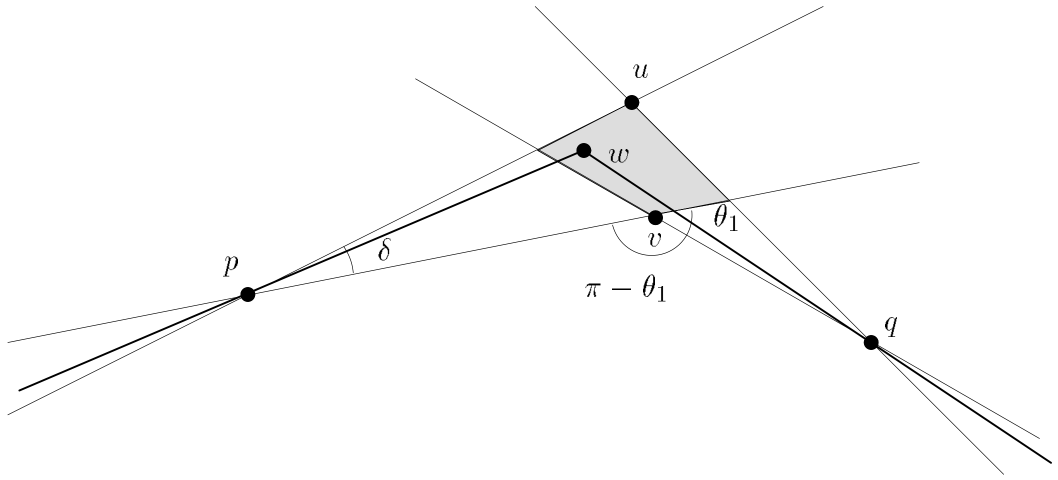



Figure 5.25: Trapping Lemma, no bash points.

because $r$ cannot be collinear with $p$ and $q$ and also be within the cone. Thus, $\pi - \theta_1$ must be larger than $\theta_{\min}$.

Recall that $\theta_{\min}$ can be no smaller than $\Omega(1/N^2)$. We can then use trigonometry and observe the following equations hold:

$$|puq| = |pu| + |uq| = |pq| \left[ \frac{\sin(\delta + A)}{\sin(A + B + 2\delta)} + \frac{\sin(\delta + B)}{\sin(A + B + 2\delta)} \right]$$

$$|pvq| = |pv| + |vq| = |pq| \left[ \frac{\sin A}{\sin(A + B)} + \frac{\sin(B)}{\sin(A + B)} \right]$$

$$\frac{|puq|}{|pvq|} = \frac{\sin(A + B)}{\sin(A + B + 2\delta)} \frac{\sin(\delta + A) + \sin(\delta + B)}{\sin A + \sin B}.$$

The next equation follows from the trigonometric identities for $\sin(x + y)$, the fact that sine must be bounded away from $\theta_{\min}$, and the fact that $\theta_{\min} \geq 1/N^2$:

$$\frac{|puq|}{|pvq|} \leq 1 + 2\delta + 2N^2\delta + 4N^2\delta^2.$$

Thus, by picking $\delta(\epsilon) = \epsilon/5N^2$ we will have the desired bound on the path length.

The second case we consider is when the paths traced using $\gamma_{\min}$ and $\gamma_{\max}$ do hit (bash) polygon walls and potentially use different rocking points (but have the

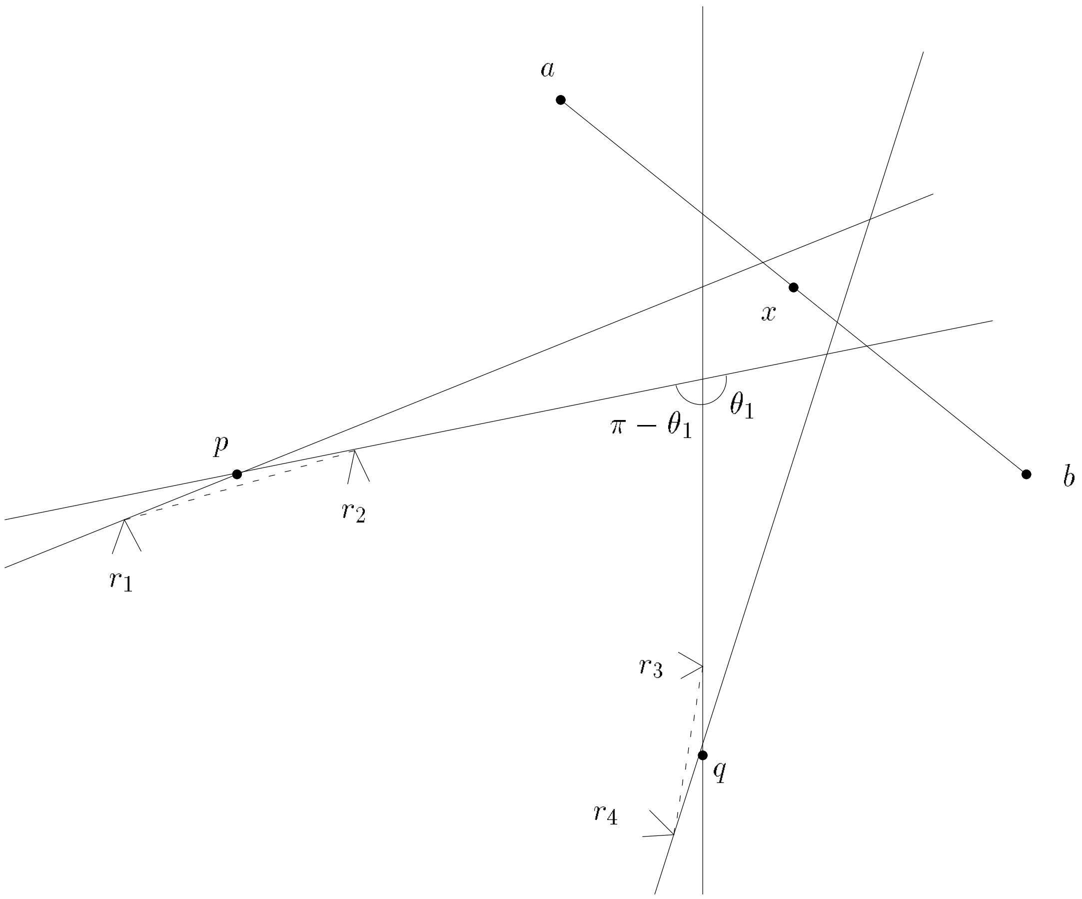



Figure 5.26: Trapping Lemma, with bash points.

same combinatorial type, i.e. they hit the same walls when they bash). We call this the "Bash point" case. See Figure 5.26.

In this case, a true optimal path might not pass through $p$ and $q$, which are points in free space, but will lie in the area described by the wedges and the dotted lines in Figure 5.26. The main consideration is to give a bound on the length of the approximate path that we choose in this region with respect to the optimal path length. Again, this is accomplished by bounding the angle $\pi - \theta_1$ from below. For example, if $p$ and $q$ are rocking points, then the angle formed by $pxq$ is again in the grid. (Points in the diamond region but outside the bash edge $\overline{ab}$ might not be part of the $N$-by-$N$ grid. However, turns are not allowed outside the bash edge.) Since this angle formed by $pxq$ is bounded below by $\theta_{\text{min}}$ (again appealing

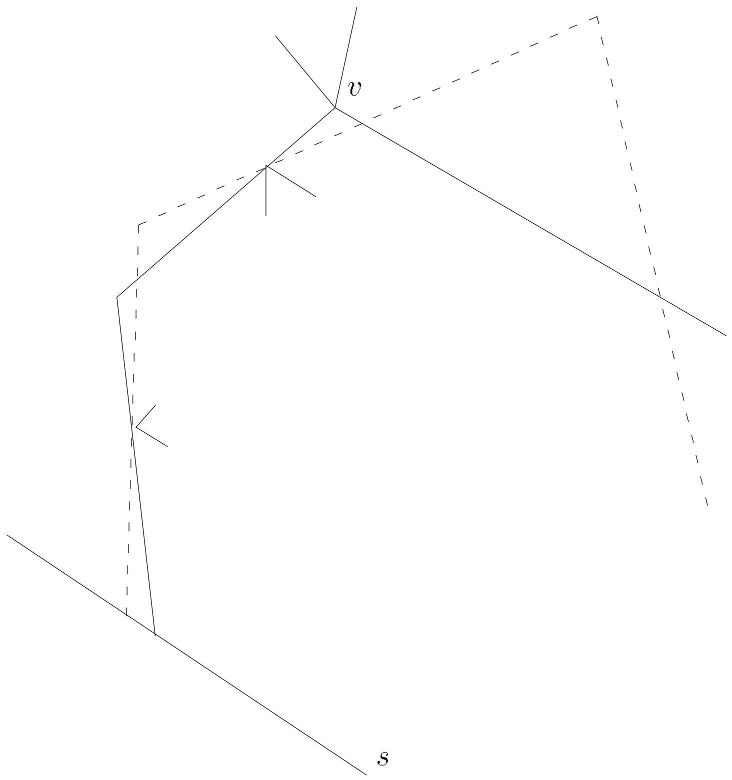



Figure 5.27: Discontinuities at leaning edges.

to Corollary 2.10), the angle $\pi - \theta_1$ will also be bounded away from $1/N^2$. A proof similar to the one used for the No Bash point case above then applies. A more extensive case analysis must be considered when $p$ and $q$ are not the rocking points. Then we need to bound pieces of the path length in separate triangular regions. In each case, we note that the rocking points $r_1$, $r_2$, $r_3$ and $r_4$ are non-collinear grid points, hence the angle $\pi - \theta_1$ cannot be less than $\theta_{\min}$, by Corollary 2.10. Also, in each case the ratio of longest to shortest path lengths will be at most $(1 + O(\delta N^2))$. So choosing $\delta(\epsilon) = \epsilon / cN^2$ will suffice for a $(1 + \epsilon)$-optimal path to be trapped. $\Box$

**Corollary 5.17** *For $\epsilon$-optimality, it suffices to select $\delta(\epsilon) = O(\frac{\epsilon}{N^2})$.*

Based on a careful analysis of the balanced-chain condition, we need to show that we are able to bound the dilation factor. We can then combine this with the Trapping Lemma above, finally validating our binary search procedure.

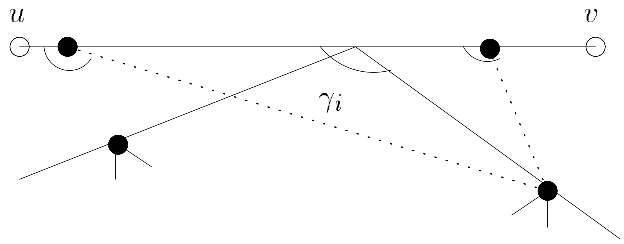



Figure 5.28: Bounding an interior turn $\gamma_i$.

First we note that "leaning" edges can cause an unbounded dilation factor. (See Figure 5.27.) Leaning edges are edges of the complete visibility polygon formed by tangencies between the inner geodesic path and the outer part of the polygon. These leaning edges can "split" our of traced paths, which would mean the angular ranges of each link would amplify in an unbounded fashion. For a small angular change crossing a leaning edge would also cause a huge change in the combinatorial type. Thus leaning edges would prevent us from obtaining a bounded dilation factor. However, we have avoided this problem by adding the $O(n)$ possible leaning edges to our dynamic programming table. Thus, the following lemma holds only for *elementary paths* (i.e. paths with interior or bash turns only, and no flush or leaning edges) with common combinatorial type:

**Lemma 5.18 (Dilation)** *Let $\theta_j$ denote the orientation of the jth link of path $\pi(\gamma, i)$, and let $\theta_j = F(\gamma)$ denote the functional dependence on the first interior bend angle, $\gamma$. Then,*

$$|F'(\gamma)| \le k^{O(1)} N^{O(k)},$$

*implying that the dilation $D = k^{O(1)} N^{O(k)}$.*

**Proof.** For our local optimality condition (see Section 5.4.5) we showed that

$$\frac{d\gamma_{i+1}}{d\gamma_i} = \frac{h_{j+1} \csc^2(\gamma_j - A_{j+1})}{H_{j+1} \csc^2 \gamma_{j+1}}.$$

We know that $c/N^2 < \gamma_i < \pi - c/N^2$ for some constant $c$. Again, this is because an angle formed by two grid points and some other point within the $N$-by-$N$ grid cannot be too small. If the other point lies on some line formed by grid points in the $N$-by-$N$ grid then the angle also cannot be too large. (Recall Lemma 2.7 about geometry in a grid.) Also, by similar reasoning, we know that $c/N^2 < A_i < \pi - c/N^2$ for all $A_i$. (Refer to Figure 5.12.)

We can express our local optimality condition as a function of $\gamma_1, \ldots, \gamma_{k-2}$, $A_2, \ldots, A_{k-2}$, $x_1, \ldots x_{k-3}$, and $y_1, \ldots, y_{k-2}$. This function can be written as a rational polynomial of $\sin \gamma_i$, $\cos \gamma_i$, $\sin A_i$, $\cos A_i$, $x_i$, $y_i$, $h_i$, $H_i$.

We know that the sine and cosine expressions are at least $> d/N^4$ for some constant $d$. We also know that $x_i, y_i, h_i$ and $H_i$ are bounded between $a/N^2$ and $N$ for some constant $a$. Thus, we know that each derivative is bounded, i.e., $O(1/N^{c'}) \leq d\gamma_{i+1}/d\gamma_i \leq O(N^{c'})$, for a small constant term $c'$. As a corollary, we also know that $c/N^{O(i)} \leq d\gamma_i/d\gamma_1 \leq N^{O(i)}$.

We can also bound the second derivatives, $d^2\gamma_i/d\gamma_1^2$. We can write $d^2\gamma_i/d\gamma_1^2$ as $d/d\gamma_1[d\gamma_i/d\gamma_1]$, or in general,

$$\frac{d}{d\gamma_i}\left[\frac{h_i \csc^2(\gamma_{i-1} - A_i)}{H_i \csc^2 \gamma_i} \cdots \frac{h_2 \csc^2(\gamma_{2-1} - A_2)}{H_2 \csc^2 \gamma_2}\right].$$

This can be bounded by $O(i^2 N^{O(i)})$. The numerator will be $O(i^2)$ terms of degree $2i$ and can be bounded by $O(i^2)N^{O(i)}$. The denominator can be as small as $1/N^{O(i)}$ (recall that $\csc^2\theta = 1/sin^2\theta \geq 1$). Thus the second derivative is bounded by $O(i^2 N^{O(i)})$.

Combining these bounds we get the desired bound $|F'(\gamma)| \leq k^{O(1)} N^{O(k)}$. □

Using these Lemmas, we can now state the running time for solving all the elementary raceway problems, and tabulating the solutions:

**Lemma 5.19 (Tabulating the Solutions)** *A table of all of the values $g_i(a,b)$ can be computed in total time $O(n^3k^3\log(Nk/\epsilon^{1/k}))$, where $N$ is the largest integer coordinate of any vertex of $P$.*

**Proof.** There are $O(n^2k)$ values of $g_i(a,b)$ to tabulate. Each requires a binary search for its first bend point. There are

$$O\left(\frac{|\Gamma_0|}{\delta_0}\right) = O\left(k\log\left(\frac{k}{\delta(\epsilon)}\right)^{1/k} D\right) = O\left(k^2 \log\frac{Nk}{\epsilon^{1/k}}\right)$$

tests necessary to find the first bend point. Each test uses a tracing procedure which requires running time $O(n)$. □

### 5.4.11 Putting the Pieces Together

We have tabulated $g_i(a,b)$, an $\epsilon$-optimal approximation to the shortest $i$-link elementary path from $a$ and $b$ (where $a$ and $b$ are flush or leaning edges in the same raceway) for all values of $i$. We now show how the path pieces computed so far can be assembled into a global solution by means of dynamic programming.

We find an $\epsilon$-optimal approximation to a shortest $k$-link path from $s$ to $t$ in two stages. In the first stage we compute $f_i(j)$, the length of the $\epsilon$-optimal approximation to the shortest $i$-link path in raceway $j$. Using these values, we compute $h_i(j)$, the $\epsilon$-approximation to the shortest $i$-link path from $e_j$ to $t$, where $e_j$ is an inflection edge. This gives us an $\epsilon$-optimal path from $s$ to $t$.

For the first stage, let us fix the raceway $R_j$, bounded by inflection edges $e_j$ and $e_{j+1}$. Define $F(a,i)$ to be the $\epsilon$-optimal length of the $i$-link path from $a$ to $e_{j+1}$, where $a$ is a flush or leaning edge in raceway $j$. We need to tabulate a total of $O(nk)$ such values. Initialize $F(a,i)$ to be $g_i(a,e_{j+1})$. Compute $F(a,i) = \min\{\min_{m,c}\{g_m(a,c)+F(c,i-m+1)\}, F(a,i)\}$ for $a < c < e_{j+1}$ (and for $c$ which are leaning edges with rocking points between $a$ and $e_{j+1}$) and $1 < m < i$, with $a$ travelling backwards along $\pi_G(s,t)$ from $e_{j+1}$ to $e_j$. There are $O(kn)$ choices for $m$ and $c$, so it takes total time $O(n^2k^2)$ to compute all the values $f_i(j) = F(e_j,i)$. Summing over all $O(k)$ possible raceways, it takes time $O(n^2k^3)$ to compute all values $f_i(j)$.

For the second stage let $h_i(j)$ be the $\epsilon$-approximate length of a shortest $i$-link path from $e_j$ to $t$. We can compute these recursively by using the following: $h_i(j) = \min_{i'}\{f_{i'}(j) + h_{i-i'}(j+1)\}$. There are $k$ choices for $i'$ and $k \cdot k$ choices for $i$ and $j$. Thus, we can tabulate all the $h_i(j)$'s in $O(k^3)$ time.

The time bounds for these two stages are dominated by the time to tabulate the values $g_i(a,b)$ (the elementary paths between flush edges and leaning edges).

We summarize our main result:

**Theorem 5.20** *One can compute an $\epsilon$-optimal shortest $k$-link path in an $n$-sided simple polygon $P$ in time $O(n^3k^3\log(Nk/\epsilon^{1/k}))$, where $N$ is the largest integer coordinate of any vertex of $P$.*

## 5.5 Open Problems

Among the triple of criteria, (links, $L_2$, total turn), we now know that the bicriteria path problem associated with one of the pairs, ($L_2$, total turn), is NP-hard, that the problem associated with the pair (links, total turn) is exactly solvable in polynomial time, and that the pair (links, $L_2$) has a fully polynomial approximation scheme (but we do not know its exact computational complexity).

Many open problems remain. For example, we have not solved the approximation problem for polygons with holes. Can we find an approximation that is within an *additive* constant rather than a factor of 2? Can we obtain an algorithm whose running time is polynomial in $n$ and polylogarithmic in $1/\epsilon$? The difficulty in this problem is that geodesic paths are not unique and they may have a completely different homotopy type than that of a shortest $k$-link path. If the homotopy type is specified in advance, it should be possible to find the shortest $k$-link path in using results similar to those obtained for the similar case of finding link shortest paths of a given homotopy type [HS91].

Some questions about the computational complexity of these problems remain open. Is the decision problem of finding a $k$-link path in a simple polygon with holes with Euclidean length less than or equal to $L$ NP-hard? Even in a simple polygon without holes this question is open.

In this chapter we have assumed that our source point $s$ and goal point $t$ are fixed. It would be interesting to answer a preprocessing version for this problem, in which we either fixed a source point $s$, or to preprocessed the polygon for two-point queries. Another interesting problem is to allow the source and destination to be segments or polygons instead of points.

We also have not answered the question posed by Chang [Cha86] about the complexity of finding the minimum perimeter $k$-link enclosing polygon. This problem seems difficult, because seems to require a “double” binary search. In a simple polygon we knew the first link from $s$ would be at a fixed orientation, the same as the first link of the geodesic path. To find an *enclosing* polygon we do not have a start point $s$ and thus do not know the starting orientation of the first edge. Obviously the path leans on some vertex of the polygon. It is certainly possible to “guess” a starting vertex, and a starting edge (orientation) leaning on this vertex. Then we can use our tracing procedure using this starting edge and an initial guess at the first turn angle to trace a locally optimal path around the polygon (based on the local optimality condition without bashes) until the path closes upon itself. The path obtained in this way would almost certainly not obey optimality with its last link. Perhaps it would be possible to adjust both starting orientation and the first turn angle based on how unbalanced the last link is. This would require proving a monotonicity condition to explain how to adjust both orientation and angle at the same time.

Even if we had such a double-binary search method it might not be very efficient because we would have to try all vertices as potential rocking points. It may be that the first point we pick is one that is not touched by the minimum perimeter $k$-link enclosing polygon. A "rotating calipers" argument which explains how to move from vertex to vertex in an efficient way might help to find the minimum perimeter path quickly.

# Chapter 6

# Approximating Link Distance in a Polyhedral Environment

Since we have had some success using link distance as a discrete measure of path quality, we would like to extend our results to higher dimensions. In the plane, we know how to compute the link distance between two points in a simple polygon, or in a polygonal environment, in polynomial time [Sur87,MRW92]. However, in higher dimensions this problem is not as well understood. At present, algorithms exist for restricted sets of obstacles, for example rectilinear (also called orthohedral) obstacles [dBvKN91], or obstacles with edges coming from a fixed set of orientations.

In the plane, the basic principle underlying link distance is to use "illumination" steps [MRW92]. From the start point, imagine a light bulb illuminating the obstacle scene. What is illuminated by the light bulb is what can be reached in one step from the start point. The boundaries of this illuminated region consist of straight line segments. By "illuminating" from this region you then find what can be reached in two steps. By continuing this process $k$ times, the region of points link distance $k$ away from the start point can be determined. In the planar case this can be done efficiently because the boundaries of illumination regions always consist of straight line segments.

In three dimensions the problem of computing the exact link distance between two points seems difficult because the boundaries of the illumination regions are more complicated. After one step of illumination from a point the boundaries are planar polygonal patches, but after two steps of illumination the boundaries (formed by illumination from one segment to another) can be hyperbolic regions. It is not clear how algebraically difficult it becomes to continue the illumination process from this point.

It should be possible to obtain an exponential-time polynomial space (PSPACE) algorithm for the minimum-link problem, using the theory of the reals to compute an exact solution. The constraints that the path be piecewise linear and not intersect the obstacles could be written down as an expression in the first order theory of the reals with a bounded number of alternations between quantifiers. This could then be converted to a polynomial of exponential degree using Renegar's PSPACE algorithm for the theory of the reals [Ren92a,Ren92b,Ren92c]. Using Neff's root-finding techniques [Nef90] one could then find the roots of this polynomial to obtain an optimal solution. This technique has been used for other motion planning algorithms, for example to obtain an exact algorithm for a kinodynamic planning problem in the plane [CRR91].

In this chapter we take a different approach and *approximate* the link distance between two points in a polyhedral environment to within a constant factor.

Planning a Euclidean shortest path in three dimensions is known to be NP-hard [CR87,Can87]. Examining Canny and Reif's NP-hardness construction we can see that it also implies that the bicriteria version of (links, Euclidean length) in three dimensions is also NP-hard. Each of the exponentially many equivalent shortest path classes that they construct has the same link length. Those path classes which are lengthened to go around barriers also have their link length increased by 1. Thus the bicriteria version of links and length is hard in three dimensions. There may be some way to modify this construction to show the single criterion link problem, just minimizing the number of links in the shortest path, is also NP-hard.

As an aside we note that the proof of the NP-hardness of computing the shortest path in three dimensions under any $L_p$ metric depends crucially on the existence of obstacle edges that can "split" a particular path via reflection into two equivalent paths, doubling the number of equivalent path classes. Using $n$ obstacles one can generate $2^n$ equivalent path classes. Roughly speaking each such path represents a different satisfying assignment to a given 3SAT formula, and by manipulating these paths through obstacles representing clauses one can show that a short path exists if and only if the formula is satisfiable.

However, not all questions about computing shortest paths in higher dimensions have been completely answered. As an example, consider that Clarkson, Kapoor and Vaidya [CKV87] gave an $O(n^2(\log n)^3)$ algorithm to compute the $L_1$ shortest path amongst orthohedral obstacles (the union of rectangular boxes). This does not contradict the NP-hardness result because the restricted orientation of both the paths and the boxes does not allow path splitting. Finding the boundary between which shortest path problems are solvable in polynomial time and which are most likely intractable in higher dimensions remains an interesting open question.

Finally, we note that there exist approximation algorithms for computing shortest paths in higher dimensions. Papadimitriou [Pap85] and Clarkson [Cla87] give algorithms to compute paths within $(1+\epsilon)$ of the length of the shortest path, where the running times of their algorithms are dependent on the bit complexity of the problem instance and on $(1/\epsilon)$. Papadimitriou's algorithm takes time $O(n^3(L+\log(n/\epsilon))^2/\epsilon)$ where $L$ depends on the problem instance. Clarkson's approximation algorithm for shortest path motion planning takes $O(n\log n)/\epsilon$ time to build a data structure of size $O(n/\epsilon)$ that will give a $(1+\epsilon)$ shortest path in roughly linear time.

In the planar case, recall that it suffices to use the visibility graph to find the Euclidean shortest path between a source and destination. However, in three dimensions it does not suffice to use visibility between vertices. (See O'Rourke [O'R87] for the picture of an example due to Seidel that shows that vertex guards may leave many regions of a polyhedron unseen.)

One approach to approximating link distance in the plane is to use the arrangement of the extensions of visibility graph edges. Each link in a minimum-link path can be perturbed to rest on a vertex. The bendpoint between them is a witness to there being 2-link visibility between two vertices. Furthermore, if two vertices have 2-link visibility, there will be some witness vertex on the arrangement of visibility graph extension edges. Searching this arrangement gives a factor two approximation to link distance. Obviously this technique is inefficient when dealing with link distance in the plane since there are more efficient exact solutions [MRW92].

We briefly note a result for ray shooting in higher dimensions that will be useful. For more detail on current work in ray shooting we refer the reader to [Pel93] and [Moh93]. Currently, the best results for ray-shooting in three dimensions show that, given a collection of $n$ arbitrary triangles in three dimensions and an

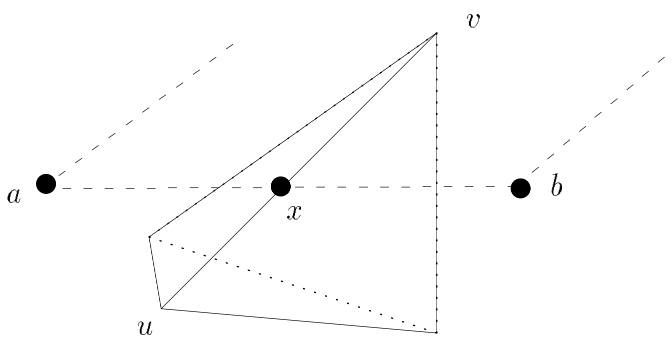



Figure 6.1: Replace link $\overline{ab}$ with four links.

$\epsilon > 0$, one can use $O(n \log^4 n)$ preprocessing time to obtain a data structure of size $O(n \log^3 n)$ which supports ray shooting queries in $O(n^{3/4+\epsilon})$ time per query. This is the best result which uses nearly linear space. When linear space is not required, there is a tradeoff between time and space. For example, if $O(n^{4+\epsilon})$ space is allowed, the query time drops to $O(\log n)$.

We also note that a polytope in three dimensions with $n$ vertices and $r$ reflex edges can be decomposed into $O(n + r^2)$ tetrahedra (by adding $r^2$ Steiner points) in $O(nr + r^2 \log r)$ time [CP90].

Again, we will use the principle of perturbing an optimal path onto a canonical path and show that the canonical path can approximate the optimal one to within a constant number of links. We assume that the optimal path has $L$ links.

**Theorem 6.1** *A minimum-link path in a polyhedral environment can be approximated to within a constant factor in polynomial time.*

**Proof.** We will consider three methods to obtain this approximation.

### Method 1

First we show that a minimum-link path in a polyhedral environment can be approximated to within a factor of four in polynomial time.

We replace each link of an optimal path by four links: perturb the link $\overline{ab}$ so that it rests against an obstacle edge $\overline{uv}$ at contact point $x$; replace $\overline{ab}$ with $\overline{axvxb}$.

Doing this for every link yields $4L$ links. Between each pair of obstacle vertices on the new path we have four links. See Figure 6.1.

Thus, it suffices to compute 4-visibility (or 4-link visibility) between every pair of obstacle vertices. How can we do this? Illuminate each vertex. The "windows" are flat polygonal regions, so they can be triangulated. It suffices to compute 2-visibility between pairs of such triangles. (There can be a total of $O(n^2)$ triangles per vertex v; $O(n^3)$ in all; thus, $O(n^6)$ pairs of triangles.) Computing 2-visibility between triangles can certainly be accomplished in polynomial time, although the time bound is quite high (a 2-linkage between two triangles has seven degrees of freedom).

### Method 2

Next, we show a minimum-link path in a polyhedral environment can be approximated to within a factor of three in $O^*(n^{7.75})$ time.

Again, we perturb an optimal path so each link rests on an obstacle edge. We can now think of approximating this using 2-visibility between pairs of contact edges since the optimal path is a witness to there being 2-visibility between consecutive pairs of contact edges. Thus, finding a path in the 2-visibility graph on obstacle edges will yield a path with at most $3L$ links (one link will follow the first contact edge, while the next two links will be used to reach the next contact edge).

For this approach, we need to be able to compute 2-visibility between pairs of edges. There are $O(n^2)$ pairs of edges. Consider two of them: $e$, $f$. A 2-link path from $e$ to $f$ has 5 degrees of freedom (one on $e$, one on $f$, three for the bend point). Thus, 5 "critical contacts" will "pin" such a linkage. For each of the $O(n^5)$ choices of contacts, we can check if each of the (constant number of) corresponding pinned linkages is feasible by doing ray-shooting in three dimensions (currently $O^*(n^{3/4})$ time) along each of the two links. This gives a total of $O(n^7)$ ray-shoots, or $O^*(n^{7.75})$ to get within a factor of three of optimum.

### Method 3

Finally, we show a minimum-link path in a polyhedral environment can be approximated to within a factor of two in $O(n^8\alpha(n))$ time.

This last approach shows a different approximation method based on using a tetrahedralization of free space. A tetrahedralization, T, of free space has size $n^2$,

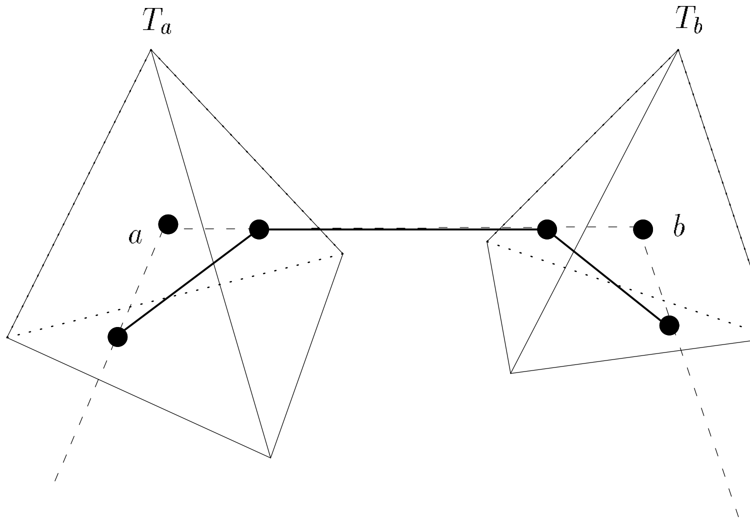



Figure 6.2: Replace link $\overline{ab}$ with triangle visibility.

after adding necessary Steiner points. For each link $\overline{ab}$ in a minimum-link path, we find the simplex $T_a$ and $T_b$ containing each endpoint.

We truncate the link $\overline{ab}$ where it enters $T_a$ and $T_b$. We do this for every link and then add one link per simplex to join the clipped ends. See Figure 6.2. This yields a $2L$ length path.

To find this approximate path, we need to compute 1-visibility between $O(n^4)$ pairs of triangles (those defining facets of Steiner simplices). The methods of McKenna and O'Rourke [McK87,MO88] show how to compute the visibility between triangles in $O(n^4\alpha(n))$ time. This gives an $O(n^8\alpha(n))$ method to get within a factor of two of optimal. □

Obviously, the bounds presented here for link distance in three dimensions are quite crude, but they are the first results for link distance without orientation restrictions. It would be interesting to improve the running times of these approximation algorithms, and to investigate the complexity of computing an exact minimum-link path in a polyhedral environment.

# Chapter 7

# Conclusion

This thesis has examined the complexity of solving various bicriteria path problems in a geometric setting. Chapter 2 provided some background material, reviewing bicriteria path problems in graphs and previous results for shortest paths in a geometric setting. It also reviewed some useful geometric lemmas.

In Chapter 3 we showed that two geometric bicriteria path problems, simultaneously minimizing path length in two different norms, and simultaneously minimizing length of travel in two regions, are NP-hard.

Chapter 4 presented more positive results when one criterion is total turn. Although the problem of finding a path with bounded Euclidean length and bounded total turn with each subject to its own bounds was shown to be NP-hard, we gave a pseudo-polynomial time approximation algorithm. The running time of this algorithm is $O(En^2N^2)$, where $N$ depends on the size of the underlying grid from which obstacle vertices are chosen. We also gave a $O(E^3 n \log^2 n)$ time algorithm for the problem of exactly computing a path with a bounded number of links and a bounded amount of total turn.

Chapter 5 concerned finding paths with short Euclidean length and a bounded number of links. We showed how to find an approximation to the optimal path, relaxing the number of links allowed, in a polygon with holes. We also gave a more accurate approximation algorithm for computing a path with $k$ links and length within $(1 + \epsilon)$ times optimal in a simple polygon. The running time of this algorithm, $O(n^3 k^3 \log(Nk/\epsilon^{1/k}))$ depends on the bit complexity of the input and a user specified tolerance $\epsilon$.

Finally, Chapter 6 discussed several approaches for approximating a minimum-link path in a polyhedral environment.

# Bibliography


[AAG$^+$86] Ta. Asano, Te. Asano, L. J. Guibas, J. Hershberger, and H. Imai. Visibility of disjoint polygons. *Algorithmica*, 1:49–63, 1986.

[ABO$^+$89] A. Aggarwal, H. Booth, J. O'Rourke, S. Suri, and C. K. Yap. Finding minimal convex nested polygons. *Information and Computation*, 83(1):98–110, October 1989.

[ACY85] A. Aggarwal, J. S. Chang, and C. K. Yap. Minimum area circumscribing polygons. *The Visual Computer*, 1:112–117, 1985.

[AMP91] E. M. Arkin, J. S. B. Mitchell, and C. D. Piatko. Bicriteria shortest path problems in the plane. In *Proceedings of the Third Annual Canadian Conference on Computational Geometry*, pages 153–156, 1991.

[AMS92] E. M. Arkin, J. S. B. Mitchell, and S. Suri. Optimal link path queries in a simple polygon. In *Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 269–279, 1992.

[AW88] H. Alt and E. Welzl. Visibility graphs and obstacle-avoiding shortest paths. *Zeitschrift für Operations Research*, 32:145–164, 1988.

[Baj86] C. Bajaj. Proving geometric algorithm non-solvability: An application of factoring polynomials. *Journal of Symbolic Computation*, 2:99–102, 1986.

[BRYY90] R. E. Burkard, G. Rote, E. Y. Yao, and Z. L. Yu. Shortest polygonal paths in space. *Computing*, 45:51–68, 1990.

[Can87] J. Canny. *The Complexity of Robot Motion Planning.* MIT Press, Cambridge, MA, 1987.

[CDRX88] J. Canny, B. R. Donald, J. Reif, and P. Xavier. On the complexity of kinodynamic planning. In *Proceedings of the 29th Annual IEEE Symposium on the Foundations of Computer Science*, pages 306–316, 1988.

[CE92] B. Chazelle and H. Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *Journal of the ACM*, 39:1–54, 1992.

[Cha86] J. S. Chang. Polygon optimization problems. Technical Report 240, Department of Computer Science, New York University, New York, NY, 1986.

[Cha91] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete and Computational Geometry*, 6:485–524, 1991.

[CKV87] K. L. Clarkson, S. Kapoor, and P. M. Vaidya. Rectilinear shortest paths through polygonal obstacles in $O(n(\log n)^2)$ time. In *Proceedings of the Third Annual ACM Symposium on Computational Geometry*, pages 251–257, 1987.

[Cla87] K. L. Clarkson. Approximation algorithms for shortest path motion planning. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 56–65, 1987.

[CP90] B. Chazelle and L. Palios. Triangulating a non-convex polytope. *Discrete and Computational Geometry*, 5:505–526, 1990.

[CR87] J. Canny and J. H. Reif. New lower bound techniques for robot motion planning problems. In *Proceedings of the 28th Annual IEEE Symposium on the Foundations of Computer Science*, pages 49–60, 1987.

[CRR91] J. Canny, A. Rege, and J. Reif. An exact algorithm for kinodynamic planning in the plane. *Discrete and Computational Geometry*, 6:461–484, 1991.

[dB91] M. de Berg. On rectilinear link distance. *Computational Geometry: Theory and Applications*, 1:13–34, 1991.

[dBvKN91] M. de Berg, M. van Kreveld, and B. J. Nilsson. Shortest path queries in rectilinear worlds of higher dimension. In *Proceedings of the 7th Annual ACM Symposium on Computational Geometry*, pages 51–60, 1991.

[dBvKNO90] M. de Berg, M. van Kreveld, B. J. Nilsson, and M. H. Overmars. Finding shortest paths in the presence of orthogonal obstacles using a combined $L^1$ and link metric. In *Proceedings of the 2nd Scandinavian Workshop on Algorithm Theory*, volume 447 of *Lecture Notes in Computer Science*, pages 213–224. Springer-Verlag, 1990.

[dRLW89] P. J. de Rezende, D. T. Lee, and Y. F. Wu. Rectilinear shortest paths in the presence of rectangular barriers. *Discrete and Computational Geometry*, 4:41–53, 1989.

[Ede87] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer-Verlag, Heidelberg, West Germany, 1987.

[FW88] S. Fortune and G. Wilfong. Planning constrained motion. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 445–459, 1988.

[GHL+87] L. J. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2:209–233, 1987.

[GHMS91] L. J. Guibas, J. E. Hershberger, J. S. B. Mitchell, and J. S. Snoeyink. Approximating polygons and subdivisions with minimum link paths. In *Proceedings of the 2nd Annual SIGAL International Symposium on Algorithms*, volume 557 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.

[Gho91] S. Ghosh. Computing visibility polygon from a convex set and related problems. *Journal of Algorithms*, 12:75–95, 1991.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, 1979.

[GM90] S. K. Ghosh and A. Maheshwari. An optimal algorithm for computing a minimum nested nonconvex polygon. *Information Processing Letters*, 36:277–280, 1990.

[GM91] S. K. Ghosh and D. M. Mount. An output-sensitive algorithm for computing visibility graphs. *SIAM Journal on Computing*, 20:888–910, 1991.

[Han80] P. Hansen. Bicriterion path problems. In G. Fandel and T. Gal, editors, *Multiple Criteria Decision Making: Theory and Applications*, volume 177 of *Lecture Notes in Economics and Mathematical Systems*, pages 109–127. Springer Heidelberg, 1980.

[Hen85] M. I. Henig. The shortest path problem with two objective functions. *European Journal of Operational Research*, 25:281–291, 1985.

[HS91] J. Hershberger and J. Snoeyink. Computing minimum length paths of a given homotopy class. In *Proceedings of the 2nd Workshop on Algorithms and Data Structures*, volume 519 of *Lecture Notes in Computer Science*, pages 331–342. Springer-Verlag, 1991.

[HZ80] G. Y. Handler and I. Zang. A dual algorithm for the constrained shortest path problem. *Networks*, 10:293–310, 1980.

[II88] H. Imai and M. Iri. Polygonal approximations of a curve – formulations and algorithms. In G. T. Toussaint, editor, *Computational Morphology*, pages 71–86. North-Holland, Amsterdam, Netherlands, 1988.

[Ke89] Y. Ke. An efficient algorithm for link-distance problems. In *Proceedings of the 5th Annual ACM Symposium on Computational Geometry*, pages 69–78, 1989.

[Law76] E. Lawler. *Combinatorial Optimization: Networks and Matroids.* Holt, Rinehart and Winston, New York, NY, 1976.

[LP84] D. T. Lee and F. P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks*, 14:393–410, 1984.

[McK87] M. McKenna. Worst-case optimal hidden-surface removal. *ACM Transactions on Graphics*, 6:19–28, 1987.

[Mit86] J. S. B. Mitchell. Planning shortest paths. Ph.D. Thesis, Stanford University, Stanford, CA, 1986.

[Mit89] J. S. B. Mitchell. An optimal algorithm for shortest rectilinear paths among obstacles. In *Proceedings of the 1st Canadian Conference on Computational Geometry*, page 22, 1989.

[Mit90] J. S. B. Mitchell. Algorithmic approaches to optimal route planning. Technical Report 937, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY, October 1990. Appeared in *Proceedings of the SPIE Conference on Mobile Robots*, November 4–9, Boston, MA, 1990.

[Mit91] J. S. B. Mitchell. A new algorithm for shortest paths among obstacles in the plane. *Annals of Mathematics and Artificial Intelligence*, 3:83–106, 1991.

[Mit92] J. S. B. Mitchell. $L_1$ shortest paths among polygonal obstacles in the plane. *Algorithmica*, 8:55–88, 1992.

[Mit93] J. S. B. Mitchell. Shortest paths among obstacles in the plane. In *Proceedings of the 9th Annual ACM Symposium on Computational Geometry*, pages 308–317, 1993.

[MMO] J. Mote, I. Murthy, and D. Olsen. An empirical investigation of the number of pareto optimal paths obtained for bicriterion shortest path problems. Manuscript.

[MO88] M. McKenna and J. O'Rourke. Arrangements of lines in 3-space: a data structure with applications. In *Proceedings of the 4th Annual ACM Symposium on Computational Geometry*, pages 371–380, 1988.

[Moh93] S. Mohaban. Ray shooting amidst spheres in three dimensions. M.Sc. Thesis, School of Mathematical Sciences, Tel Aviv University, Tel Aviv, Israel, 1993.

[MP91] J. S. B. Mitchell and C. H. Papadimitriou. The weighted region problem: finding shortest paths through a weighted planar subdivision. *Journal of the ACM*, 38:18–73, 1991.

[MRW92] J. S. B. Mitchell, G. Rote, and G. Woeginger. Minimum-link paths among obstacles in the plane. *Algorithmica*, 8:431–459, 1992.

[Nef90] C. A. Neff. Specified precision polynomial root isolation is in NC. In *Proceedings of the 31st Annual IEEE Symposium on the Foundation s of Computer Science*, pages 152–162, 1990.

[NR92] B. K. Natarajan and J. Ruppert. On sparse approximations of curves and functions. In *Proceedings of the 4th Canadian Conference Computational Geometry*, pages 250–256, 1992.

[OAMB86] J. O'Rourke, A. Aggarwal, S. Maddila, and M. Baldwin. An optimal algorithm for finding minimal enclosing triangles. *Journal of Algorithms*, 7:258–269, 1986.

[O'R87] J. O'Rourke. *Art Gallery Theorems and Algorithms.* Oxford University Press, New York, NY, 1987.

[Pap85] C. H. Papadimitriou. An algorithm for shortest-path motion in three dimensions. *Information Processing Letters*, 20:259–263, 1985.

[Pel93] M. Pellegrini. Ray shooting on triangles in 3–space. *Algorithmica*, 1:49–63, 1993.

[PS85] F. P. Preparata and M. I. Shamos. *Computational Geometry: an Introduction.* Springer-Verlag, New York, NY, 1985.

[Ren92a] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals. Part I: Introduction. Preliminaries. The geometry of semi-algebraic sets. The decision problem for the existential theory of the reals. *Journal of symbolic computation*, 13:255–299, 1992.

[Ren92b] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals. Part II: the general decision problem. Preliminaries for quantifier elimination. *Journal of symbolic computation*, 13:301–327, 1992.

[Ren92c] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals. Part III: Quantifier elimination. *Journal of symbolic computation*, 13:329–352, 1992.

[RS85] J. H. Reif and J. A. Storer. Minimizing turns for discrete movement in Euclidean space with polyhedral obstacles. Report, Aiken Computing Laboratory, Harvard University, Cambridge, MA, 1985.

[Sha87] M. Sharir. On shortest paths amidst convex polyhedra. *SIAM Journal on Computing*, 16:561–572, 1987.

[SS86] M. Sharir and A. Schorr. On shortest paths in polyhedral spaces. *SIAM Journal on Computing*, 15:193–215, 1986.

[Sta89] M. Stanton. Pareto optimal paths among obstacles in the plane. M.Sc. Thesis, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY, 1989.

[Sur86] S. Suri. A linear time algorithm for minimum link paths inside a simple polygon. *Computer Vision, Graphics and Image Processing*, 35:99–110, 1986.

[Sur87] S. Suri. Minimum link paths in polygons and related problems. Ph.D. Thesis, Department of Computer Science, Johns Hopkins University, Baltimore, MD, 1987.

[Sur90] S. Suri. On some link distance problems in a simple polygon. *IEEE Transactions on Robotics and Automation*, 6(1):108–113, 1990.

[Wel85] E. Welzl. Constructing the visibility graph for $n$ line segments in $O(n^2)$ time. *Information Processing Letters*, 20:167–171, 1985.

[WWW87] P. Widmayer, Y. F. Wu, and C. K. Wong. On some distance problems in fixed orientations. *SIAM Journal on Computing*, 16:728–746, 1987.

[Yao82] A. C. Yao. On constructing minimum spanning trees in $k$-dimensional spaces and related problems. *SIAM Journal on Computing*, 11:721–736, 1982.

[YLW91] C. D. Yang, D. T. Lee, and C. K. Wong. On bends and lengths of rectilinear paths: a graph-theoretic approach. In *Proceedings of the 2nd Workshop on Algorithms and Data Structures*, volume 519 of *Lecture Notes in Computer Science*, pages 320–330. Springer-Verlag, 1991.