SCHOOL OF OPERATIONS RESEARCH
AND INDUSTRIAL ENGINEERING
COLLEGE OF ENGINEERING
CORNELL UNIVERSITY
ITHACA, NEW YORK 14853

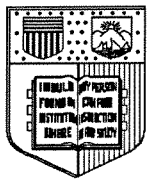
TECHNICAL REPORT NO. 896

March 1990
(Revised, November 1990)

OPTIMAL MOTION OF COVISIBLE POINTS AMONG OBSTACLES IN THE PLANE

By

Joseph S. B. Mitchell
Erik L. Wynters



OPTIMAL MOTION OF COVISIBLE POINTS AMONG OBSTACLES IN THE PLANE

Joseph S. B. Mitchell[1]

School of Operations Research
and Industrial Engineering
Cornell University
Ithaca, NY 14853

Erik L. Wynters[2]

Center for Applied Mathematics
Cornell University
Ithaca, NY 14853

Abstract

We consider the problem of determining an optimal pair of paths for two point robots that must remain covisible while moving in a plane cluttered with polygonal obstacles. We solve the (MIN-SUM) problem of minimizing the sum of the path lengths with an algorithm that requires time $O(E + n \log n)$ and space $O(E)$, where E is the size of the visibility graph induced by the set of obstacles. We also solve the (MIN-MAX) problem of minimizing the length of the longer path in time $O(E^2 + n^2 \log n)$ and space $O(E)$. In addition, we discuss the (MIN-TIME) problem of obtaining a coordinated motion parameterized by time that minimizes the time needed for both robots to reach their destinations. Assuming a common upper bound on velocity, we find a pair of paths together with a parameterization guaranteed to be within a factor of 2 of optimality.

Key Words: shortest paths, motion planning, coordinated motion, mission planning, visibility graphs, funnels, computational geometry

---

[1] Partially supported by NSF Grants IRI-8710858 and ECSE-8857642 and by a grant from Hughes Research Laboratories.

[2] Supported by a grant from Hughes Research Laboratories.

1 Introduction

Imagine that two robots must communicate with one another while traveling among obstacles. The robot geometry, the obstacle geometry, and the effectiveness of communication can be modeled in various ways, but let's assume that we have point-sized robots, polygonal obstacles, and a communication system that works only when the robots can see one another. Assume, also, that initial and final positions for each robot have been specified and that both robots have the same maximum speed. Under these assumptions, we consider the following optimal motion planning problems:

MIN-SUM Find a pair of paths between the initial and final positions that minimizes the sum of the two path lengths subject to the communication constraint.

MIN-MAX Find a pair of paths between the initial and final positions that minimizes the larger of the two path lengths subject to the communication constraint.

MIN-TIME Find a coordinated motion parameterized by time that minimizes the time needed for both robots to reach their destinations while maintaining communication.

Note: we define these problems more precisely later in the paper.

Alternatively, we can view these problems as motion planning problems for a single robot with four degrees of freedom; imagine that the robot is a collapsible rod that can simultaneously translate, rotate, and expand or contract. We seek optimal motions for the rod, using the distances traveled by its endpoints as optimization criteria.

Prior work in this field suggested that shortest path planning for a robot with more than two degrees of freedom was inherently difficult. While shortest paths among polygonal obstacles for a point robot or a translating polygon (two degrees of freedom) can be found exactly in worst-case quadratic time, finding a shortest path for a point robot among polyhedra in three-space (three degrees of freedom) is NP-hard [CR]. The problem of moving a ladder (or any noncircular body free to rotate) in an optimal manner among obstacles in the plane (three degrees of freedom) is also very

challenging and has been solved only for a restricted class of allowable motions [PS]. A characterization of optimal motions that doesn't presuppose a certain class of motions was obtained recently [IRWY], but it applies only when no obstacles are present.

We were, therefore, somewhat surprised to discover that we can solve the MIN-SUM problem, which admits *four* degrees of freedom and allows obstacles to be present, in the same complexity as the best shortest path algorithm for a *single* point robot with *two* degrees of freedom. Our algorithm requires time $O(E + n \log n)$ and space $O(E)$.

The main ideas behind our algorithm are as follows: we characterize the geometric form of optimal solutions; we bound the number of candidate paths satisfying these geometric constraints; and we use the theory of funnel trees and graph searching to enumerate and evaluate the candidate paths in time and space proportional to the number of candidates found.

A variation of our algorithm solves the MIN-MAX problem in time $O(E^2 + n^2 \log n)$ and space $O(E)$. More generally, we can solve the "bi-criteria" version of the problem: given an upper bound on the length of one agent's path, find a motion of both covisible agents that minimizes the length of the other agent's path.

The MIN-TIME problem with bounded robot velocity seems to be quite difficult. We show that the optimal-time motion may correspond to a pair of paths that are not even locally optimal with respect to distance. Although we have not yet solved this problem exactly, we give a method of obtaining an approximate solution guaranteed to be within a factor of 2 of optimality.

Our algorithms rely on a method for efficiently enumerating the lengths of all "hourglasses" and "funnels" defined by an obstacle space. This same method has applications to several other optimization problems. In particular, in [MW] we apply this method to yield an efficient solution to the following *bipartition problem*: Given a set S of n points in the plane, find a bipartition of S, $S = S_1 \cup S_2$, such that the sum (or the maximum) of the perimeters of $conv(S_1)$ and $conv(S_2)$ is minimized. The method further extends to the case of bipartitioning a set of polygons according to various optimality criteria.

This paper is organized as follows:

- Section 2 gives basic terminology;
- Section 3 characterizes the form of optimal motions for the MIN-SUM problem;
- Section 4 outlines the MIN-SUM algorithm;
- Section 5 gives the combinatorial analysis and algorithmic details;
- Section 6 solves the MIN-MAX problem;
- Section 7 discusses the MIN-TIME problem; and, finally,
- Section 8 gives several extensions and future directions.

2 Definitions and Preliminaries

Consider a set $\mathcal{O}$ of disjoint, simple-polygon obstacles defined by a total of n vertices. We define *freespace*, $\mathcal{F}$, to be the set of all points in the plane that are not contained in the relative interior of any obstacle. Note that the boundaries of obstacles are included in freespace. A *configuration* is an ordered pair of points in freespace that are visible to one another in the sense that the line segment joining them lies within freespace. Such points are called *covisible*, and the line segment connecting them is called a *feasible* line segment.

The *visibility graph*, $VG(\mathcal{O})$, of the obstacle set $\mathcal{O}$ associates a node with each vertex of an obstacle and an edge with each pair of covisible vertices. It is known that taut-string (locally-shortest) paths lie on the visibility graph, and hence that the visibility graph can be used to search for shortest paths ([Le],[LP],[LW],[Mi],[SS]). The visibility graph can be constructed in time $O(n^2)$ ([AAGHI],[We]) or in output-sensitive time $O(E + n \log n)$, where E is the number of edges in the graph ([GM],[KM]). Alternatively, a representation of the visibility graph can be constructed in $O(E \log n)$ time using only $O(n)$ space by the algorithm of [OW].

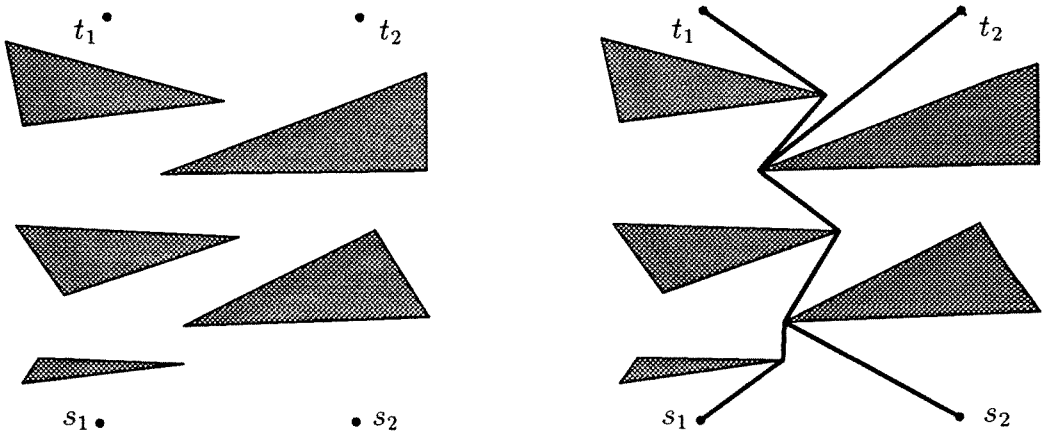


Figure 1: An instance of the MIN-SUM problem (left) together with its solution (right).

A *path function* is a continuous function from the unit interval $[0,1]$ into freespace that determines the position of a moving point at time $\tau \in [0,1]$. The image of a path function, the subset of freespace traversed by the point, is called a *path*; and any path function with image p is called a *parameterization* of p. We denote the length of path p by $\mu(p)$. A *path-pair* is an ordered pair of paths (p_1, p_2) and is *feasible* if there exist parameterizations f_1 and f_2 of p_1 and p_2, respectively, such that for all times τ the points $f_1(\tau)$ and $f_2(\tau)$ are covisible.

Now we can formally state the MIN-SUM problem (which is illustrated in Figure 1):

(MIN-SUM) The Two Point Covisible MIN-SUM Problem

Instance: *A set of polygonal obstacles, an initial configuration* (s_1, s_2), *and a final configuration* (t_1, t_2).

Question: *Find a feasible path-pair* (p_1, p_2) *between these configurations that minimizes the sum* $\mu(p_1) + \mu(p_2)$.

3 Characterizing Optimal Motions

An optimal path-pair for the MIN-SUM problem must satisfy two necessary conditions. We discuss these conditions below and show that path-pairs satisfying them form characteristic shapes.

3.1 A Necessary Condition

The first necessary condition for optimality is topological feasibility. Loosely speaking, we say that a path-pair is *topologically feasible* if its paths lie in the same "channel" between the obstacles. This definition provides a good intuitive "feel" for what topologically feasible path-pairs look like, but it isn't very rigorous. To remedy this situation, we now provide a more precise definition based on obstacle-free polygons.

Define the *path-polygon* P determined by a path-pair (p_1, p_2) from (s_1, s_2) to (t_1, t_2) to be the closed path from s_1 to s_1 that consists of path p_1, line segment $\overline{t_1 t_2}$, path p_2 (traversed from t_2 to s_2), and line segment $\overline{s_2 s_1}$. We call p_1 and p_2 the *defining paths* of P and call $\overline{s_1 s_2}$ and $\overline{t_1 t_2}$ the *defining segments* of P.

> *Remark:* The term "path-polygon" is somewhat misleading for two reasons: the individual paths in a feasible path-pair need not be polygonal, and the "path-polygon," even if polygonal, may not be a *simple* polygon. However, we will show that only path-pairs consisting of polygonal paths are important to us, and we define what it means for a polygon to be obstacle-free in such a way that it makes sense for non-simple polygons, too.

We call a path-polygon *obstacle-free* if it can be continuously deformed in freespace to a single point (see Figure 2 for an example). More precisely, we say that P is obstacle-free if P, viewed as a closed path, has a parameterization homotopic to the null loop $f(\tau) = v$ where v, the basepoint of freespace, is taken to be a vertex of P. We now present a more mathematically useful definition of topological feasibility: a path-pair is topologically feasible if it determines an obstacle-free path-polygon.

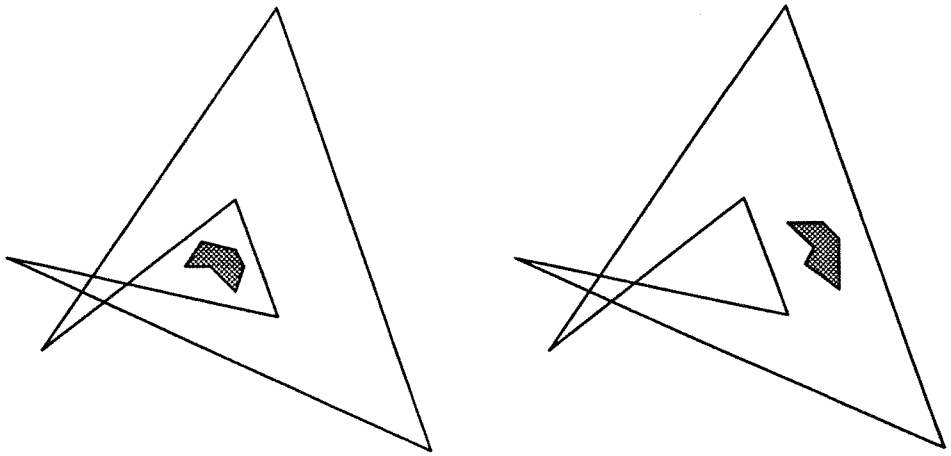


Figure 2: An obstacle-free (non-simple) polygon (left) and a non-obstacle-free polygon (right).

Our first lemma states that topological feasibility is a necessary condition for feasibility.

Lemma 1 *If a path-pair is feasible, then it must be topologically feasible.*

Proof: Suppose (p_1, p_2) is a feasible path-pair from (s_1, s_2) to (t_1, t_2). We show that the path-polygon P determined by (p_1, p_2) has a parameterization p homotopic to the null loop at the point s_1.

Since (p_1, p_2) is feasible, we have parameterizations f_1 and f_2 of p_1 and p_2, respectively, with $f_1(\tau)$ and $f_2(\tau)$ covisible for all τ. Define parameterizations s and t of the line segments $\overline{s_1 s_2}$ and $\overline{t_2 t_1}$ by

$$s(\tau) = (1 - \tau)s_1 + \tau s_2$$

and

$$t(\tau) = (1 - \tau)t_2 + \tau t_1.$$

In addition, define two constant functions v and u by

$$v(\tau) = s_1$$

and

$$u(\tau) = t_1.$$

The loop given by the composite path function

$$p = s f_2 t {f_1}^{-1}$$

is a parameterization of the path-polygon of (p_1, p_2). We show that p is homotopic to q (denoted $p \sim q$), where

$$q = v f_1 u {f_1}^{-1}.$$

This proves the lemma, because, clearly, $q \sim f_1 f_1^{-1}$ and $f_1 f_1^{-1} \sim v$.

Consider the function H given by

$$H(\sigma, \tau) = (1 - \sigma)p(\tau) + \sigma q(\tau).$$

The restriction of H to each of the closed regions shown in Figure 3 is clearly a continuous function into $\mathbf{R}^2$, so H is a continuous function into $\mathbf{R}^2$. Also, H contains only points of freespace in its range, because for every fixed τ_0, the path function $H_{\tau_0}(\sigma)$ given by $H_{\tau_0}(\sigma) = H(\sigma, \tau_0)$ is a parameterization of a feasible segment. Therefore, H is a continuous function into freespace that transforms p into q as σ varies from zero to one. In other words, $p \sim q$. ∎

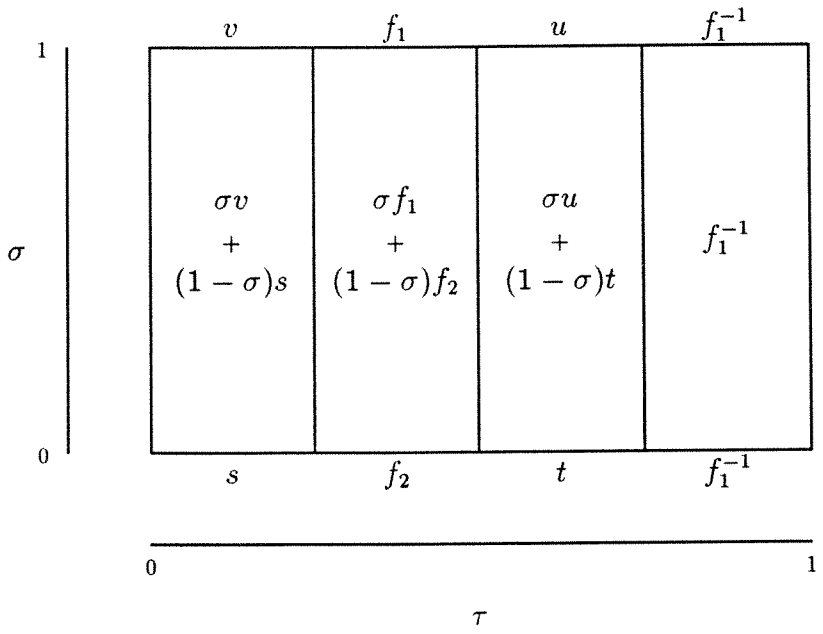


Figure 3: The homotopy H from p to q.

3.2 Hourglasses, Funnelglasses, and Bowties

Lemma 1 establishes that we can limit our search for an optimal path-pair to the set of topologically feasible ones. But there is a continuous family of topologically-feasible path-pairs for every homotopy class of paths from s_1 to t_1. Since we find optimal path-pairs for the MIN-SUM and MIN-MAX problems by enumerating and comparing feasible candidates, we must reduce the search space further to do this efficiently.

We make the search efficient by considering only those path-pairs that are locally optimal. A locally-optimal path for the single-point shortest path problem is a path that cannot be shortened by making local changes; it is a "taut-string" path, which is known to consist of visibility graph edges ([Le], [SS], [Mi]). Similarly, a path-pair is *locally optimal* if each individual path is a taut-string path.

Local optimality is the second of the two necessary conditions for optimality that we mentioned in the beginning of this section. In order to show this, we have to address the question of whether a feasible path-pair can always be "pulled taut" without destroying its feasibility. We show that the answer to this question is "yes" in two steps: first, we show that locally-optimal, topologically-feasible path-pairs form geometric objects called "hourglasses," "funnelglasses," and "bowties," and then we show that path-pairs forming these objects are always feasible. We now define these objects.

An *hourglass between a feasible line segment* $\overline{s_1 s_2}$ *and another feasible segment* $\overline{t_1 t_2}$ is an obstacle-free path-polygon determined by two *disjoint* paths p_1 and p_2, where p_1 is a locally optimal path from s_1 to t_1, and p_2 is a locally optimal path from s_2 to t_2. Figure 4 illustrates a simple hourglass.

A *funnel between vertex* v *and line segment* $\overline{s_1 s_2}$ is a simple, obstacle-free polygon bounded by $\overline{s_1 s_2}$, a locally optimal path from v to s_1 and a locally optimal path from v to s_2 (see Figure 5). The vertex v is called the *apex* of the funnel, and the segment $\overline{s_1 s_2}$ is called the *base*. The path from apex to base with the interior of the funnel on its left side is called the *lower chain* of the funnel and the other path is called the *upper chain*.

A *funnelglass between a feasible line segment* $\overline{s_1 s_2}$ *and another feasible segment* $\overline{t_1 t_2}$ is a pair of funnels (f_1 and f_2 with bases $\overline{s_1 s_2}$ and $\overline{t_1 t_2}$), respectively, joined by a shortest path between their apices in such a way that both the path p_1 from s_1 to t_1 and the path p_2 from s_2 to t_2 are taut.

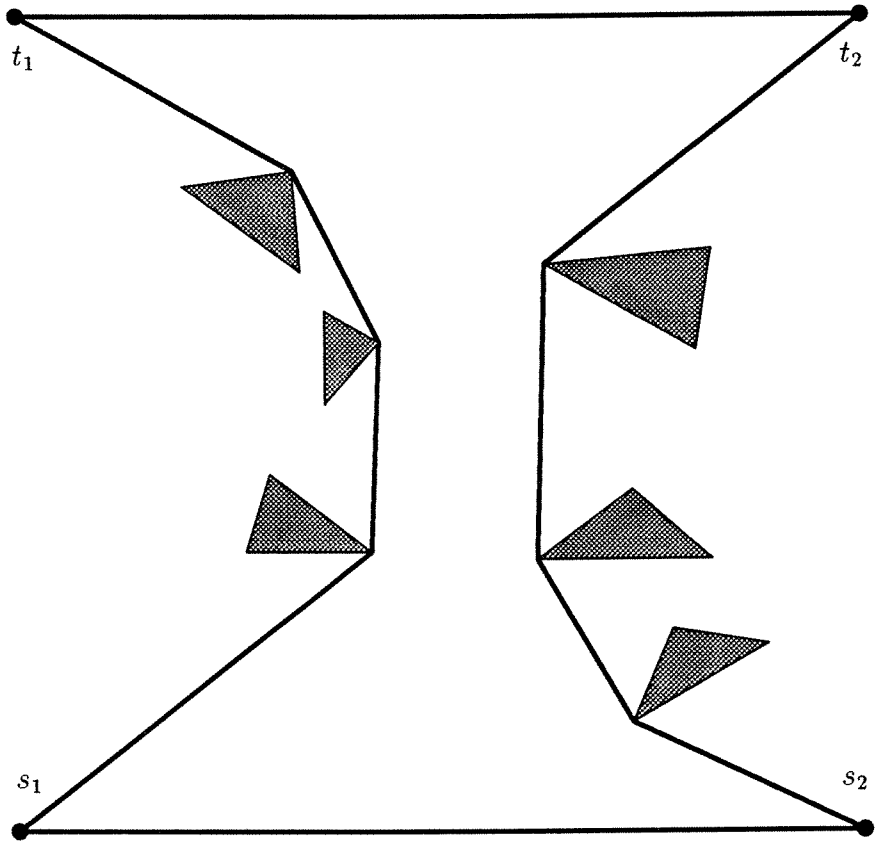


Figure 4: A simple hourglass.

Figure 6 gives an example of a funnelglass.

A *bowtie between a feasible line segment* $\overline{s_1 s_2}$ *and another feasible segment* $\overline{t_1 t_2}$ is an obstacle-free path-polygon determined by these line segments and two locally optimal paths, p_1 from s_1 to t_1 and p_2 from s_2 to t_2, that intersect at a single point that is not a vertex. See Figure 7 for an example of a bowtie.

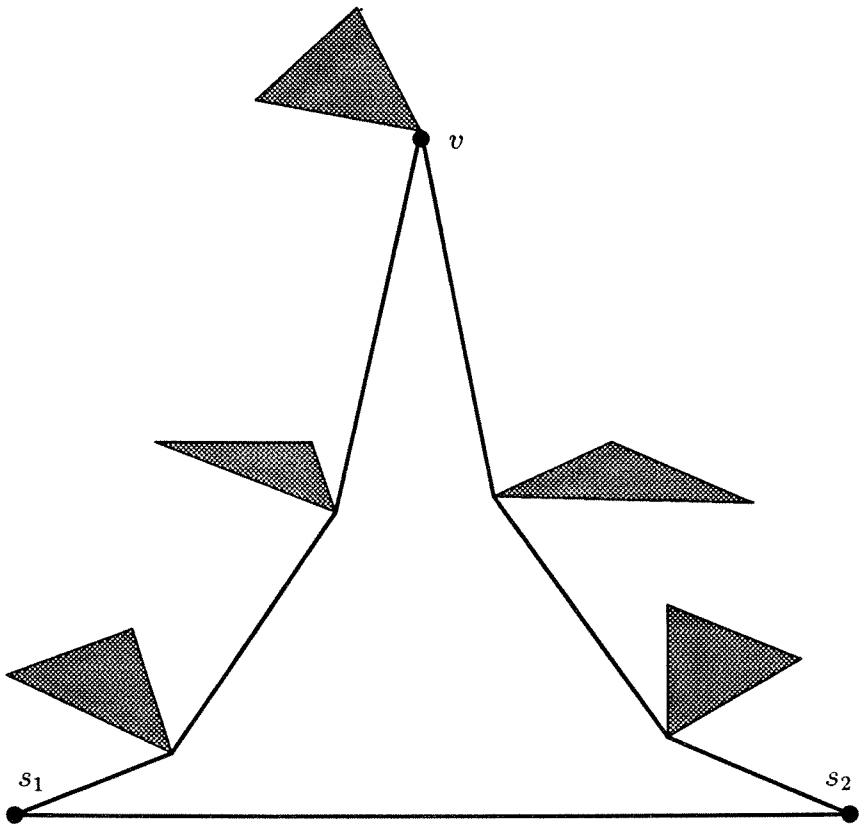


Figure 5: A funnel with apex v and base $\overline{s_1 s_2}$.

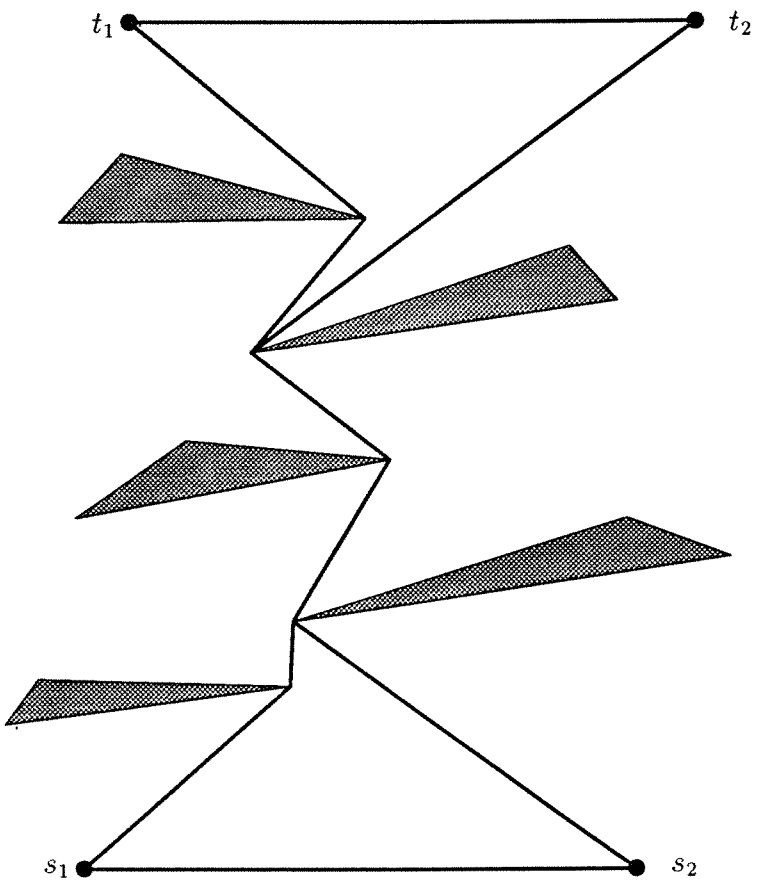


Figure 6: A funnelglass.

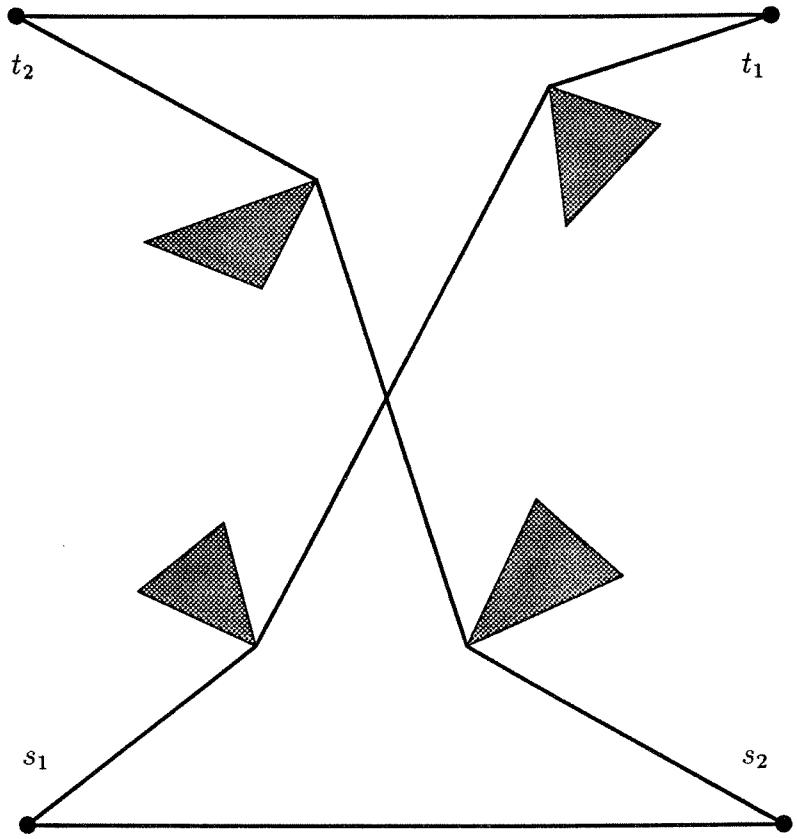


Figure 7: A bowtie.

3.3 Previous Work and Miscellaneous Remarks

In this section we provide additional background information on funnels, hourglasses, funnelglasses, and bowties. We mention previous work related to these objects, state a few of their important properties, and illustrate some special configurations that can occur.

Funnels have been described before in the computational geometry literature. They arise naturally in shortest path and visibility problems in simple polygons ([LP],[GHLST],[He],[GH]). They also play a crucial role in Ghosh and Mount's algorithm for finding the visibility graph of a polygon with holes ([GM]).

Although funnels are not, themselves, path-polygons, they are of utmost importance in our algorithm, too. We use funnels to construct funnelglasses, to generate hourglasses and bowties efficiently, and to analyze the combinatorial complexity of the set of locally-optimal, topologically-feasible path-pairs.

We also define something called a pseudo-funnel. A *pseudo-funnel* is an object that must satisfy every requirement for being a funnel except the one requiring its apex to be located at a vertex. We use pseudo-funnels in arguments where we need to apply funnel properties to a funnel-like object that may or may not have its apex at a vertex.

Funnels (or pseudo-funnels) have two important properties that we make use of in this section: first, the upper and lower chains of a funnel are inward-convex [LP]; and second, the apex of a funnel is visible from a point on the interior of its base [GM]. Some other funnel properties established by previous workers are mentioned in Section 5.

Hourglasses also have a history in computational geometry. They arise in visibility-from-an-edge problems ([GHLST],[AGT]) and in shortest path problems in a simple polygon ([GH]). In addition, Ghosh and Mount use hourglasses to prove the correctness of their funnel-splitting method of constructing the visibility graph ([GM]).

The hourglasses that arose in [AGT], [GHLST], and [GM] were always simple polygons with inward-convex defining paths. Our definition of an hourglass is less restrictive; it forces us to consider non-simple hourglasses, which can arise as follows:

- one of the defining paths may be a closed (polygonal) curve, as shown in Figure 8;

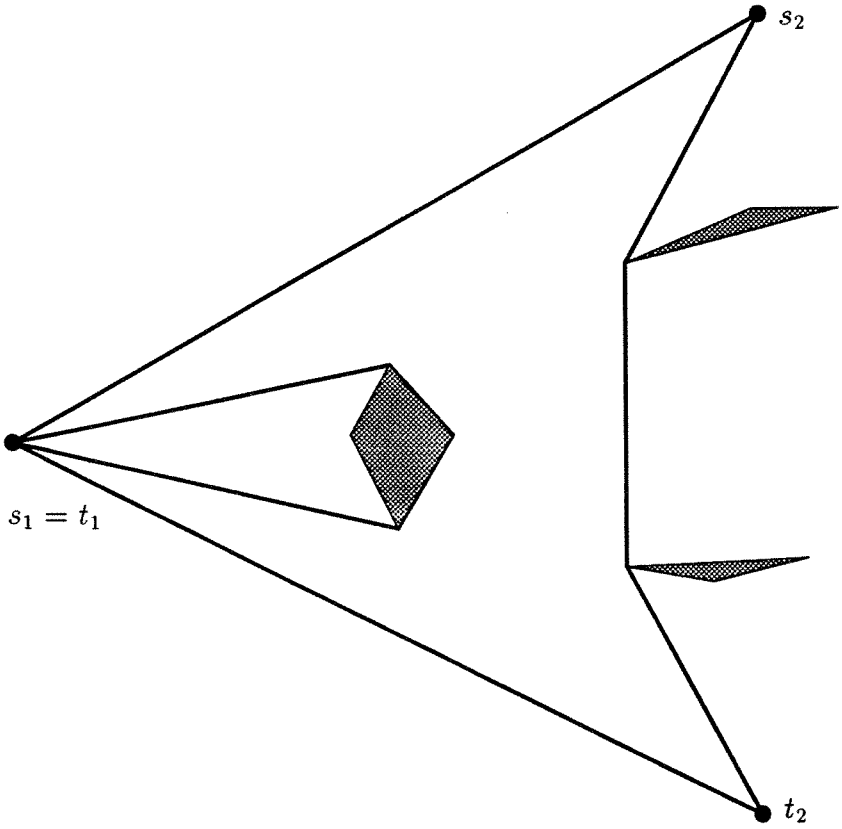


Figure 8: An hourglass for which p_1 is a closed curve.

- one of the defining paths may intersect itself and the defining line segments (which necessarily intersect each other in this case), as shown in Figure 9; or
- the defining line segments may intersect each other while the paths between them do not self-intersect and do not strictly intersect the line segments, as shown in Figure 10.

In the last case above, the hourglass is called a *twisted hourglass*; we show in Section 5 that twisted hourglasses require special processing by the algorithm.

Funnelglasses were also used to find short paths in a simple polygon [GH] but were called closed hourglasses by Guibas and Hershberger. Bowties

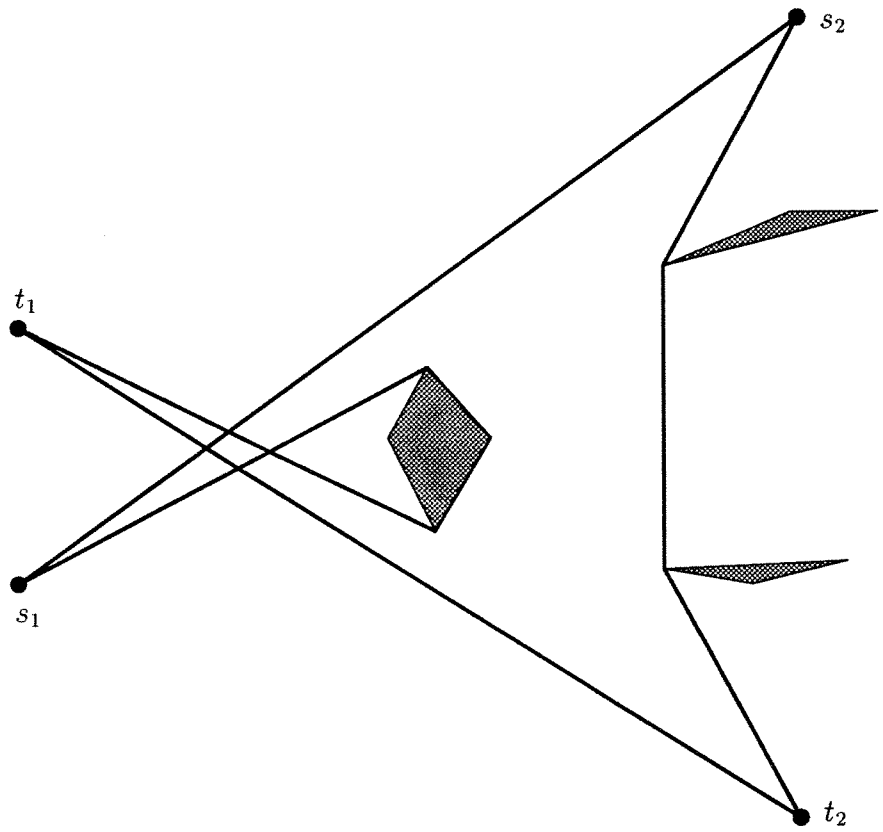


Figure 9: An hourglass with p_1 crossing itself.

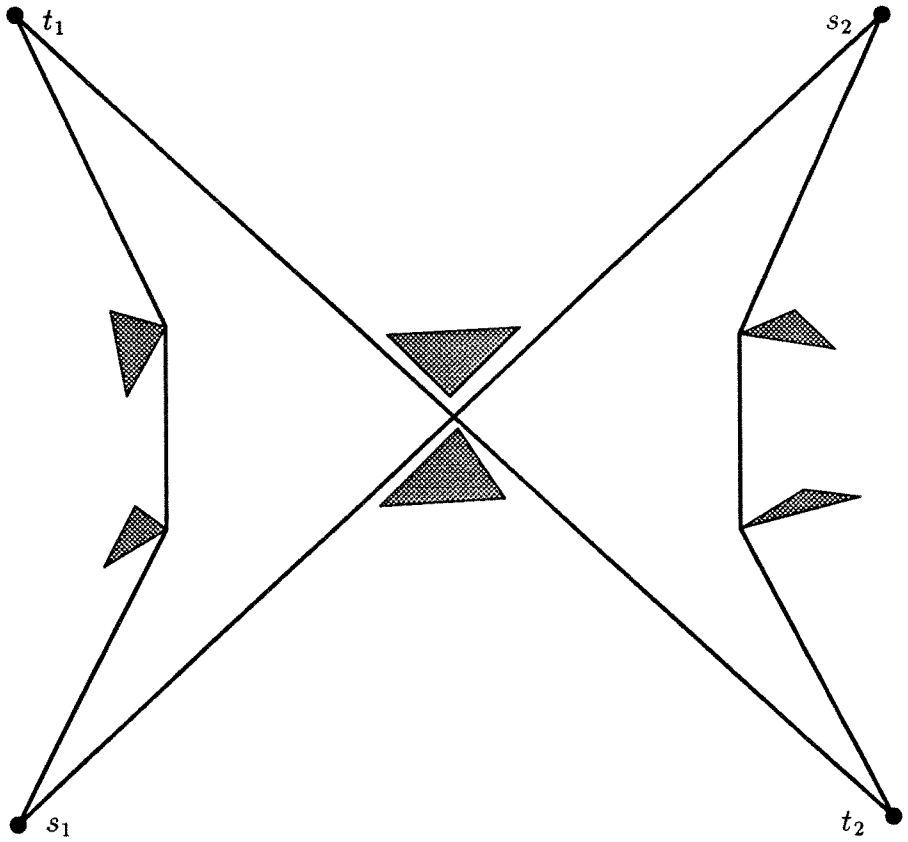


Figure 10: A twisted hourglass.

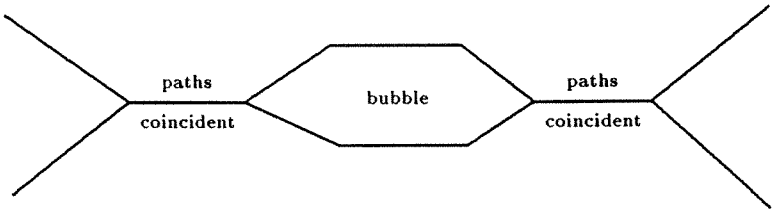


Figure 11: A bubble.

have not been named before in the computational geometry listerature, but the notion of a bowtie is evident in [GH] and in [GHLST]; these papers describe shortest paths for a single agent that cut across an hourglass, using portions of each convex chain of the hourglass together with a common tangent to these chains.

To conclude this section of general remarks, we mention a way of distinguishing between a bowtie and a degenerate funnelglass formed from two funnels with the same apex. A funnelglass of this sort looks a lot like a bowtie; these funnelglasses and bowties are both unions of two pseudo-funnels with a common apex. But the common apex of a funnelglass occurs at a vertex, while the common apex of a bowtie does not.

3.4 A Second Necessary Condition

Before we can show that optimal path-pairs must be locally optimal, we must show that locally-optimal, topologically-feasible path-pairs only form path-polygons that are hourglasses, funnelglasses, or bowties. To do this, we define something called a bubble.

If the intersection of two polygonal paths is not connected, we define a *bubble* to be a simple obstacle-free polygon formed by these paths between two consecutive connected components of their intersection; Figure 11 illustrates this idea.

We show that bubbles cannot occur in the kinds of path-pairs we're interested in:

Lemma 2 *If a path-pair is locally optimal and topologically feasible, then it determines a bubble-free path-polygon.*

Proof: The proof is by contradiction. Suppose we have a bubble between two points b_1 and b_2 formed by the paths of a topologically-feasible path-pair (p_1, p_2) that is locally optimal. Two cases are possible, each of which leads to a contradiction.

Case 1: The path p_1 between b_1 and b_2 consists of a single line segment. In this case, we can shorten p_2 by replacing the portion of it involved in the bubble by the corresponding portion of p_1. This contradicts our assumption that p_2 is locally optimal.

Case 2: We have a vertex b_3 not coincident with b_1 or b_2 such that b_3 is the first vertex reached when traveling from b_2 to b_1 on p_1. Since the paths are assumed to be locally-optimal and the bubble is obstacle-free, the segment $\overline{b_3, b_2}$ is the base of a pseudo-funnel, with apex b_1, formed by p_1 between b_1 and b_3 and from p_2 between b_1 and b_2. It is known that the apex of a funnel (or pseudo-funnel) must be visible from the interior of its base ([GM]), so we have a point i in the interior of $\overline{b_3 b_2}$ visible to b_1. Since i is a point on p_1, we can shorten p_1 by replacing the portion of it between b_1 and i by the line segment $\overline{b_1 i}$. This contradicts our assumption that p_1 is locally optimal. ∎

Now we can prove the following lemma:

Lemma 3 *If a path-pair is locally-optimal and topologically feasible, then its path-polygon must be an hourglass, a funnelglass, or a bowtie.*

Proof: The paths of a path-pair satisfying the hypotheses of the lemma cannot form a bubble by Lemma 2 and consist of visibility graph edges. Therefore the paths must satisfy exactly one of the following three conditions.

1. the paths are disjoint;
2. the intersection of the paths is a polygonal chain whose endpoints are vertices of the visibility graph (this includes the case in which the intersection is a single vertex); or
3. the intersection of the paths is a single point that is not a vertex of the visibility graph:

If the paths satisfy the first condition, they determine an hourglass; if they satisfy the second condition, they determine a funnelglass; and if they satisfy the third condition, they determine a bowtie. ∎

Our next lemma states that locally-optimal, topologically-feasible path-pairs not only form characteristic shapes but also admit feasible parameterizations.

Lemma 4 *If a path-pair is locally optimal and topologically feasible, then it must be feasible.*

Proof: We need to show that the paths that determine hourglasses, funnelglasses, and bowties can be parameterized so as to keep the moving points covisible at all times. After partitioning one of these path-polygons into simple polygons and triangulating its components, we construct a feasible parameterization using three types of elementary motions that we call "single-step motions," "double-cross motions," and "telescoping see-saw motions."

A *single-step motion* is one in which one of the two moving points remains stationary at a vertex while the other moves the entire length of a single straight line section of its polygonal path. A sequence of single step motions is sufficient to parameterize a funnel or a simple hourglass since a triangulation of one of these objects gives a sequence of diagonals between initial and final configurations in which adjacent diagonals correspond to configurations obtainable from one another by a single-step motion (this follows from the inward-convexity of the defining paths). Non-simple non-twisted hourglasses like those shown in Figures 8 and 9 can be partitioned by a tangent edge (see Section 5) into a sequence of two simple hourglasses with the final configuration of one hourglass equal to the initial configuration of the next one. So these hourglasses, along with funnelglasses, can be parameterized by single-step motions.

A *double-cross motion* is a coordinated motion in which the two points move simultaneously along two intersecting line segments in such a way that they meet at the intersection point. Double-cross motions together with single-step motions are sufficient to parameterize bowties, because a bowtie can be decomposed into two simple hourglasses joined by two intersecting line segments.

A *telescoping see-saw motion* is a coordinated motion obtained from a rotating line segment. We use these motions to parameterize twisted hourglasses like the one shown in Figure 10. We rotate a line segment through the intersection point of the defining segments of the hourglass, from the initial configuration to the final configuration. The intersection points of this rotating segment with the two polygonal paths gives a simultaneous parameterization of the paths that keeps the moving points covisible. ∎

We are finally at the point where we can state that local optimality is a necessary condition for optimality.

Lemma 5 *If a path-pair is optimal, then it must be locally optimal.*

Proof: An optimal path pair is topologically feasible by Lemma 1. It must also be locally optimal; otherwise, it could be improved by "pulling it taut" and the resulting locally-optimal, topologically-feasible path-pair would still be feasible by Lemma 4. ∎

3.5 Main Results for the MIN-SUM Problem

Our algorithm for the MIN-SUM problem is based on the following result:

Theorem 1 *An optimal path-pair for the MIN-SUM problem is obtained by selecting one for which the path-polygon is an hourglass, a funnelglass, or a bowtie, and the associated cost* $(\mu(p_1) + \mu(p_2))$ *is minimum.*

Proof: Lemmas 1 and 5 establish that topological feasibility and local optimality are both necessary conditions for optimality. This, together with Lemma 3, proves that an optimal path-pair determines an hourglass, funnelglass, or bowtie as its path-polygon. The hourglass, funnelglass, or bowtie that minimizes $\mu(p_1)+\mu(p_2)$ is feasible by Lemma 4, so it corresponds to the optimal path-pair. ∎

Theorem 1 establishes the correctness of our algorithm, which appears in section 4. The efficiency of our algorithm depends on the number of hourglasses, funnels, and bowties that can occur. In section 5, we show that the following combinatorial bound on the number of these objects holds:

Theorem 2 *There are at most* $O(E)$ *hourglasses, funnels, and bowties.*

4 The MIN-SUM Algorithm

Instance: A set $\mathcal{O}$ of polygonal obstacles, an initial configuration (s_1, s_2), and a final configuration (t_1, t_2).

Algorithm:

1. If $\overline{s_1 s_2}$ and $\overline{t_1 t_2}$ don't intersect, then compute the visibility graph VG_1 of the obstacle set $\mathcal{O}$ and the points s_1, s_2, t_1, and t_2 treating $\overline{s_1 s_2}$ and $\overline{t_1 t_2}$ as line segment obstacles. Otherwise, compute the visibility graph VG_1 of the obstacle set $\mathcal{O}$ and the points s_1, s_2, t_1, and t_2 treating $\overline{s_1 s_2}$ as a line segment obstacle, then compute the visibility graph VG_2 of the obstacle set $\mathcal{O}$ and the points s_1, s_2, t_1, and t_2 treating $\overline{t_1 t_2}$ as a line segment obstacle, and then match corresponding edges in VG_1 and VG_2.

2. Generate all funnels with base $\overline{s_1 s_2}$ or $\overline{t_1 t_2}$ while maintaining and updating at each vertex a representation of the best funnel (in terms of total chain length) on each base having that vertex as its apex.

3. Generate all hourglasses and bowties from $\overline{s_1 s_2}$ to $\overline{t_1 t_2}$ and determine which has the best associated path-pair.

4. Define an *augmented visibility graph* $\mathcal{G}$ as follows:

   - Beginning with the visibility graph VG_1 obtained in step 1, create an additional "funnel" node fn for each minimal funnel f found in step 2.
   - Connect each funnel-node fn to the node v corresponding to the apex of funnel f. Let the length of edge (fn, v) be half the sum of the two lengths of the defining paths of f.
   - Create a "supersource" node s, linking it with an edge of length 0 to each funnel-node corresponding to a minimal funnel based on $\overline{s_1 s_2}$. Similarly, create and link a "supersink" node t to each funnel-node corresponding to a minimal funnel based on $\overline{t_1 t_2}$.

5. Find a shortest path from s to t in the graph $\mathcal{G}$. This determines the best path-pair for which the path-polygon is a funnelglass.

6. Compare the results of Step 3 and Step 5. If Step 3 produced a value smaller than twice the length of the shortest path found in Step 5, then return the path-pair corresponding to the best hourglass or bowtie. Otherwise, return the path-pair corresponding to the best funnelglass found in Step 5.

5 Discussion and Analysis

Step 1 of the algorithm builds each visibility graph in $O(E+n\log n)$ time and $O(E)$ space using the algorithm of Ghosh and Mount [GM]. The structure obtained in this step, which is called the "enhanced visibility graph" by Ghosh and Mount, provides, for each node, a representation of the nodes visible to it sorted by angle.

In the edge-matching portion of Step 1, we link together with pointers, any edges e_1 and e_2 satisfying the following conditions: e_1 is an edge of VG_1, e_2 is an edge of VG_2, and the two edges correspond to the same feasible line segment. In other words, for each edge that occurs in both graphs, we link together its two different representations. Because matching edges in this way is equivalent to merging two sorted lists, it takes $O(E)$ time, where E is the maximum of the number of edges in VG_1 and the number of edges in VG_2.

We generate funnels in step 2 of the algorithm in time and space proportional to the number of funnels found. We can represent each funnel with a fixed amount of storage space because a funnel is uniquely determined by the first edge on its lower (or upper) chain when its base is fixed [GM]. Since this fact also guarantees that the number of funnels generated is at most $2E$, the total space required to represent the funnels is $O(E)$.

Funnels can be generated in output-sensitive time because they are linked to one another in a structure called a funnel tree [GM]. Funnel trees, which are contained implicitly within the enhanced visibility graph, allow us to compute the lengths of each funnel's chains in constant time from the lengths of its parent's chains. A traversal of the "lower" funnel tree produces the lengths of all lower chains, while a traversal of the "upper" funnel tree produces the lengths of all upper chains.

In addition to providing us with the length of each funnel in amortized constant time, a funnel tree traversal produces a linear ordering of the funnels

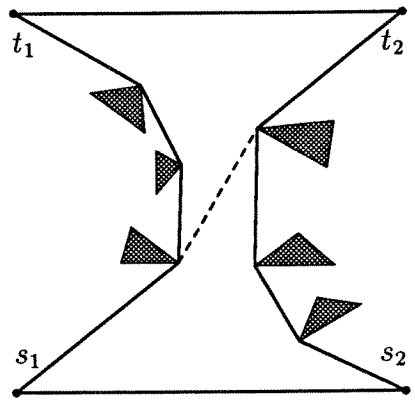


Figure 12: A tangent edge partitions an hourglass into two funnels.

called a funnel sequence [GM]. We show later that one such funnel sequence enables us to calculate the length of a bowtie efficiently.

Ghosh and Mount show that funnel trees can be traversed in $O(E)$ time after constructing the enhanced visibility graph, so step 2 of the algorithm takes $O(E)$ time.

Step 3 of the algorithm generates hourglasses and bowties between $\overline{s_1 s_2}$ and $\overline{t_1 t_2}$ and stores a representation of an hourglass or bowtie with the best associated path-pair. We discuss the generation of hourglasses first.

First, we explain why an hourglass, like a funnel, is completely specified by a single edge of the visibility graph. Each defining path of an hourglass determines a convex polygon. If the path crosses itself, the region enclosed by the path is a convex polygon; otherwise, the region bounded by the path and the line segment between its endpoints is a convex polygon. The two convex polygons formed by the paths of an hourglass have two inner common tangents which lie on edges of the visibility graph. Call these edges *tangent edges*. In the case of a twisted hourglass one tangent edge is the line segment $\overline{s_1 s_2}$ and the other tangent edge is $\overline{t_1 t_2}$. In every other case, each tangent edge determines two unique funnels, one on $\overline{s_1 s_2}$ and one on $\overline{t_1 t_2}$, that together form a bipartition of the hourglass as shown in Figure 12. Therefore, the tangent edge determines the hourglass. This argument also shows that there are at most E non-twisted hourglasses.

We generate all non-twisted hourglasses in $O(E)$ time by scanning the funnel sequence obtained from the lower funnel tree on $\overline{s_1 s_2}$. If an edge that

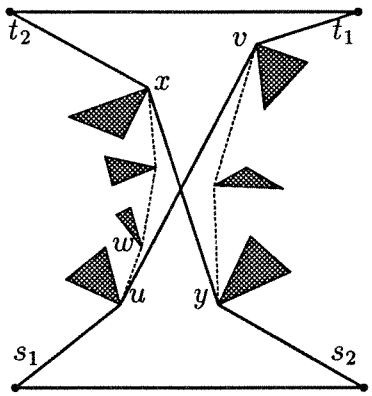


Figure 13: The crossing edges of this bowtie determine four funnels.

determines a funnel in this sequence also determines a funnel in the upper funnel tree on $\overline{t_1 t_2}$, then that edge determines a non-twisted hourglass; its length can be found in constant time from the upper and lower chain lengths of the funnels.

If $\overline{s_1 s_2}$ and $\overline{t_1 t_2}$ intersect at a point p, then the optimal path-pair could determine a twisted hourglass. Since a twisted hourglass is uniquely specified by the initial and final configurations, there is only one twisted hourglass to check. We find its defining paths by taking convex hulls of the vertices in triangle $\triangle s_1 p t_1$ and triangle $\triangle s_2 p t_2$, respectively. This part of Step 3, if required, takes $O(n \log n)$ time and $O(n)$ space.

Now we consider bowties. A bowtie is also completely specified by a single edge of the visibility graph. We explain this property of bowties below.

Each defining path of a bowtie has a special edge: the edge containing the intersection point of the two paths. Each of these *crossing edges* specifies two funnels, one on $\overline{s_1 s_2}$ and one on $\overline{t_1 t_2}$ as shown in Figure 13. Given both of these edges, we can calculate the bowtie's length from the chain lengths of these four funnels in constant time.

But we don't want to check all *pairs* of intersecting visibility graph edges to generate bowties; there could be $O(n^2)$ of these pairs. Instead, we show that if we are given one of these crossing edges, then we can find the other one in constant time. This means that a bowtie can be represented by a single crossing edge, and it means that there are at most E bowties.

We represent a bowtie by the crosssing edge (v, u) that determines a

funnel f_v with apex v in the lower funnel tree of $\overline{s_1 s_2}$ (i.e. (v, u) is the first edge on the lower chain of a funnel with apex v and base $\overline{s_1 s_2}$). The other crossing edge (x, y) determines a funnel f_x with apex x in the upper funnel tree of $\overline{s_1 s_2}$. Consider the sequence of vertices along the lower chain of funnel f_x between u and x (see Figure 13). Each of these vertices (except u) is the apex of a funnel that is a descendent of the parent of funnel f_v in the lower funnel tree. Let w be the first vertex after u in this sequence. The funnel at each vertex except w in this sequence must be the extreme clockwise child of its parent in the lower funnel tree; otherwise, the funnel f_u determined by (u, v) in the lower funnel tree of $\overline{t_1 t_2}$ wouldn't be obstacle-free. Also f_x must be a leaf node in the lower funnel tree; otherwise, its children's apices would be inside funnel f_u. This establishes that f_x is the predecessor to f_v in the funnel sequence obtained from a clockwise preorder traversal of the lower funnel tree of $\overline{s_1 s_2}$. So if we are given the crossing edge (v, u), we can find the crossing edge (x, y) in constant time.

We generate all bowties in $O(E)$ time by scanning through the funnel sequence obtained from the lower funnel tree on $\overline{s_1 s_2}$. If an edge that determines a funnel in this sequence also determines a funnel in the lower funnel tree on $\overline{t_1 t_2}$, then that edge determines a bowtie; its other crossing edge and its length can be found in constant time.

Finally, the best funnelglass is found by searching the graph constructed in Step 4 of the algorithm. The construction of this graph takes $O(E)$ time since $O(E)$ vertices and edges are added to the visibility graph. The shortest path from the supersource s to the supersink t in this graph must use a funnel node adjacent to s and a funnel node adjacent to t. In other words, this path represents a funnel on $\overline{s_1 s_2}$ connected to a funnel on $\overline{t_1 t_2}$ by a shortest path in the visibility graph. The shortest such combination must correspond to the best funnelglass. Finding the shortest path in this graph takes $O(E + n \log n)$ time ([Di], [FT]).

In summary, Step 1 and Step 5 take $O(E + n \log n)$ time, and everything else takes $O(E)$ time, except for calculating the length of the twisted hourglass (which, if necessary, takes $O(n \log n)$ time). This establishes our main result.

Theorem 3 *An optimal solution to the MIN-SUM problem can be found in $O(E + n \log n)$ time and $O(E)$ space.*

6 The MIN-MAX Problem

In the MIN-MAX problem we want to minimize the *larger* of the two path lengths, rather than the sum of the two path lengths. We use the same basic approach to solve this problem as the one we used for the MIN-SUM problem.

We enumerate funnels, hourglasses, and bowties keeping track this time of each of the two path lengths p_1 and p_2. We now choose the best bowtie or hourglass according to the min-max criterion, but the main difference is that the augmented visibility graph now has two different lengths associated with each edge. We are not able to solve as efficiently the resulting graph search problem since it is a bicriteria shortest path problem. In general, the bicriteria shortest path problem is NP-hard, but our problem has a special structure that allows an efficient solution; namely, the only edges of the graph with two *different* edge lengths are those edges incident to a funnel node.

We find the best funnelglass by enumerating all funnel pairs (with one funnel on base $\overline{s_1 s_2}$ and the other on base $\overline{t_1 t_2}$) and connecting each pair of apices by a shortest path in the visibility graph. To determine the distances between all pairs of vertices without using $\Omega(n^2)$ space we repeat the following steps for each vertex v:

1. Compute the shortest path tree rooted at v using Dijkstra's Algorithm.
2. Find the best funnelglass such that one funnel has apex v and base $\overline{s_1 s_2}$ and update the representation of the best funnelglass found so far, if necessary.

The time needed for the n iterations of Dijkstra's algorithm is $O(En + n^2 \log n)$. Since there are $O(E)$ funnels on each base, it takes $O(E^2)$ comparisons to find the best funnelglass (the cost of a funnelglass is calculated in constant time using the shortest path tree).

In summary, we have the following theorem:

Theorem 4 *An optimal solution to the MIN-MAX problem can be found in $O(E^2 + n^2 \log n)$ time and $O(E)$ space.*

We can generalize the above result to the case in which we are given an upper bound on the length of one agent's path, and we desire to minimize the length of the other agent's path subject to this constraint. The running time remains the same.

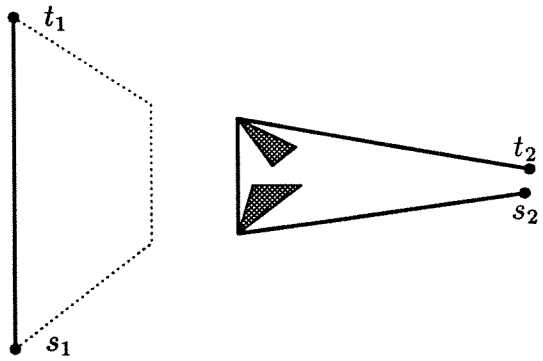


Figure 14: The deviation from local optimality shown in dots saves time.

7 The MIN-TIME problem

Our methods can be used to find an approximate solution to the following MIN-TIME problem: Given a common upper bound on the velocities of two moving points, find a coordinated motion from an initial configuration (s_1, s_2) to a final configuration (t_1, t_2) that keeps the points covisible at all times and minimizes the time needed to complete the motion.

It seems unlikely that our approach can be used to find the optimal solution exactly since min-time paths need not be locally optimal with regard to distance. For example, Figure 14 shows that deviating from a "taut string" path can reduce the time needed to complete the motion. Therefore, min-time paths need not lie on the visibility graph and will not, in general, form hourglasses, funnelglasses, and bowties. We show below, however, that any parameterization of the min-sum path-pair satisfying two specific conditions gives a provably good approximation to the optimal solution.

Theorem 5 *Any parameterization of the MIN-SUM path-pair that always keeps at least one agent moving at full speed and that never causes an agent to backtrack gives an approximate solution to the MIN-TIME problem that takes no more than twice the time of an optimal motion. Furthermore, such a parameterization always exists.*

Proof: Without loss of generality, assume that the maximum velocity is 1. Let t^* denote the time value associated with the min-time motion, and let l denote the length of the longer path in the min-time path-pair. Then $t^* \geq l$.

But $l \geq t/2$ where t is the distance associated with the min-sum path-pair; otherwise, the path-pair corresponding to the min-time motion would have a better distance value than the min-sum path-pair. So $t \leq 2t$.

If p is a parameterization satisfying the hypotheses of the theorem, then p takes time at most t to move both agents to their destinations. Thus p approximates the min-time motion to within a factor of two.

The proof of Lemma 4 – in which we showed that an optimal path-pair can always be parameterized by single-step motions, double-cross motions, and telescoping see-saw motions – implies that a parameterization satisfying these hypotheses can always be found. ∎

8 Conclusions and Further Research

We have shown that optimal solutions to the MIN-SUM and MIN-MAX problems can be computed efficiently. Our algorithm for these problems is, to the best of our knowledge, the first polynomial-time algorithm that finds an optimal motion for a robot system with more than two degrees of freedom. Many questions about optimal multi-agent motion remain open, however.

We are currently exploring the following extensions to the work presented here and hope to have additional results to report on them soon:

Transparent Obstacles Our algorithm can be extended to the following slightly more general case. Assume that the "obstacles" in the planar environment are of two varieties: "mountains" (through which both sight and travel is impossible), and "lakes" (over which we can see, but through which we cannot travel). Then our problem is to find a pair of feasible paths for the two agents such that their line-of-sight is not obstructed by mountains.

Different Constraints There are a variety of other constraints that we wish to impose on the pair of agents. For example, we are investigating the difficult problem of putting an upper bound on the distance between the agents, without requiring that they remain covisible.

Visibility Breaks Instead of demanding that the two agents *always* be covisible, we would like to allow brief breaks in the visibility. We may want to find the best pair of paths (min-sum or min-max) subject to

the constraint that the agents not lose communication for more than some given ϵ percent of their journeys. This gets us into the domain of more general bicriteria optimization problems.

Better Parameterizations The parameterization described in Section 7 approximates the min-time motion to within a constant factor, but is far from intelligent; in most cases, we move only one agent at a time. To improve these parameterizations we could try to find the minimum-time parameterization of the min-sum path-pair. This may not improve the worst-case time bound but should be considerably better on average. Another possibility is to approximate the min-time motion or the best motion having a particular homotopy class using numerical methods.

Several Agents The case in which there are more than two agents is especially interesting. There are many tricky topological cases to consider, but it seems that our general method will apply to the version in which we wish to minimize the sum of the path lengths subject to *clique visibility* (that every pair of agents remain covisible at all times). Other specifications of visibility constraints among multiple agents include requiring that there is *some* path of line-of-sight communication joining any two agents (i.e., that the communication network remain connected at all times).

Acknowledgements

We would like to thank Esther Arkin, Robert Freimer, Samir Khuller, and Christine Piatko for helpful discussions. This research was partially supported by a grant from the Hughes Research Laboratories, Malibu, CA, and by NSF Grants IRI-8710858 and ECSE-8857642.

References

[AAGHI] T. Asano, T. Asano, L. Guibas, J. Hershberger, and H. Imai, "Visibility of Disjoint Polygons", *Algorithmica*, Vol. 1, (1986), pp. 49-63.

[AGT] D. Avis, T. Gum, and G. Toussaint, "Visibility between Two Edges of a Simple Polygon", *The Visual Computer*, Vol. 2, (1986), pp. 342-357.

[CR] J. Canny and J. Reif, "New Lower Bound Techniques for Robot Motion Planning Problems", *Proc. 28th FOCS*, pp. 49-60, Oct. 1987.

[Di] Dijkstra "A Note On Two Problems in Connection With Graphs", *Numerische Mathematik*, 1 (1959), pp. 269-271.

[FT] M. Fredman and R. Tarjan, "Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms", *Proc. 25th Annual IEEE Symposium on Foundations of Computer Science*, pp. 338-346, 1984.

[GHLST] L. Guibas, J. Hershberger, D. Levin, M. Sharir, and R. E. Tarjan, "Linear Time Algorithms for Visibility and Shortest Path Problems inside Simple Polygons", *Proc. of the 2nd ACM Symp. on Computational Geometry*, pp. 1-13, 1986.

[GH] L. Guibas, and J. Hershberger, "Optimal Shortest Path Queries in a Simple Polygon", *Proc. of the 3rd ACM Symp. on Computational Geometry*, pp. 50-63, 1987.

[GM] S.K. Ghosh and D.M. Mount, "An Output Sensitive Algorithm for Computing Visibility Graphs", Technical Report CS-TR-1874, Department of Computer Science, University of Maryland, July 1987. (Also appears in FOCS, 1987.)

[He] J. Hershberger, "Finding the Visibility Graph of a Simple Polygon in time Proportional to its Size", *Proc. of the 3rd ACM Symp. on Computational Geometry*, pp. 11-20, 1987.

[IRWY] C. Icking, G. Rote, E. Welzl, and C. Yap, "Shortest Paths for Line Segments", Technical Report, Fachbereich Mathematik, Freie Universitaet Berlin, September 1989.

[KM] S. Kapoor and S.N. Maheshwari, "Efficient Algorithms for Euclidean Shortest Path and Visibility Problems with Polygonal Obstacles", *Proc. Fourth Annual ACM Symposium on Computational Geometry*, Urbana-Champaign, IL, June 6-8, 1988, pp. 172-182.

[Le] D.T. Lee, "Proximity and Reachability in the Plane", Ph.D. Thesis, Technical Report ACT-12, Coordinated Science Laboratory, University of Illinois, Nov. 1978.

[LP] D.T. Lee and F.P. Preparata, "Euclidean Shortest Paths in the Presence of Rectilinear Boundaries", *Networks*, **14** (1984), pp. 393-410.

[LW] T. Lozano-Pérez and M.A. Wesley, "An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles", *Communications of the ACM*, Vol. 22, No. 10 (1979), pp. 560-570.

[Mi] J.S.B. Mitchell, "A New Algorithm for Shortest Paths Among Obstacles in the Plane", Technical Report No. 832, School of Operations Research and Industrial Engineering, Cornell University, October, 1988.

[OW] M.H. Overmars and E. Welzl, "New Methods for Computing Visibility Graphs", *Proc. Fourth Annual ACM Symposium on Computational Geometry*, Urbana-Champaign, IL, June 6-8, 1988, pp. 164-171.

[SS] M. Sharir and A. Schorr, "On Shortest Paths in Polyhedral Spaces", *SIAM Journal on Computing* Vol. 15, No. 1, pp. 193-215, February 1986.

[We] E. Welzl, "Constructing the Visibility Graph for n Line Segments in $O(n^2)$ Time", *Information Processing Letters*, Vol. 20 (1985), pp. 167-171.