Geometric Shortest Paths and Network Optimization

Joseph S.B. Mitchell *

August 4, 1998

1 Introduction

A natural and well-studied problem in algorithmic graph theory and network optimization is that of computing a "shortest path" between two nodes, s and t, in a graph whose edges have "weights" associated with them, and we consider the "length" of a path to be the sum of the weights of the edges that comprise it. Efficient algorithms are well known for this problem, as briefly summarized below.

The shortest path problem takes on a new dimension when considered in a geometric domain. In contrast to graphs, where the encoding of edges is explicit, a geometric instance of a shortest path problem is usually specified by giving geometric objects that implicitly encode the graph and its edge weights. Our goal in devising efficient geometric algorithms is generally to avoid explicit construction of the entire underlying graph, since the full induced graph may be very large (even exponential in the input size, or infinite).

Computing an optimal path in a geometric domain is a fundamental problem in computational geometry, having many applications in robotics, geographic information systems (GIS) (see [134]), wire routing, etc. The most basic form of the problem is: Given a collection of obstacles, find a Euclidean shortest *obstacle-avoiding* path between two given points. A much broader collection of problems is defined by considering the several parameters that define the problem, including the

objective function: How do we measure the "length" of a path? Options include the Euclidean length, L_p length, "link distance", etc.

constraints on the path: Are we simply to get from point s to point t, or must we also visit other points or other regions along a path or cycle?

input geometry: What types of "obstacles" or other entities are specified in the input map?

dimension of the problem: Are we in 2-space, 3-space, or higher dimensions?

type of moving object: Are we moving a single point along the path, or is the robot specified by some more complex geometry?

single shot vs. repetitive mode queries: Do we want to build an effective data structure for efficient queries?

static vs. dynamic environments: Do we allow obstacles to be inserted or deleted, or do we allow obstacles to be moving along known trajectories?

exact vs. approximate algorithms: Are we content with an answer that is guaranteed to be within some small factor of optimal?

*jsbm@ams.sunysb.edu; http://www.ams.sunysb.edu/~jsbm/. Department of Applied Mathematics and Statistics, State University of New York, Stony Brook, NY 11794-3600. This research was largely conducted while the author was a Fulbright Research Scholar at Tel Aviv University. The author is also partially supported by NSF grant CCR-9504192, and by grants from Boeing Computer Services, Bridgeport Machines, Hughes Aircraft, and Sun Microsystems.

known vs. unknown map: Is the complete geometry of the map known in advance, or is it discovered on-line, using some kind of sensor?

In this survey chapter, we discuss several forms of the geometric shortest path problem, primarily for a single point moving in a 2- or 3-dimensional space. We assume that the map of the environment is *known*, except in Section 5, where we discuss on-line path planning problems.

We also discuss other geometric network optimization problems, including minimum spanning trees, Steiner trees, and the traveling salesperson problem. Many versions of these problems are known to be NP-hard; thus, much of our attention is devoted to approximation algorithms.

We focus mostly on sequential algorithms in this survey, listing only a few results on parallel algorithms. See the surveys by Atallah [43] and by Reif and Sen [336] in this handbook, or the survey by Goodrich [179] for more extensive lists of results on parallel algorithms in geometry.

We will freely use the "big-Oh" notation for upper bounds on time and space requirements. We also use "big-Omega" notation for lower bounds. (See [123] for definitions.) We use "$\tilde{O}(\cdots)$" to indicate an upper bound in which we suppress polylogarithmic factors.

Many of the results discussed in this survey are also reported, in a more tabular form, in a survey chapter [288] of the recently released CRC Handbook, edited by Goodman and O'Rourke [178].

Finally, we make a disclaimer that our survey concentrates primarily on *theoretical* results. Some of these results may well imply practical algorithms that may be implementable and useful; however, in many cases, the algorithms are too complex or have too large of a constant buried in the big-Oh notation to be of practical significance. We hope that a future survey will address the important choices and issues facing practitioners in the implementation of geometric shortest path and network optimization algorithms. One of the major issues facing an implementer of any geometric algorithm is, of course, *robustness*; see the survey by Schirra [353] in this handbook.

Shortest Paths in Graphs

Shortest paths in graphs and networks are well studied; see, e.g., Ahuja, Magnanti, and Orlin [10]. Here, we mention the case in which all edge weights are non-negative, as this is the most relevant for geometric instances. Then, a standard algorithm given by Dijkstra [138] allows one to compute a tree of shortest paths from any one source node to all other nodes of the graph. Early implementations of Dijkstra's algorithm required time $O(v^2)$ or $O(e \log v)$, where v denotes the number of vertices and e the number of edges. Using Fibonacci heaps, Fredman and Tarjan [160] gave an $O(e + v \log v)$ time implementation, and argued that this is optimal in a comparison-based model of computation. Exploiting planarity, Henzinger, Klein, and Rao [198] have obtained a *linear-time* algorithm for computing all shortest paths from a single source in planar graphs having nonnegative edge weights.

There has been some recent progress too in devising new algorithms that differ from Dijkstra's algorithm in that they do not necessarily visit nodes in increasing order of distance from the source node. Thorup [374] has in fact obtained an optimal $O(e)$-time algorithm for computing a tree of shortest paths in a graph having integer edge weights; see his paper, as well as the recent article of Raman [328], for a survey of other recent results that led up to this one.

Approximation Algorithms

Several of the problems we will discuss in this survey are "provably hard" (e.g., NP-hard), meaning that no polynomial-time algorithm is known to exist to solve it. An increasingly popular approach to "solving" NP-hard optimization problems is to obtain provably-good approximation algorithms, which are guaranteed, in polynomial time, to produce an answer that is close to optimal – say, whose objective function value at most some factor $c > 1$ times optimal, for a minimization problem. Such an approximation algorithm is then called a *c-approximation algorithm.* (For a *maximization* problem, a c-approximation algorithm produces a solution whose objective function value is *at least* $(1/c)$ times optimal.)

A *polynomial time approximation scheme* (PTAS) is a method that allows one to compute a $(1+\epsilon)$-approximation to the optimal (minimum), in time that is polynomial in n, for any fixed $\epsilon > 0$. (In general, the dependence on ϵ may be exponential in $(1/\epsilon)$.)

The recent book edited by Hochbaum ([210]) contains several articles surveying the state of knowledge on approximation algorithms for NP-hard problems. In particular, the survey of Bern and Eppstein [64] gives an excellent overview of the subject of approximating NP-hard geometric optimization problems.

Approximation algorithms can also be quite useful for problems that are not necessarily NP-hard. First, an approximation algorithm may be considerably simpler and easier to implement than an algorithm that solves the problem to optimality. Further, the running time (both worst-case and average-case) for the approximation algorithm may be much better than the best known for the exact solution, even when the exact algorithm has polynomial running time.

Further, approximation algorithms are known for some problems whose complexity status is still open, such as the MAX TSP in the plane and the minimum-weight triangulation problem; see Section 7.

Geometric Preliminaries

Throughout the survey, we will have need of some basic terminology, which we outline in this section.

First, a *path* is a continuous image of an interval. A *polygonal s-t path* is a path from point s to point t consisting of a finite number of line segments (*edges*, or *links*) joining a sequence of points (*vertices*).

The *length* of an *s-t* path is a nonnegative number associated with the path, measuring its total cost according to some prescribed metric. Unless otherwise specified, the length will be the Euclidean length of the path.

A *shortest path* is then a path of minimum length among all paths that are feasible (satisfying all imposed constraints). We often refer to a shortest path also as an "optimal path" or a "geodesic path". (The word "geodesic" is sometimes used differently, to refer to paths that are "locally optimal", as defined below.)

The shortest-path problem induces a metric, the *shortest path metric*, in which the distance between two points s and t is given by the length of a shortest *s-t* path; in many geometric contexts, this metric is also referred to as *geodesic distance*.

A *simple polygon*, P, having n vertices, is a closed, simply-connected region whose boundary is a union of n (straight) line segments (edges), whose endpoints are the vertices of P. A *polygonal domain*, P, having n vertices and h holes, is a closed, multiply-connected region whose boundary is a union of n line segments, forming $h+1$ closed (polygonal) cycles. (A simple polygon is a polygonal domain with $h = 0$.)

A *triangulation* of P is a decomposition of P into triangles such that any two triangles either intersect in a common vertex, a common edge, or not at all. A triangulation of a simple polygon P can be computed in $O(n)$ time [91]; a polygonal domain can be triangulated in time $O(n \log n)$ [327] or $O(n + h \log^{1+\epsilon} h)$ [54] time. (See the chapter of Bern and Plassman [66] in this handbook, or the survey by Bern [63] for more information on triangulations.)

We will use the term *obstacle* to refer to any region of space whose interior is forbidden to paths. The complement of the set of obstacles is the *free space*. If the free space is a polygonal domain P, the obstacles are the $h+1$ connected components of the complement of P (h *holes*, plus the *face at infinity*).

A path that cannot be improved by making a small change to it that preserves its *combinatorial structure* (e.g., the ordered sequence of triangles visited, for some triangulation of a polygonal domain P) is called a *locally shortest* or *locally optimal* path. It is also known as a *taut-string* path in the case of a shortest obstacle-avoiding path.

The *visibility graph*, $VG(P)$, is a graph whose nodes are the vertices of P and whose edges join pairs of nodes for which the corresponding segment lies inside P. An example is shown in Figure 2.

Given a *source point*, s, a *shortest path tree*, $\mathrm{SPT}(s, P)$, is a spanning tree of s and the vertices of P such that the (unique) path in the tree between s and any vertex of P is a shortest path in P.

A *single-source query* is a type of shortest path problem in which a source point, s, is fixed, and for each *query (goal) point*, t, one requests the length of a shortest path from the source point s to t. The query may also require the retrieval of an actual instance of a shortest s-t path; in general, this can be reported in additional time $O(k)$, where k is the complexity of the output (e.g., number of edges).

One method of handling the single-source query problem is to construct a *shortest path map*, SPM(s), which is a decomposition of free space into regions (*cells*) according to the "combinatorial structure" of shortest paths from a fixed source point s to points in the regions. Specifically, for shortest paths in a polygonal domain, SPM(s) is a decomposition of P into cells such that for all points t interior to a cell, the sequence of obstacle vertices along an s-t path is fixed. In particular, the *last* obstacle vertex along a shortest s-t path is the *root* of the cell containing t. Each cell is *star-shaped* with respect to its root, which lies on the boundary of the cell, meaning that the root can "see" all points within the cell. Typically, we will store with each vertex, v, of P the geodesic distance, $d(s,v)$, from s to v, as well as a pointer to the *predecessor* of v, which is the vertex (possibly s) preceding v in a shortest path from s to v. (The predecessor pointers provide an encoding of the SPT(s,P).) Note that v will appear on the boundary of the star-shaped cell rooted at its predecessor. The boundaries of cells consist of portions of obstacle edges, *extension segments* (extensions of visibility graph edges incident on the root), and *bisector curves*. The bisector curves are, in general, hyperbolic arcs that are the locus of points p that are (geodesically) equidistant from two roots, u and v: they satisfy $d(s,u)+d_2(u,p)=d(s,v)+d_2(v,p)$, where $d_2(\cdot,\cdot)$ denotes Euclidean distance. (Extension segments can be considered to be degenerate cases of bisector curves.) In Figure 1, the root of the cell containing t is labeled r. If SPM(s) is preprocessed for point location (see the chapter by Goodrich [180] in this handbook), then single-source queries can be answered efficiently by locating the query point t within the decomposition: If t lies in the cell rooted at r, the geodesic distance to t is given by $d(s,t)=d(s,r)+d_2(r,t)$. A shortest s-t path can then be output in time $O(k)$, where k is the number of vertices along the path, by simply following predecessor pointers back from r to s.

In a *two-point query* problem, we are asked to construct a data structure that allows us to answer efficiently a query that specifies two points, s and t, and requests the length of a shortest path between them. In all cases discussed here, an actual instance of a shortest path can be reported in additional time $O(k)$, where k is the complexity of the output (e.g., number of edges).

A *geodesic Voronoi diagram* (VD) is a Voronoi diagram for a set of *sites*, in which the underlying metric is the geodesic distance. See the chapter of Aurenhammer and Klein [45] in this handbook for details about Voronoi diagrams.

The *geodesic center of P* is a point within P that minimizes the maximum of the shortest-path lengths to any other point in P. The *geodesic diameter of P* is the maximum of the lengths of the shortest paths joining pairs of vertices of P.

Finally, we remark that in most of the algorithmic results reported here, the model of computation assumed has been the real RAM, which assumes that exact operations on real numbers can be done in constant time per operation. We acknowledge that this model is not, in general, realistic. At a couple places in the survey, we will point to results involving bit complexity models.

2 Geodesic Paths in a Simple Polygon

We begin by considering the most basic geometric shortest-path problem, that of finding a shortest s-t path inside a *simple* polygon, P (having no "holes"). The complement of P serves as an "obstacle" through which the path is not allowed to travel. In this case, simple local optimality arguments, based on the triangle inequality, yield:

Proposition 1 *There is a unique shortest s-t path in a simple polygon P; consequently, $SPT(s,P)$ is unique.*

We now sketch an $O(n)$ time algorithm for computing a shortest s-t path within a simple polygon P. We begin with a triangulation of P ($O(n)$ time; [91]), whose dual graph is a tree. The *sleeve* is comprised of the triangles that correspond to the (unique) path in the dual that joins the triangle containing s to that containing t. By considering the effect of adding the triangles in order along the sleeve, [89, 252] have shown how to obtain an $O(n)$-time algorithm for collapsing the sleeve into a shortest path. At a generic step of

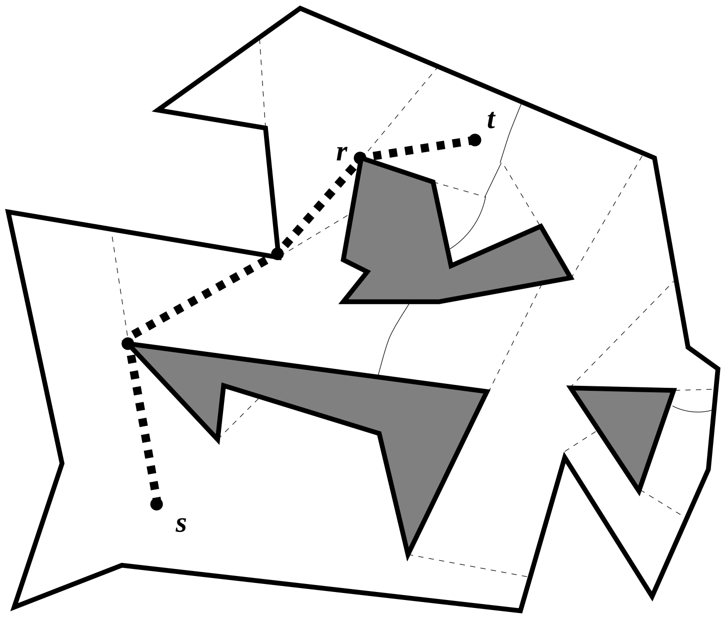

Figure 1: A shortest path map with respect to source point s within a polygonal domain with $h = 3$. The heavy dashed path indicates the shortest s-t path, which reaches t via the root r of its cell. Bisector curves are shown in narrow solid curves; extension segments are shown thin and dashed.

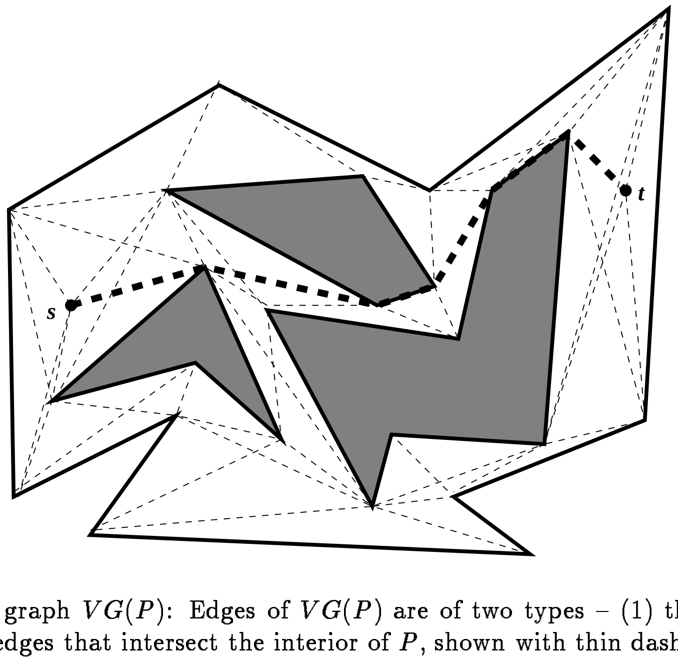

Figure 2: The visibility graph $VG(P)$: Edges of $VG(P)$ are of two types – (1) the heavy dark boundary edges of P, and (2) the edges that intersect the interior of P, shown with thin dashed segments. A shortest s-t path is highlighted.

the algorithm, the sleeve has been collapsed to a structure called a "funnel" (with "base" ab and "root" r) consisting of the shortest path from s to a vertex r, and two (concave) shortest paths joining r to the endpoints of the segment ab that bounds the triangle abc that is about to be considered (see Figure 3). In adding triangle abc, we "split" the funnel in two according to the taut-string path from r to c, which will, in general, include a segment, uc, joining c to some (vertex) point of tangency, u, along one of the two concave chains of the funnel. After the split, we keep that funnel (with base ac or bc) that contains the s-t taut-string path. The work needed to search for u can easily be charged off to those vertices that are discarded from further consideration. The end result is that a shortest s-t path is found in time $O(n)$, which is worst-case optimal.

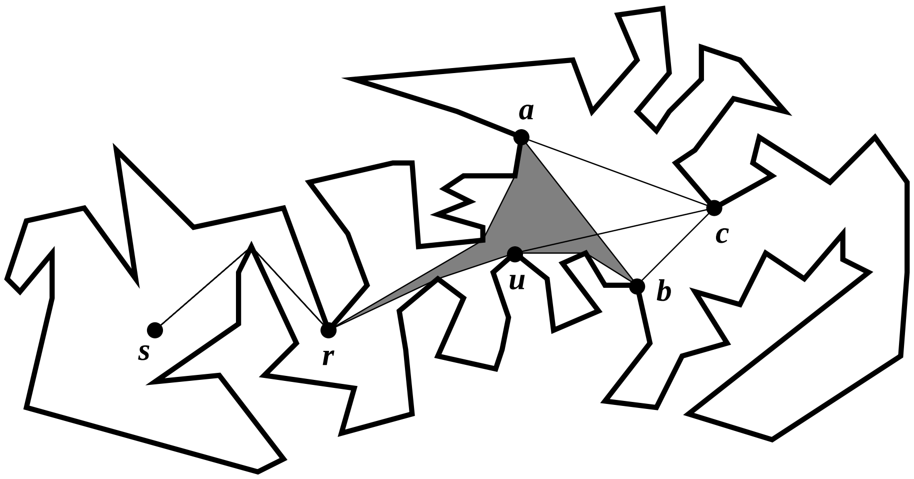

Figure 3: Computing a shortest path in a simple polygon: Splitting a funnel.

In order to answer single-source query problems, we are interested in also computing the shortest path map in P. SPM(s) has a particularly simple structure, as the boundaries between cells in the map are simply (line segment) chords of P obtained by extending appropriate edges of the visibility graph $VG(P)$. Guibas et al. [186] have shown how it can be computed in time $O(n)$, by using somewhat more sophisticated data structures to do funnel splitting efficiently (since, in this case, we cannot discard one side of each split funnel). Then, after storing the SPM(s) in an appropriate $O(n)$-size point location data structure (see, e.g., [180]), single-source queries can be answered in $O(\log n)$ time. Hershberger and Snoeyink [203] have substantially simplified the original algorithm of [186].

The above result can be strengthened even further to the case of two-point queries. Guibas and Hershberger [185] have shown how a simple polygon can be preprocessed in time $O(n)$, into a data structure of size $O(n)$, to support shortest-path queries between any two points $s, t \in P$. In time $O(\log n)$ the length of the shortest path can be reported, and in additional time $O(k)$, the shortest path can be reported, where k is the number of vertices in the output path. The method has been simplified with a new data structure introduced by Hershberger [200].

Theorem 2 ([185, 200]) *For a simple polygon P having n vertices, there is a data structure of size $O(n)$ that can be built in time $O(n)$ so that the length of the shortest path between any two points $s, t \in P$ can be reported in time $O(\log n)$, and the shortest path itself can be reported in additional time proportional to its number of vertices.*

We should emphasize that the above methods all rely on starting with a triangulation of the simple polygon. Given the complexity of linear-time triangulations of polygons, we pose the following open problem:

Open Problem 1 *Can one devise a simple $O(n)$ time algorithm for computing the shortest path between two points in a simple polygon,* without *resorting to a (complicated) linear-time triangulation algorithm?*

In the *dynamic* version of the shortest path problem, one allows the polygon P to change, with the addition or deletion of edges and vertices. If the changes are always made in such a way that the set of all edges yields a *connected planar subdivision* of the plane into simple polygons (i.e., no "islands" are created), then one can maintain a data structure of size $O(n)$ that supports two-point query time of $O(\log^2 n)$ (plus $O(k)$ if the path is to be reported), and update time of $O(\log^2 n)$ for each addition/deletion of an edge/vertex [183]. (The result of [183] improves the first results on the dynamic problem, obtained by Chiang, Preparata, and Tamassia [108, 109], who gave a data structure achieving $O(\log^3 n)$ query and update bounds, using $O(n \log n)$ space. The same data structure also gives the best known dynamic point location solution for connected maps, with optimal $O(\log n)$ query time.)

We turn briefly to some results on parallel algorithms. ElGindy and Goodrich [147] gave a parallel algorithm to compute a shortest path in a simple polygon in time $O(\log n)$, using $O(n)$ processors (in the CREW PRAM model). Goodrich, Shauck, and Guha [181, 182] show how, with $O(n/\log n)$ processors and $O(\log n)$ time, one can compute a data structure that supports $O(\log n)$ (sequential) time shortest-path queries between pairs of points in a simple polygon. They also give an $O(\log n)$-time algorithm using $O(n)$ processors to compute a shortest path tree. Hershberger [201] builds on the results of [181, 182] and gives an algorithm for shortest path trees requiring only $O(\log n)$ time and $O(n/\log n)$ processors (CREW); he also obtains optimal parallel algorithms for related visibility and geodesic distance problems.

Other Geodesic Distance Problems

The geodesic Voronoi diagram of k sites inside P can be constructed in time $O((n+k)\log(n+k))$, using $O(n)$ space [320]; this improves an earlier result of Aronov [32] that required time $O((n+k)\log(n+k)\log n)$. The furthest-site Voronoi diagram for geodesic distance can also be computed in time $O((n+k)\log(n+k))$, and space $O(n+k)$, using an algorithm of Aronov, Fortune, and Wilfong [33]. Given that shortest paths in simple polygons require only linear time, it is natural to ask if the superlinear portion of the complexities of these algorithms can be moved to the "k" term; the only lower bound known is $\Omega(n + k\log k)$.

Open Problem 2 *Can the geodesic Voronoi diagram (closest-site or furthest-site) for k sites within a simple polygon P be computed in time $O(n + k\log k)$?*

The geodesic diameter of a simple polygon can be computed in time $O(n)$, using the method of "matrix searching" in the geodesic distance, as developed by Hershberger and Suri [209]. This algorithm improves an earlier $O(n\log n)$-time solution given by Suri [366, 185]. Matrix searching also provides a powerful tool for obtaining linear-time solutions to other geodesic distance problems, such as all nearest neighbors and all furthest neighbors.

The geodesic center of a simple polygon P can be computed in time $O(n\log^2 n)$ [326] (see also [41]); however, it is believed that this bound can be improved.

Open Problem 3 *Can the geodesic center of a simple polygon be computed in $O(n)$ time?*

Shortest paths within simple polygons give a wealth of structural information about the polygon. In particular, they have been used to give an output-sensitive algorithm for constructing the visibility graph of a simple polygon ([199]) and can be used for constructing a *geodesic triangulation* of a simple polygon, which allows for efficient ray-shooting (see [92, 207]). They also form a crucial step in solving *link distance* problems (Section 4.2).

3 Paths in a Polygonal Domain

In contrast to the situation in simple polygons, where there is a *unique* taut-string path between any two points, in a general polygonal domain P, there can be an exponential number of taut-string (locally optimal) simple paths between two points.

A special case of the shortest path problem in polygonal domains is that in which the "homotopy type" of the desired path is specified, e.g., by giving the sequence (possibly with repetitions) of the N visited triangles, in some triangulation of P. In this case, Hershberger and Snoeyink [203] have shown how to compute a shortest path of the given homotopy type in time $O(N)$, using a generalization of the linear-time methods in simple polygons. This problem is of interest in applications to VLSI routing problems; see [122, 162, 256].

To compute a shortest path in general polygonal domains, with no constraints on the homotopy type, we must efficiently search over all possible "threadings" of paths. We discuss two methods below that have been used to do so: searching the visibility graph (see Figure 2), and performing a "continuous Dijkstra" search of the domain.

Searching the Visibility Graph

Since we can make "point" holes in P at s and t, we can assume, without loss of generality, that s and t are vertices of P. Using simple local optimality arguments, it is easy to show:

Proposition 3 *Any locally optimal s-t path in a polygonal domain P must lie on the visibility graph $VG(P)$; it consists of a union of straight line segments joining pairs of visible vertices.*

Early algorithms to construct the visibility graph required time $O(n^2 \log n)$ [251], and were based on a radial sweep about each vertex of P. The time complexity came from the use of n independent radial sortings of the vertices. Later improvements by Welzl [385] and by Asano et al. [38] gave a time bound $O(n^2)$. These methods were based on the use of point-line duality, which allowed the n sortings to be done more efficiently, in $O(n^2)$ time overall, by constructing the arrangement of the n lines that are dual to the vertices of P. But, given that the number, E_{VG}, of edges in the visibility graph may be much smaller than its worst-case quadratic size (in particular, E_{VG} may be only linear in n), researchers pursued "output-sensitive" algorithms to compute it in time that is a function of E_{VG}. Hershberger [199] studied the special case of visibility graphs in simple polygons, obtaining an $O(E_{VG})$-time and $O(n)$-space algorithm to compute the visibility graph of a simple polygon. Overmars and Welzl [313] obtained a relatively simple $O(E_{VG} \log n)$-time method, requiring $O(n)$ space. Then, Ghosh and Mount [173] obtained an algorithm with worst-case optimal running time, $O(E_{VG} + n \log n)$, using $O(E_{VG})$ working storage space. More recently, Pocchiola and Vegter [324, 325] and Rivière [346] have given algorithms to compute the visibility graph in optimal time $(O(E_{VG} + n \log n))$ and optimal space $(O(n))$.

Once we have computed the graph $VG(P)$, whose edges are weighted by their Euclidean lengths, we can use Dijkstra's algorithm[1] to construct a tree of shortest paths from s to all vertices of P, in time $O(E_{VG} + n \log n)$ [160, 141]. Thus, Euclidean shortest paths among obstacles in the plane can be computed in time $O(E_{VG} + n \log n)$. This bound is worst-case quadratic in n, since $E_{VG} \leq \binom{n}{2}$; note too that domains exist with $E_{VG} = \Omega(n^2)$.

If our goal is to obtain the shortest path map, then, given the tree of shortest paths from s, we can compute SPM(s) in time $O(n \log n)$ [282].

Another method based on visibility graphs leads to an algorithm whose running time is only *linear* in n, while being quadratic in the number, h, of holes in P. Kapoor, Maheshwari, and Mitchell [237] have given an $O(n + h^2 \log n)$-time, $O(n)$-space algorithm, using visibility graph techniques developed by Rohnert [348, 347] for convex obstacles, and visibility "corridor" structure developed by Kapoor and Maheshwari [236].

There has been an effort for many years to *characterize* which graphs correspond to visibility graphs of some geometric domain. For example, it is an interesting open problem to characterize the class of graphs that can be realized as the visibility graph of a simple polygon; see, e.g., Abello and Kumar [1], Ghosh [172], and O'Rourke and Streinu [312] for some recent results and some pointers to related work.

[1]In practice, it may be faster to apply the A* heuristic search algorithm (e.g., see Pearl [322]), using the straight-line Euclidean distance as heuristic function, $h(\cdot)$ (which is a lower bound, so it implies an "admissible" algorithm).

For further information on visibility, visibility graphs, and their use in shortest path problems, we refer the reader to the survey of Alt and Welzl [20], the survey (on visibility) by O'Rourke [310], and the chapter on visibility by Ghosh [171] in this handbook.

Continuous Dijkstra Method

Instead of searching the visibility graph (which may have quadratic size), an alternative paradigm for shortest-path problems is to construct the (linear-size) shortest path map directly. The *continuous Dijkstra* method [278, 279, 280, 282, 283, 291, 292] was developed for this purpose.

Building on the success of the method in solving (in nearly linear time) the shortest-path problem for the L_1 metric (see Section 4.1), Mitchell [284, 286] developed a version of the continuous Dijkstra method applicable to the Euclidean shortest-path problem, obtaining the first subquadratic ($O(n^{3/2+\epsilon})$) time bound. Subsequently, this result was improved by Hershberger and Suri [205, 206], who achieve a nearly optimal algorithm based also on the continuous Dijkstra method. They give an $O(n \log n)$ time and $O(n \log n)$ space algorithm, coming close to the lower bounds of $\Omega(n + h \log h)$ time and $O(n)$ space.

The continuous Dijkstra paradigm involves simulating the effect of a "wavefront" propagating out from the source point, s. The *wavefront* at distance δ from s is the set of all points of P that are at geodesic distance δ from s. It consists of a set of curve pieces, called *wavelets*, which are arcs of circles, centered at obstacle vertices that have already been reached. At certain critical "events," the structure of the wavefront changes due to one of the following possibilities:

(1) a wavelet disappears (due to the "closure" of a cell of the SPM); or

(2) a wavelet collides with an obstacle vertex; or

(3) a wavelet collides with another wavelet; or

(4) a wavelet collides with an obstacle edge at a point interior to that edge.

It is not difficult to see from the fact that SPM(s) has linear size that the total number of such events is $O(n)$. The challenge in applying this propagation scheme is in devising an efficient method to know *what* events are going to occur and in being able to *process* each event as it occurs (updating the combinatorial structure of the wavefront).

One approach, used in [284, 286], is to track a "pseudo-wavefront," which is allowed to run over itself, and "clip" only when a wavelet collides with a vertex that has already been labeled due to an earlier event. Detection of when a wavelet collides with a vertex is accomplished with range searching techniques, at a cost of $O(n^{0.5+\epsilon})$ per query. This leads to an overall running time of $O(n^{3/2+\epsilon})$, for any fixed $\epsilon > 0$, using $O(n)$ space.

An alternative approach, used in [205, 206], simplifies the problem by first decomposing the domain P using a "conforming subdivision," which allows one to propagate an "approximate wavefront" on a *cell-by-cell* basis. A key property of a conforming subdivision is that for any edge (of length L) of the subdivision, there are only a *constant* number of (constant-sized) cells within geodesic distance L of it.

While the algorithm of [206] is optimal worst-case time when there are a large number of obstacles (e.g., $h = \Omega(n)$), it fails to be optimal in its space complexity ($O(n \log n)$) and in its complexity as a function of n and h. One of the most intriguing open problems here is to obtain an (optimal) algorithm whose running time asymptotically matches the lower bound of $\Omega(n + h \log h)$, while using only $O(n)$ space. Currently, the only algorithm known that is *linear* in n is also *quadratic* in h [237].

Open Problem 4 *Can one solve the Euclidean shortest-path problem in $O(n + h \log h)$ time and $O(n)$ space?*

Approximation Algorithms

Efficient methods to approximate the Euclidean shortest path, in time $O(n \log n)$, have existed for some time. Clarkson [119] gave an algorithm that spent $O((n \log n)/\epsilon)$ time to build a data structure of size $O(n/\epsilon)$, after which a $(1+\epsilon)$-approximate shortest path query could be answered in time $O(n \log n + n/\epsilon)$. (These bounds rely also on an observation in [93].) Using a related approach, based on approximating Euclidean distance with fixed orientation distances (see Section 4.1), Mitchell [279, 283] gave a method requiring $O((n \log n)/\sqrt{\epsilon})$ time and $O(n/\sqrt{\epsilon})$ space to give an approximate Euclidean shortest path. Chen, Das, and Smid [95] have shown an $\Omega(n \log n)$ lower bound, in the algebraic computation tree model, on the time required to compute a $(1+\epsilon)$-approximate shortest path; they also give $\Omega(n \log n)$ lower bounds on computing various types of "t-spanners," which are graphs that, for every pair of points, contain a path whose length is at most t times the interpoint distance (Euclidean, geodesic, etc.); see the survey on spanners in this handbook by Eppstein [150], as well as [79, 106, 351].

Two-Point Queries

Two-point queries in a polygonal domain are much more challenging than the case of simple polygons, where optimal algorithms are known.

One approach, observed by Chen, Daescu, and Klenk [94], is to proceed as follows. Using $O(n^2)$ space, we can store the shortest path map, SPM(v, P), rooted at all n vertices. Then, for any s and t, we can use the visibility complex of Pocchiiola and Vegter [323] to compute the set of k_s vertices visible to s and k_t vertices visible to t, in time $O(K \log n)$, where $K = \min\{k_s, k_t\}$ (using a standard "lock step" computation of the visibility from the two points). Then, assuming that $K = k_s$, we simply locate t in each of the k_s SPM's rooted at the vertices visible from s. This permits two-point queries to be answered in time $O(K \log n)$, which is $\Omega(n \log n)$ in the worst case, making this method no better than starting the computation from scratch. However, this approach may be effective in cases in which K may be expected to be small.

A recent study by Chiang and Mitchell [107] has yielded more efficient query times, with various tradeoffs between preprocessing time and space. They use a visibility-based approach to achieve query time $O(\log n+h)$ using $O(n^5)$ preprocessing time and space. They also achieve optimal query time, $O(\log n)$, using high polynomial space (roughly n^{10}), and they achieve slightly sublinear query time, using $O(n^{5+\epsilon})$ space. These results utilize an "equivalence decomposition" of the domain P, so that for all points z within a cell of the decomposition, the shortest path maps with respect to z are topologically equivalent. Then, for given query points s and t, one locates s within the decomposition, and then uses the resulting SPM, along with a parametric point location data structure, to locate t within the SPM with respect to s. The complexity of the decomposition can be quite high; there can be $\Omega(n^4)$ topologically distinct shortest path maps with respect to points within P. Unfortunately, the upper bound on the complexity of the equivalence decomposition is still considerably higher than this; obtaining tight bounds remains an interesting open question.

Approximations have also been useful in attacking the two-point query problem. As observed in [93], the method of Clarkson [119] can be used to construct a data structure of size $O(n^2 + n/\epsilon)$ in $O(n^2 \log n + (n/\epsilon) \log n)$ time, so that two-point $(1+\epsilon)$-optimal queries can be answered in time $O((\log n)/\epsilon)$, for any fixed $\epsilon > 0$. Chen [93] was the first to obtain nearly *linear*-space data structures for approximate shortest path queries; these were obtained, though, at the cost of a higher approximation factor. He obtains a $(6+\epsilon)$-approximation, using $O(n^{3/2}/\log^{1/2} n)$ time to build a data structure of size $O(n \log n)$, after which queries can be answered in time $O(\log n)$. (Within this time bound, the approximate length is reported; in additional time proportional to the number of vertices, a path can be reported that achieves the length bound.) These results have been improved recently by Arikati et al. [21], who give a family of results, based on planar spanners (see [150]), with tradeoffs among the approximation factor and the preprocessing time, storage space, and query time. One such result obtains a $(3\sqrt{2}+\epsilon)$-approximation using $O(n^{3/2}/\log^{1/2} n)$ time to build a data structure of size $O(n \log n)$, after which queries are performed in time $O(\log n)$. For other results, and for bounds that apply to other metrics (L_p metrics), we refer the reader to the paper.

Open Problem 5 *How efficiently, and using what size data structure, can one preprocess a polygonal do-*

main for exact two-point queries? Can exact two-point queries be done in sublinear query time using subquadratic storage? Can $O(1)$-approximate two-point queries be done in polylogarithmic time, using nearly linear storage?

Other Geodesic Distance Problems

The geodesic Voronoi diagram of k sites inside P can be constructed in time $O((n+k)\log(n+k))$, using the continuous Dijkstra method, simply starting with multiple source points [206].

While the geodesic center/diameter problem has been carefully examined for the case of simple polygons (Section 2), we are unaware of results (other than brute force) for polygonal domains:

Open Problem 6 *How efficiently can one compute a geodesic center/diameter for a polygonal domain?*

4 Shortest Paths in Other Metrics

So far, we have considered only shortest path problems in the Euclidean metric. We turn now to other possible objective functions for measuring the length of a path.

4.1 L_1 Metric

The *L_p metric* defines the distance between $q=(q_x,q_y)$ and $r=(r_x,r_y)$ by $d_p(q,r)=[|q_x-r_x|^p+|q_y-r_y|^p]^{1/p}$. The L_p length of a polygonal path is the sum of the L_p lengths of each edge of the path. Special cases of the L_p metric include the L_1 metric (*Manhattan metric*) and the L_∞ metric ($d_\infty(q,r)=\max\{|q_x-r_x|,|q_y-r_y|\}$).

A polygonal path with each edge parallel to a coordinate axis is called a *rectilinear* (or *isothetic*) path. (For a rectilinear path, the L_1 and L_2 lengths are identical.) A natural generalization of the notion of a rectilinear path is that of *C-oriented* paths, having each edge parallel to one of a set C of $c=|C|$ *fixed orientations.* (See Widmayer, Wu, and Wong [386], who initiated the study of fixed orientation metrics in computational geometry.)

As with Euclidean shortest paths, algorithms for computing shortest paths in the L_1 metric fall into two general categories: searching a sparse "path preserving graph" (analogous to a visibility graph), or applying the continuous Dijkstra paradigm or tracking a wavefront.

Clarkson, Kapoor, and Vaidya [121] showed how to construct a sparse graph, having $O(n\log n)$ nodes and $O(n\log n)$ edges, that is *path preserving* in that it is guaranteed to contain a shortest path between any two vertices. Applying Dijkstra's algorithm then gives an $O(n\log^2 n)$ time ($O(n\log n)$ space) algorithm for L_1 shortest paths. (Alternatively, one gets $O(n\log^{3/2} n)$ time and $O(n\log^{3/2} n)$ space.) Using observations in [97, 98] the time-space tradeoff has been improved to yield somewhat improved bounds of $O(n\log^{3/2} n)$ time and $O(n\log n)$ space.

The continuous Dijkstra paradigm has also been applied to the L_1 shortest path problem, resulting in the computation of the SPM(s) in time $O(n\log n)$, using $O(n)$ space [279, 283]. The special property of the L_1 metric that is exploited in this algorithm is the fact that the wavefront in this case is piecewise-linear, with "wavelets" that are line segments of slope ± 1, so that the first vertex hit by a wavelet can be determined efficiently using rectangular range searching techniques (e.g., see [90]).

Two-point query problems have also been studied for the L_1 geodesic metric. In a simple rectilinear polygon, Lingas, Maheshwari, and Sack [259] and Schuierer [354] give optimal algorithms, achieving $O(\log n)$ query time ($O(1)$ for vertex-to-vertex queries), using $O(n)$ preprocessing time and space; an optimal path can be reported in additional $O(k)$ time, where k is the number of links. (A previous algorithm of de Berg [130] achieved optimal query time using $O(n\log n)$ space and preprocessing.) Their methods are based upon a histogram decomposition of the polygon and yield a path that is "smallest" – simultaneously optimal in both the L_1 and rectilinear link metric (see also [157, 274], as well as Section 4.7). They also yield an $O(n)$ algorithm for computing the L_1 geodesic diameter and furthest neighbors for all vertices. Further, the algorithm of Lingas, Maheshwari, and Sack [259] is actually based on an optimal *parallel* (EREW PRAM) algorithm that preprocesses a polygon (with a given trapezoidization) in time $O(\log n)$, using $O(n/\log n)$ processors.

Two-point queries in a polygonal domain, under the L_1 metric, have been studied by Chen, Klenk, and Tu [97, 98], who have shown how a polygonal domain can be preprocessed, using $O(n^2 \log^2 n)$ time and $O(n^2 \log n)$ space, so that two-point queries can be answered in time $O(\log^2 n)$. The special case in which obstacles are disjoint axis-aligned rectangles has been studied by Atallah and Chen [44, 42] and by ElGindy and Mitra [148]; $O(\log n)$ query time is achievable, using $O(n^2)$ preprocessing time and space, or $O(\sqrt{n})$ query time is achievable, using $O(n^{3/2})$ preprocessing time and space. In fact, they give parallel algorithms: with $O(n^2/\log n)$ CREW processors, a data structure of size $O(n^2)$ can be built that permits two-point queries to be answered in time $O(\log^2 n)$ on a single processor ([42]). Mitra and Bhattacharya [298] and Chen and Klenk [96] have obtained approximation algorithms in the special case of disjoint rectangular obstacles; [96] describe a method achieving $O(\log n)$ query time for a 3-approximate query, using $O(n \log n)$ space and $O(n \log^2 n)$ preprocessing time. (If the query points are both obstacle vertices, then the query time is only $O(1)$.) Arikati et al. [21] have recently obtained approximation results for two-point queries in polygonal domains, as we mentioned already in the Euclidean case. Their results apply also to L_p metrics, where they obtain various tradeoffs between space and time resources, to achieve approximation factors that are $c + \epsilon$, $2c + \epsilon$, or $3c + \epsilon$, where $c = 2^{(p-1)/p}$ (so $c = 1$ for the L_1 metric).

Methods for finding L_1 shortest paths often generalize to the case of C-oriented paths, in which $c = |C|$ fixed directions are given. Shortest C-oriented paths can be computed in time $O(cn \log n)$ [279, 283]. Two-point queries can be answered in query time $O(c^2 \log^2 n)$, after $O(c^2 n^2 \log^2 n)$ time and space preprocessing [94]. Since the Euclidean metric is approximated to within accuracy $O(1/c^2)$ if we use c equally spaced orientations, this results in an algorithm to compute, in time $O((n/\sqrt{\epsilon}) \log n)$, a path guaranteed to have length within a factor $(1 + \epsilon)$ of the Euclidean shortest path length [279, 283]. Clarkson [119] gave an alternative approximation algorithm based also on discretizing directions that computes an ϵ-optimal (Euclidean) shortest path in time $O(n/\epsilon + n \log n)$, after spending $O((n/\epsilon) \log n)$ time to build a data structure of size $O(n/\epsilon)$.

4.2 Link Distance

The *link distance* within P from s to t is the minimum number of edges in an s-t path in P. If the paths are restricted to be rectilinear or C-oriented, then we speak of the *rectilinear link distance* or *C-oriented link distance*. A *min-link s-t path* is a polygonal path from s to t that achieves the link distance.

In many problems, the link distance provides a more natural measure of path complexity than the Euclidean length. The link distance also has applications to curve simplification [187, 222, 295].

Since this handbook contains a chapter by Maheshwari and Sack [271] devoted entirely to the subject of link distance, we refer the reader to that survey for further information.

4.3 The Weighted Region Metric

In the "weighted region problem", we are given a piecewise-constant function, $f : \Re^2 \rightarrow \Re$, that is defined by assigning a nonnegative *weight* to each face of a given triangulation in the plane. The *weighted length* of an s-t path π is the path integral, $\int_\pi f(x,y) d\sigma$, of the weight function along π. The *weighted region metric* associated with f defines the distance $d_f(s,t)$ to be the infimum over all s-t paths π of the weighted length of π. The *weighted region problem* (WRP) asks for an s-t path of minimum weighted length.

The WRP is a natural generalization of the shortest-path problem in a polygonal domain: Consider a weight function that assigns weight 1 to P and weight ∞ (or a sufficiently large constant) to the obstacles (the complement of P). The WRP models the minimum-time path problem for a point robot moving in a terrain of varied types (e.g., grassland, brushland, blacktop, bodies of water, etc), where each type of terrain has an assigned weight equal to the reciprocal of the maximum speed of traversal for the robot.

We usually assume that f is specified by a triangulation having n vertices, with each face assigned an integer weight $\alpha \in \{0, 1, \ldots, W, +\infty\}$. (We can allow edges of the triangulation to have a weight that is possibly distinct from that of the triangular facets on either side of it; in this way, "linear features" such as "roads" can be modeled.) Using an algorithm based on the continuous Dijkstra method, Mitchell and Papadimitriou [292] show how to find a path whose weighted length is guaranteed to be within a factor of $(1+\epsilon)$ of optimal, where $\epsilon > 0$ is any user-specified degree of precision. The time complexity of their algorithm is $O(E \cdot S)$, where E is the number of "events" in the continuous Dijkstra algorithm, and S is the complexity of performing a numerical search to solve the following subproblem: Find a $(1+\epsilon)$-shortest path from s to t that goes through a given sequence of k edges of the triangulation. It is shown that $E = O(n^4)$ and that there are examples where E can actually achieve this upper bound. The numerical search can be done using a form of binary search that exploits the local optimality condition: An optimal path bends according to "Snell's Law of Refraction" when crossing a region boundary. (The earliest reference we have found to the use of Snell's Law in optimal route planning applications is to the work of Warntz [384].) This leads to a bound of $S = O(k^2 \log(nNW/\epsilon))$ on the time needed to perform a search on a k-edge sequence, where N is the largest integer coordinate of any vertex of the triangulation. Since one can show that $k = O(n^2)$, this yields an overall time bound of $O(n^8 L)$, where $L = \log(nNW/\epsilon)$ can be thought of as the bit complexity of the problem instance.

Various special cases of the weighted region problem admit faster and simpler algorithms. In the case that region weights are restricted to $\{0, 1, \infty\}$ (while edges may have arbitrary (nonnegative) weights), then an $O(n^2)$-time algorithm can be based on constructing a path-preserving graph similar to a visibility graph, as shown by Gewali et al. [168]. This also leads to an efficient method for performing *lexicographic* optimization, in which one prioritizes various types of regions according to which is most important for path length minimization. Lee, Yang, and Chen [253] consider the case in which the plane has weight 1, while each of a set of pairwise-disjoint rectilinear polygonal "obstacles" has a weight greater than 1, indicating that it is more costly to travel through it than to go around it. They apply the techniques of [121], searching a path-preserving graph, to obtain an algorithm for minimum-cost rectilinear paths that takes time $O(n \log^2 n)$ (with space $O(n \log n)$) or $O(n \log^{3/2} n)$ (with $O(n \log^{3/2} n)$ space). A path-preserving graph approach can also be applied to the more general case of rectilinear paths in an arbitrarily weighted rectilinear subdivision, to yield efficient algorithms for single-source and two-point queries. Specifically, Chen, Klenk, and Tu [97] give an $O(n \log^{3/2} n)$-time algorithm to construct a data structure of size $O(n \log n)$, permitting $O(\log n)$-time single-source queries to be answered; for two-point queries, they use $O(n^2 \log^2 n)$ space and preprocessing time, and answer queries in time $O(\log^2 n)$.

In recent experimental investigations, Mata and Mitchell [273] and Lanthier, Maheshwari, and Sack [246], have shown the practicality of solving the WRP using very simple methods based on searching a discrete graph which is assured of containing an approximately optimal path. One graph is based on discretizing the edges of the subdivision, placing evenly-spaced new (Steiner) vertices along each edge, with separation at most weighted length δ. The vertices on the boundary of each (convex) facet are interconnected (possibly implicitly) with a complete graph. Searching the resulting graph for a shortest path results in an approximate shortest path; the error is at most $K\delta$, where K is the number of segments in the path. Another option (in [273]) is to construct a "pathnet" graph, based on tracing k evenly-spaced "refraction rays" (that obey Snell's Law) out of each original vertex, and linking that vertex to one vertex (or "critical entry point") within each of the k refraction cones defined by the rays. As k increases, the pathnet more closely approximates a complete set of optimal paths connecting pairs of vertices. The experimental studies suggest that these methods are practical and are readily implementable, and that the observed dependence of the approximation factor on the algorithm parameters (δ or k) is better in practice that the worst-case bounds may suggest. Further, the graphs that are searched can be precomputed and stored, allowing reasonably efficient solutions to two-point queries. The reported path can also be postprocessed with a local optimality procedure that results in a solution even closer to optimal.

Using a slightly different discrete graph than the edge subdivision graph of [246, 273], Aleksandrov et al. [11] give alternative time bounds that depend on other parameters related to the "fatness" of the triangular facets of a weighted polyhedral surface. They place Steiner points along edges in a geometric progression, as Papadimitriou [317] has done for approximating shortest paths in three dimensions (Section 6.3). This allows one to compute a $(1+\epsilon)$-approximate shortest path from s to t in time $O(Mn \log Mn + nM^2)$ (and space

$O(nM^2)$), where $M = O(\frac{W/w}{\epsilon \sin\theta} \log \frac{\lambda W}{hw\epsilon})$, λ is the length of a longest edge, h is the minimum altitude of a triangular facet, θ is the smallest angle of any triangular facet, W is the maximum (resp., minimum) weight of a facet, and $0 < \epsilon < \frac{1}{3} + \frac{W}{2w}$. (See also Section 6.3, where the same method is mentioned in the unweighted case.) Note that, while the dependence on ϵ and on geometric precision parameters is substantially worse than in the algorithm of Mitchell and Papadimitriou [292], the worst-case dependence on n is much better. (If, as in [292], the coordinates have integral values at most N, then $\sin\theta = O(1/N^2)$ and $h = O(1/N)$, making the time bound roughly $O(\frac{N^4 W^2 n}{\epsilon^2})$.) An improved variant of their result ([12]) searches a reduced subgraph, allowing them to remove the additive term nM^2 in the complexity, resulting in time bound $O(Mn \log Mn)$ (roughly $O(\frac{N^2 Wn}{\epsilon})$).

Several other papers have also addressed practical and effective (possibly heuristic) methods for the WRP; see the work by Alexander and Rowe [13, 14, 15] and a recent pair of papers by Kindl, Shing, and Rowe [238, 239], which report practical experience with a simulated annealing approach to the WRP. Johansson [229] has implemented a version of the edge subdivision method (also investigated by [246, 273]) and studied its use in fluid flow computations for injection molding.

Papadakis and Perakis [315, 314] have generalized the WRP to the case of time-varying maps, where both the weights and the region boundaries may change over time; they obtain generalized local optimality conditions for this case and propose a search algorithm to find good paths.

4.4 Minimum-Time Paths: Kinodynamic Motion Planning

Our discussion so far has focussed on path planning problems with *holonomic* constraints — those that are completely specified in terms of the robot's configuration, which is described by a k-vector, if the robot has k degrees of freedom. In *non-holonomic* motion planning, the constraints on the robot are specified in terms of a non-integrable equation involving also the derivatives of the configuration parameters. For example, non-holonomic constraints may specify bounds on the robot's velocity, acceleration, or the curvature of its path. See Latombe [247] and Li and Canny [258] for more a more detailed discussion of non-holonomic constraints and motion planning.

The *kinodynamic motion planning problem* (also known as the *minimum-time path problem*) is a non-holonomic motion planning problem in which the objective is to compute a *trajectory* (a time-parameterized path, $(x(t), y(t))$) within a domain P that minimizes the total time necessary to move from an initial configuration (position and initial velocity) to a goal configuration (position and velocity), subject to bounds on the allowed acceleration and velocity along the path. The problem formulation is intended to model the fact that real mobile robots have a bounded acceleration vector and a maximum speed. In its general form, it is a difficult optimal control problem; optimal paths will be complicated curves given by solutions to differential equations.

The bounds on acceleration and velocity are most often given by upper bounds on the L_∞ norm (the "decoupled case") or the L_2 norm (the "coupled case").

Exact solutions to the kinodynamic motion planning problem are known in one dimension (O'Dúnlaing [306]) and in two dimensions (Canny, Rege, and Reif [81]). The algorithm of [81] is for the decoupled case (L_∞ bounds on velocity and acceleration); it requires exponential time and polynomial space. Their method is based on characterizing a set of "canonical solutions" (related to "bang-bang" controls) that are guaranteed to include an optimal solution path. This leads to an expression in the first-order theory of the reals, which can then be solved exactly in exponential time. It remains open, however, whether or not a polynomial-time algorithm exists in two dimensions. For three or more dimensions, the problem is at least NP-hard, as implied by the lower bounds of Canny and Reif [82].

Approximation methods have been developed by Donald et al. [140], who have given a polynomial-time algorithm that produces a trajectory requiring time at most $(1 + \epsilon)$ times optimal, for the decoupled case. Their approach is to discretize (uniformly) the four-dimensional phase space that represents position and velocity, with special care to ensure that the size of the grid is bounded by a polynomial in $1/\epsilon$ and n. They prove that shortest paths in the induced grid graph are guaranteed to be close to optimal. The running time of their algorithm has been improved by Donald and Xavier [139]. Approximation algorithms for the coupled case have been given independently by Donald and Xavier [139] and by Reif and Tate [341]. By using a *non*-uniform discretization of d-dimensional configuration space, Reif and Wang [338] have obtained an approximation algorithm with a time complexity that improves that of [139], reducing the dependency on ϵ from $O((1/\epsilon)^{6d-1})$ to $O((1/\epsilon)^{4d-2})$.

4.5 Curvature-Constrained Shortest Paths

Related to the kinodynamic motion planning problem is the problem of finding shortest paths subject to a bound on their *curvature*. The *curvature-constrained shortest-path problem* is to compute a shortest obstacle-avoiding smooth (C^1) path joining point s, with prescribed orientation, to point t, with prescribed orientation, such that for every subinterval of the path, the average curvature is at most 1. (The *average curvature* of a path $p : I \rightarrow \Re^d$ in the interval $[u_1, u_2] \subseteq I$ is defined to be $||p'(u_1) - p'(u_2)||/|u_1 - u_2|$, where the parameter u denotes arc length.) Placing a bound on the curvature can be thought of as a means of handling an upper bound on the acceleration vector of a point robot (e.g., an idealized aircraft) whose speed is constant, or can be thought of as the constraint imposed when modeling a car-like mobile robot having a minimum turning radius. The complexity of solving the general problem in a polygonal domain has been open until very recently; Reif and Wang [339] have shown that it is NP-hard in a polygonal domain having n vertices, each having coordinates specified by $n^{O(1)}$ bits.

Since the general problem is difficult to solve exactly, algorithms for restricted versions of the problem, as well as approximation algorithms, have been the topic of recent investigations.

Early investigations into the problem were by Dubins [144], who characterized shortest curvature constrained paths in the absence of obstacles: a shortest path consists of a sequence of at most three segments, each of which is a straight line segment ("S") or an arc of a unit radius circle ("C"), with the allowable sequences being CCC, CSC, or a subsequence of one of these two. Reeds and Shepp [334] extended this result, obtaining a characterization of shortest paths in the case in which the robot is allowed to move in reverse, as well as forward. Boissonnat, Cérézo, and Leblond [75] give an alternative method of obtaining characterizations in both cases, based on optimal control theory. (See also [368].)

Approximation algorithms for a shortest "ϵ-robust" path were given by Jacobs and Canny [227, 228]. (See also Barraquand and Latomb [55].) Here, "ϵ-robust" roughly means that small perturbations of certain points along the path do not cause the path to penetrate an obstacle. They place points that discretize the boundaries of the polygonal obstacles and connect these points by paths ("jumps") of standard shapes (circular arcs and straight segments); the resulting algorithm takes time $O((\frac{n^3}{\delta})\log n + \frac{n^2}{\delta^2})$, where δ is the spacing of the discretization points on the boundary; δ controls the robustness of the path as well as the degree of approximation. They also give an alternative quadtree-based algorithm, having complexity $O(n^4 \log n + (\frac{n}{\delta})^2)$. Wang and Agarwal [383] give time bounds that do not depend on the length parameter δ: they give (1) an $O((\frac{n}{\epsilon})^2 \log n)$-time algorithm that produces a feasible path (not necessarily ϵ-robust) that is at most $(1+\epsilon)$ times the length of a shortest ϵ-robust path; and (2) an $O((\frac{n}{\epsilon})^{2.5} \log n)$-time algorithm that produces a feasible path that is $(\epsilon/2)$-robust, with length at most $(1+\epsilon)$ times the length of a shortest ϵ-robust path.

For the special case in which the obstacles are "moderate" (have differentiable boundary curves, with radius of curvature at least 1), Agarwal, Raghavan, and Tamaki [8] give an algorithm requiring time $O(n^2 \log n)$ to compute exactly a shortest curvature-constrained path from a starting configuration (position-orientation pair) to a goal location (no orientation specified), and an algorithm requiring time $O(n^2 \log n + \frac{1}{\epsilon})$ for computing an approximate shortest path (having length at most ϵ greater than optimal) between two configurations. Boissonnat and Lazard [77] obtain exact algorithms between two configurations for moderate obstacles whose boundaries consist of unit-radius circular arcs and straight segments. If the boundary arcs (straight or curved) are each of length at least some constant, then their algorithm requires time $O(n^2 \log n)$; otherwise, the complexity is $O(n^4 \log n)$. (Their algorithm remains polynomial even if the obstacles are not pairwise disjoint.)

Sellen [358] uses a simple discretization of the unit square to search, in $O(\epsilon^{-3})$ time, for a path among a set of constant-complexity obstacles that is "ϵ-approximate" (which roughly means that it is within factor

$(1+\epsilon)$ of being shortest, while maintaining an ϵ-clearance from obstacles and obeying an approximate (up to ϵ) curvature constraint). He also provides a decision procedure to determine the existence of a curvature-constrained path, in time polynomial in the reciprocal of a parameter that measures the difference between the radius of curvature in the constraint and the supremum of all radii for which a constrained path exists.

For the special case of curvature-constrained paths inside a convex polygon having n vertices, Agarwal et al. [4] use a careful characterization of the structure of shortest paths to obtain an algorithm with running time $O(n \log^2 n)$. Their result may be an important first step towards the solution of the more general problem inside a simple polygon:

Open Problem 7 *How efficiently can one compute a curvature-constrained shortest path in a simple polygon?*

Boissonnat et al. [76] examine curvature-constrained motion in a convex polygon (with m vertices), having a single simple polygonal hole (with n vertices). They compute, in time $O(m+n)$, a cycle surrounding the hole having the minimum possible curvature.

Wilfong [387, 388] considers the case in which the robot is to follow a given *network* of lanes, specified by a set of m line segments in free space, among a set of obstacles (having a total of n vertices). The robot is allowed to turn from one segment to another along a circular arc, of radius $\geq r_{min}$, if the two lanes intersect and the robot does not collide with the obstacles. In Wilfong [387], a polynomial-time $(O(m^2(n^2+\log m)))$ algorithm is given for preprocessing, after which, in $O(m^2)$ time, one can report a path (if one exists) having a minimum number of turns. (See also Mirtich and Canny [277].) Wilfong [388] shows that the problem of finding a minimum-length curvature-constrained path on a set of lanes is NP-complete; however, he also gives a dynamic programming algorithm to compute a shortest path (in time $O(m^6 n^2)$) for a *given* (feasible) sequence of turns (e.g., to optimize, locally, the path produced by the algorithm in [387]).

Fortune and Wilfong [159] give an exponential-time algorithm for determining if a curvature-constrained path exists between two configurations, assuming the robot is not allowed to reverse; their algorithm solves this reachability question in time and space $2^{O(poly(n,m))}$, where n is the number of vertices in the polygonal obstacles, and m is the total number of bits required to specify the vertices. Sellen [357] shows that the existence of a curvature-constrained path can be decided in time that is polynomial in d_{min}^{-1} and W^{-1}, where d_{min} is the smallest distance between obstacle features and $W = |R - R_c|/R$ is the "relative width" of the problem, relating the maximal curvature, R^{-1}, with the *critical curvature*, R_c^{-1}, which is the infimum over the curvatures R^{-1} for which a curvature-constrained path (with constraint R^{-1}) exists. Sellen also shows how to approximate the critical curvature R_c^{-1} to within any relative error $\epsilon > 0$, and to produce a corresponding path; the algorithm is polynomial in n and R_c/ϵ.

If the robot following the path *is* allowed to reverse direction, then Laumond [248] has shown that it is always possible to obtain a curvature-constrained path from s to t if the s and t lie in the same open, path-connected component of free space. Further, when allowing reversals, Laumond et al. [249] give an algorithm that determines a path (if one exists), producing a path having a local optimality property. Desaulniers [137] shows that, in the presence of reversals, in may be that *no* shortest path exists, even when there is a feasible path.

Švestka and Overmars [369] also study problems of planning routes for car-like robots, using a "probabilistic learning paradigm."

All of the discussion so far has been for paths in a two-dimensional environment. For three-dimensional spaces, Sussmann [367] gives a characterization of curvature-constrained shortest paths. Polynomial-time approximation algorithms for three and higher dimensions are given by Reif and Wang [338], by applying their discretization techniques developed for the kinodynamic motion planning problem.

Another interesting open area of research on curvature-constrained optimal paths is to consider network optimization problems in the curvature-constrained model. For example, we may desire a traveling salesperson tour (cycle) of minimum length, subject to the curvature constraint (see Section 7.2):

Open Problem 8 *What is the complexity of the curvature-constrained TSP for points in the unit square? What is the best approximation algorithm that can be given for the problem?*

4.6 Optimal Motion of Non-Point Robots

So far, we have considered only the problem of optimally moving a *point* robot. If the robot is modeled as a circle, or as a nonrotating polygon, then many of the results carry over by simply applying the standard *configuration space* approach in motion planning: "shrink" the robot to a (reference) point, and "grow" the obstacles (using a Minkowski sum) so that the complement of the grown obstacles model the region of the plane for which there is no collision with an obstacle if the robot has its reference point placed there. Chew [105] has examined the specific case of a circular robot; Hershberger and Guibas [202] have considered more general convex robots, obtaining essentially quadratic-time algorithms for optimal paths under translation.

Optimal motion of *rotating* non-circular robots is a much harder problem. Even the simplest case of moving a (unit) line segment (a *ladder*) in the plane is highly nontrivial. One notion of "optimal" motion requires that we minimize the average distance traveled by a set of k fixed points, evenly distributed along the ladder. This "d_k-distance" in fact defines a metric (for $k \geq 2$). The special case of $k = 2$ is the well-known *Ulam's problem*, for which optimal motions have been fully characterized, in the absence of obstacles, by Icking et al. [217].

The case of $k = \infty$ is an especially interesting case, requiring that we compute a minimum *work* motion of a ladder; however, no results are known yet for this problem. (The work measures the integral (over $\lambda \in [0,1]$) of the path length, $L(\lambda)$, for each infinitesimal subsegment of length $d\lambda$.) O'Rourke [308] has studied a restricted case of the d_∞-optimal motion problem.

Open Problem 9 *Characterize the d_∞-optimal (minimum-work) motion for a ladder that is allowed to translate and rotate in the plane. What if it is restricted to move within a polygonal domain?*

While d_1 does not define a metric, several cases of d_1-motion, and its generalization of measuring the distance traveled by any fixed "focus" F on the ladder, have been studied. In particular, if F is restricted to move on the visibility graph of a polygonal environment, Papadimitriou and Silverberg [318] (see also Sharir [361]) have obtained polynomial-time algorithms. Without restrictions, minimizing the d_1-distance, for any F *not* at an endpoint of the ladder, is NP-hard, but there exists an approximation algorithm; see Asano, Kirkpatrick, and Yap [40].

Open Problem 10 *Does minimizing the d_1-distance of a ladder endpoint remain NP-hard? Also, is it NP-hard to obtain a d_2-optimal motion of a ladder in a polygonal domain?*

Chen and Ierardi [100] have studied a velocity-constrained version of the problem of moving a ladder, such that no point of the ladder is allowed to have its speed exceed a given bound, and the objective is to minimize the time required to move the ladder from one configuration to another. For the case of no obstacles, they give a complete characterization of the optimal motion and give an explicit construction. See also the related work of Reister and Pin [344], who study time-optimal motion of mobile robots having independently controlled wheels.

4.7 Multiple Criteria Optimal Paths

The standard shortest-path problem asks for paths that minimize some *one* objective (length) function. Frequently, however, an application requires us to find paths to minimize *two or more* objectives; the resulting problem is a *bicriteria* (or *multi-criteria*) shortest-path problem. A path is called *efficient* or *Pareto optimal* if no other path has a better value for one criterion without having a worse value for the other criterion.

For example, in mobile robotics applications, we may wish to find a path that simultaneously is short in (Euclidean) length and has few turns. Note that a minimum-link path may be far from optimal with respect to Euclidean length; similarly, a shortest Euclidean length path may have thousands of links, while there exists a path joining start and goal that has only 2 links.

Multi-criteria optimization problems tend to be difficult. Even the bicriteria path problem in a graph is NP-hard [164]: Does there exist a path from s to t whose length is less than L and whose weight is less than W? Pseudo-polynomial time algorithms are known, such as the algorithm of Hansen [191], who finds all Pareto-optimal paths in a graph, in time polynomial in the number of paths and n. Experimental studies suggest that the average number of Pareto-optimal paths remains very small in practice, although in theory this number may be exponential. Various heuristics have also been devised; e.g., see Handler and Zang [190] and Henig [197].

In geometric problems, various optimality criteria are of interest, including any pair from the following list: Euclidean (L_2) length, rectilinear (L_1) length, other L_p metrics, link distance, total turn, etc.

NP-hardness lower bounds are known for several versions, including: [30] (1) Find a path in a polygonal domain whose L_2 length is at most L, and whose "total turn" is at most T; (2) Find a path in a polygonal domain whose L_p length is at most λ_p and whose L_q length is at most λ_q ($p \neq q$); and (3) Given a subdivision of the plane into red and blue polygonal regions, find a path whose length within blue regions is at most B and whose length within red regions is at most R.

One problem of particular interest is to compute a Euclidean shortest path within a polygonal domain, constrained to have at most k links. No exact solution is currently known for this problem. Part of the difficulty is that a minimum-link path will not, in general, lie on the visibility graph (or any simple discrete graph). Furthermore, the computation of the turn points of such an optimal path appear to require the solution to high-degree polynomials.

Open Problem 11 *For a polygonal domain (with holes) what is the complexity of computing a shortest k-link path between two given points?*

For a given k ($k \geq d_L$, where d_L is the s-t link distance), one can compute a path in a *simple* polygon P whose length is guaranteed to be within a factor $(1+\epsilon)$ of the length of a shortest k-link path, for any tolerance $\epsilon > 0$. The algorithm runs in time $O(n^3k^3 \log(Nk/\epsilon^{1/k}))$, polynomial in n and k, and logarithmic in $1/\epsilon$ and the largest integer coordinate N of any vertex of P [294]. Within the same time bound, one can compute an ϵ-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.

Aside from the problem of computing a shortest k-link path, one may ask if there always *exists* an s-t path that is *simultaneously* close to Euclidean shortest and minimum-link? In a *simple* polygon, such a path always exists and can be computed efficiently (in time $O(n)$): There is an s-t path whose link length is within a factor of 2 of the link distance from s to t, while also having Euclidean length within a factor of $\sqrt{2}$ of the Euclidean shortest-path length [31]. A corresponding result is not possible for polygons with holes. However, in $O(kE_{VG}^2)$ time, one can compute a path in a polygonal domain having at most $2k$ links and length at most that of a shortest k-link path [294].

In a rectilinear polygonal domain, some of these bicriteria path problems become easier, since there is a path-preserving graph (grid). In particular, efficient algorithms are known for the bicriteria path problem that combines *rectilinear* link distance and L_1 length. Yang, Lee, and Wong [390] and Chen, Daescu, and Klenk [94] give efficient algorithms for computing a shortest k-link rectilinear path, a minimum-link shortest rectilinear path, or any combined objective that uses a monotonic function of rectilinear link length and L_1 length in a rectilinear polygonal domain. Single-source queries can be answered in time $O(\log n)$, after $O(n \log^{3/2} n)$ preprocessing time to construct a data structure of size $O(n \log n)$ [94]; two-point queries can be answered in time $O(\log^2 n)$, using $O(n^2 \log^2 n)$ preprocessing time and space [94]. (See also the survey article of Lee, Yang, and Wong [254] on the subject of rectilinear path problems.) A related problem is studied by de Berg et al. [132, 133], who give efficient algorithms in two or more dimensions for computing optimal paths among a set of axis-parallel (possibly crossing) line segment obstacles according to a "combined metric," defined to be a linear combination of rectilinear link distance and L_1 path length: In the plane, using $O(n^2)$ preprocessing time and $O(n \log n)$ space, a data structure for a fixed source point can be computed, so that path length queries to a goal point can be answered in time $O(\log n)$. (Note, however, that optimal paths in this metric are not equivalent to the Pareto-optimal solution paths.) It would be interesting to study the complexity of the problem in a more general setting:

Open Problem 12 *How efficiently can one compute a (general) polygonal path in a polygonal domain, under a combined metric cost function that takes into account Euclidean length, the number of turns, and possibly the amount of turning?*

4.8 Other Optimal Path Problems

We briefly mention some various other optimal path problems:

(1) In the *sailor's problem*, the goal is to compute a minimum-cost path, where the cost of motion is *direction-dependent*, and there is a cost L per turn (in a polygonal path). For $L = 0$, Sellen [356] gives an algorithm for computing optimal paths in a polygonal domain, in time $O(n^2)$ times a bit complexity term. Sellen also considers the case in which $L > 0$, obtaining a $(1+\epsilon)$-approximation algorithm that requires time polynomial in n and $1/\epsilon$. See also the study by Rowe [350] on anisotropic weighted regions.

(2) In the *maximum concealment path* problem, the goal is to determine a path within a polygonal domain P that minimizes the length during which the robot is exposed to a given set of v "enemy" observers. This problem is a special case of the weighted region problem, in which weights are 0 (for travel in concealed free space), 1 (for travel in exposed free space), or ∞ (for travel through obstacles). Gewali et al. [168] use visibility graph methods, based on the local optimality conditions, to obtain polynomial-time algorithms for this problem. In a simple polygon, their time bound is $O(v^2(v+n)^2)$; in a polygonal domain, the bound becomes $O(v^4n^4)$.

(3) In the *minimum total turn* problem, the goal is to compute a polygonal s-t path that minimizes the sum of the absolute values of the turn angles at its vertices. This problem is solved in polynomial time ($O(E_{VG} \log n)$ time, $O(E_{VG})$ space) by reducing it to a shortest path problem in an augmentation of a visibility graph [30]. (See also Section 7.4, on angular-metric traveling salesperson problems.)

(4) In the *fuel-consuming* problem, one is given a set of n point sites in the plane and the goal is to find a "cheap" polygonal path from one site to another, with the vertices of the path being restricted to the set of point sites. The cost of a path, though, is not measured in terms of its Euclidean length, but in terms of a more general cost function, $l(p,q)$, which assigns a nonnegative cost to a flight from p to q. Naturally, one can compute a minimum-cost path in time $O(n^2)$ simply by searching the complete graph for a shortest path. However, it turns out that more efficient algorithms that exploit geometry are possible, if we assume that $l(\cdot,\cdot)$ has some simple properties: its description is of size $O(1)$ and $l(p,q)$ can be evaluated in $O(1)$ time, and $l(p,q) < l(p,q')$ if and only if $d_2(p,q) < d_2(p,q')$ (where $d_2(\cdot,\cdot)$ denotes Euclidean distance). Efrat and Har-Peled [146] show that a cheapest route can be computed in time $O(n^{1.5+\epsilon})$, for any fixed $\epsilon > 0$). Further, they show that if the cost function grows with at least a quadratic rate as a function of Euclidean distance (i.e., $l(p,q) = (d_2(p,q))^2 \cdot f(d_2(p,q))$, where $f(\cdot)$ is a positive, nondecreasing function), then it suffices to search the Gabriel graph (a subgraph of the Delaunay triangulation) of the point sites; thus, cheapest routes can be found in time $O(n \log n)$ in this case.

(5) In the problem of *shortest paths in an arrangement*, one is given a set of n lines in the plane, and points s and t on the lines, and must compute a shortest s-t path that is contained within the union of the lines. Since the arrangements can be computed in time $O(n^2)$ (see the chapter on arrangements by Agarwal and Sharir [2]), and shortest paths in planar graphs can be computed in linear time ([198]), the problem is trivially solved in time $O(n^2)$. It is an intriguing open question if there exists a subquadratic-time algorithm. There has been partial progress towards addressing this question: Bose et al. [78] give a 2-approximation algorithm that requires $O(n \log n)$ time, and Eppstein and Hart [151] give an algorithm for computing an exact shortest path in time $O(n + k^2)$, where k is the number of different line orientations.

(6) In the *asteroid avoidance* problem, one is given a set of obstacles, each moving along a fixed (known) trajectory, and the problem is to find a minimum-time obstacle-avoiding path for a point robot that is subject to a velocity bound. This problem was first studied by Reif and Sharir [337], who show that the general problem is PSPACE-hard in three dimensions and that the two-dimensional problem can be solved in exponential time in the case of pure translational motion. Canny and Reif [82] prove that the two-dimensional problem is NP-hard, even for convex translating obstacles, moving with fixed velocity, that do not collide. (Effectively, the fact that the obstacles are moving lifts the dimension of the problem from two to three, making it substantially more difficult; see Section 6.) Canny [80] has given a PSPACE algorithm to solve the asteroid avoidance problem.

5 On-Line Algorithms and Navigation Without Maps

In all of the optimal path problems we have discussed so far, we have assumed that we know in advance the exact layout of the environment in which the robot moves; i.e., we assume we are given a perfect *map*. In many situations, the robot does not have prior information about the obstacles in the environment; e.g., the robot may be placed in a completely new environment, or it may roam on a factory floor or an office building where there are frequent changes in the positions of obstacles. In such cases, we may have perfect information about the robot's current location, as well as the location of the goal, but we acquire information about the environment *on-line*, as the robot encounters or senses obstacles.

Common assumptions about the sensory capabilities of the robot include (1) a *tactile robot*, in which the robot learns of the boundary of an obstacle only as it encounters it, and moves along it; or a (2) *vision-based robot*, in which the robot learns of obstacles only as it is able to see them. (It is common to assume that the robot has 360-degree vision, allowing it to look in all directions; however, this assumption may be relaxed as well.) For a vision-based robot, there are also different assumptions that can be made about the nature of the sensor: (a) it may be that it knows only about that portion of the obstacle boundaries that it has seen; or (b) it may be that it has recognition capabilities, so that as soon as it sees any part of the boundary of an obstacle, it is able to determine the shape, size, and position of the obstacle, thereby learning the entire obstacle boundary.

Our goal is to obtain a *navigation strategy* that controls the motion of the robot, while utilizing sensory input, in order to minimize some notion of length (e.g., Euclidean length) of the path of the robot, which is to get from a start point, s, to a goal (target) location, t (which may be a point, a line, a region, etc.). The environment is assumed to be a polygonal domain, P, that is unknown to us. Often, there is very special structure assumed about the obstacles that constitute the holes of P.

Some of the first work that obtained worst-case bounds on the length of a path produced by a navigation strategy was that of Lumelsky and Stepanov [268, 269]. They give navigation strategies for a tactile robot moving among a set of arbitrary obstacles. The robot is assumed to know, at any given time, its own position, the position of the goal, and whether or not it is in contact with an obstacle; it is assumed to have only a small constant-size memory for recording other information that is learned along the way. One simple strategy ("BUG1"), attempts to head towards the goal until an obstacle is encountered; then, the robot follows the boundary of the obstacle, all the way around the perimeter, keeping track of the point p that is closest to the goal; finally, the robot returns to point p (by following the boundary) and heads again towards the goal. This strategy finds a path whose length is at most $d_2(s,t) + \frac{3}{2}L$, where L is the sum of the perimeters of the obstacles that intersect a disk of radius $d_2(s,t)$ centered at t. Within their model, they also prove a lower bound, showing that no strategy can guarantee a path length better than $d_2(s,t) + L - \epsilon$, for any $\epsilon > 0$. A second strategy ("BUG2") attempts to stay on the straight segment $\overline{st}$, at the cost of possibly visiting obstacles more than once. BUG2 is shown to produce a path of length at most $d_2(s,t) + \sum_i \frac{n_i L_i}{2}$, where n_i is the number of times $\overline{st}$ crosses the ith obstacle, and L_i is the perimeter of the ith obstacle. For convex obstacles, BUG2 is essentially optimal in their model. See also [129] for some further work on an extension of the Lumelsky-Stepanov model. Other papers on maze traversal strategies include [74, 329], as well as the surveys of Lumelsky [265, 266, 267].

While the Lumelsky-Stepanov result gives a worst-case additive error bound on the robot's path length, it does not give a bound on the *ratio* between the robot's path length and the (true) shortest path length, $d(s,t;P)$, in P. In order to evaluate the effectiveness of a navigation strategy, σ, in an on-line setting, it is now common to use the notion of a *competitive ratio*, $\rho(n)$, where $n = d_2(s,t)$ here denotes the Euclidean distance between s and t, and the ratio $\rho(n)$ is defined by

$$\rho(n) = \sup_{P,s,t:d_2(s,t)=n} \frac{d_\sigma(s,t;P)}{d(s,t;P)},$$

where $d_\sigma(s,t;P)$ is the length of the s-t path produced by strategy σ in P, and we assume that a unit diameter circle can be inscribed in each obstacle. In other words, our goal is to minimize the ratio between the length of the path obtained using the strategy to the length of a shortest path (with perfect information); the competitive ratio $\rho(n)$ is the maximum value of this ratio, over all environments having a given start-to-goal distance n.

The competitive ratio, in this context, has been studied first by Papadimitriou and Yannakakis [319], and independently by Eades, Lin, and Wormald [145]. In particular, [319] show that if the obstacles are all axis-aligned squares, and the robot is equipped with a vision sensor, then one can achieve a competitive ratio of $\rho(n) = \frac{\sqrt{26}}{3} + o(1)$, for all n. (The bound is $\frac{5}{3}$ if s and t are points having the same x- or y-coordinate.) If the obstacles are in fact aligned *unit* squares, they prove that $\rho(n)$ is at least $3/2$, while supplying a strategy that achieves $\rho(n) = 3/2 + o(1)$, for all n. (It is now known that a ratio of $\rho(n) = 3/2$ is possible for square obstacles, even if they have different sizes and are not axis-aligned; see the citation of Chan and Lam [86] below.) Further, by an adversary argument, they show that, for arbitrary (e.g., "thin") aligned rectangular obstacles[2], there is no strategy with a bounded competitive ratio, for a robot with line-of-sight vision. In fact, in [145, 319], it is shown that if the goal region t is an infinite vertical line ("wall"), at distance n from s, and the obstacles are aligned rectangles, then $\rho(n) = \Omega(\sqrt{n})$. Blum, Raghavan, and Schieber [73] provide a "sweep algorithm" for this wall problem that shows a matching upper bound of $\rho(n) = O(\sqrt{n})$, both for a vision-based robot and for a tactile robot (utilizing a "doubling" search procedure, suggested by Baeza-Yates, Culberson, and Rawlins [48]).

If the obstacles are aligned rectangles having aspect ratio at most f and longest side at most g (and shortest side at least 1), then Mei and Igarashi [275] give an "adjusted bias heuristic" that achieves competitive ratio $1 + \frac{3}{5}f + o(1)$, if $f = o(\sqrt{n})$ and $fg = o(n)$, assuming s and t have a common x- or y-coordinate (the competitive ratio is slightly higher otherwise). (See also [276].)

Blum et al. [73] also study the "room problem", in which P consists of an n-by-n (aligned) square room, with aligned rectangular holes (obstacles). For the room problem, they give an algorithm achieving $\rho(n) = O(2^{\sqrt{3\log n}})$. Bar-Eli et al. [53] have improved upon this result, establishing a tight bound of $\rho(n) = \theta(\log n)$, for deterministic algorithms. The wall and room problems can be combined, resulting in a competitive ratio of $\rho(n) = O(\sqrt{n})$ for point-to-point navigation among aligned rectangular obstacles.

While, for deterministic algorithms, we have the tight bound $\rho(n) = \theta(\sqrt{n})$ for the competitive ratio in both the wall and point-to-point versions of the problem, it has been shown by Berman et al. [61] that randomized strategies are "powerful", in that one can obtain a competitive ratio of $O(n^{\frac{4}{9}}\log n)$ for the wall and point-to-point navigation problems among aligned rectangular obstacles. For randomized strategies, we define $\rho(n)$ to be the supremum of the ratio of the *expected* path length to $d(s,t;P)$, assuming that P is selected by an oblivious adversary, with knowledge of the strategy, but not of the coin tosses made during a walk using the strategy. Berman and Karpinski [62] obtained a randomized strategy for general convex obstacles with competitive ratio $O(n^{\frac{3}{4}})$.

Blum and Chalasani [70] have shown that if the robot is to make *multiple* trips from s to t, it can make effective use of information gained on each trip, allowing it to improve its performance as it learns more

[2]Note that if the obstacles (holes of P) are rectangles or squares, they are disjoint, but allowed to touch; however, the robot can "squeeze" between two touching obstacles. Thus, we cannot synthesize nonconvex obstacles by putting together rectangular obstacles. Also, unless otherwise stated (e.g., in the "room problem"), we are assuming that P is the infinite plane, with holes that are the obstacles; i.e., there is no outer boundary of P.

about the environment. In particular, they show a strategy in which, for every $i \leq n$, the ith trip of the robot is a path of length $O(\sqrt{n/i})$ times $d(s,t;P)$. Their results apply to the wall problem, as well as the point-to-point problem, in the presence of aligned rectangular obstacles. They also provide a lower bound, for deterministic strategies, of $\Omega(\sqrt{n/k})$ on the *cumulative* k-trip competitive ratio (which measures the ratio of the total length of all k trips, over k times $d(s,t;P)$).

If the obstacles are arbitrary *non*aligned rectangles, then the competitive ratio for the room problem goes up: Blum et al. [73] show that $\rho(n) = \Omega(\sqrt{n})$; they also give (non-tight) upper bounds that either assume an excluded range of orientations of the rectangles, or allow a randomized algorithm. If the nonaligned rectangles have aspect ratio at most r, then a strategy of Chan and Lam [86] obtains a competitive ratio of $(\frac{r}{2}+1)$, which is shown to be tight. In particular, in the case of nonaligned squares ($r=1$), Chan and Lam's result implies a competitive ratio of 3/2, improving the earlier bound of Papadimitriou and Yannakakis [319]. An asymptotic competitive ratio of 3/2 has been obtained by Bezdek [68] for the case of nonaligned cubes in three dimensions; this result in fact implies the two-dimensional result for squares.

For even more general environments P, Blum et al. provide some results in special cases of convex obstacles, as well as general polygonal domains ("mazes"), where the competitive ratio is $\theta(|V|)$, where V is the set of vertices of P. A simple "L-shaped" maze example shows that even a randomized algorithm cannot achieve a competitive ratio better than $(|V|-10)/6$. Blum et al. also consider the three-dimensional version of the wall problem, obtaining a lower bound of $\Omega(n^{2/3})$ for the competitive ratio, and matching upper bounds in special cases (obstacles that are generalized cylinders, in the wall problem, or aligned boxes, in the point-to-point problem). Berman et al. [61] show that randomization can, again, help, allowing a strategy with competitive ratio $O(n^{2/3-\epsilon})$ for the point-to-point and wall problems.

The on-line version of the weighted region problem (Section 4.3) has been studied by Reif and Wang [342], who consider an environment in which the axis-aligned rectangular "obstacles" are penetrable, with each having a weight (cost per unit distance) greater than one (the background has weight one). Using a modified sweeping strategy of Blum et al. [73], they show that a competitive ratio of $O(\sqrt{n})$ is achievable in the wall problem with penetrable obstacles (and this is tight). See their paper for generalizations to "recursive" weighted environments, in which penetrable obstacles may include other penetrable obstacles of higher weight.

In the *search* version of the on-line problem, our objective is to search for an entity at some *unknown* target location in an unknown environment, minimizing the total distance traveled from the starting point, until the visually identifiable target is first seen; see [48, 234, 242].

While for general simple polygons P, no constant competitive ratio is possible when searching for a target t, Klein [241] has shown that if one is navigating in a special type of simple polygon, called an *s-t street* (for s and t on the polygon boundary), then there is a strategy for searching for a path from s to t that achieves a competitive ratio of $1+\frac{3}{2}\pi \approx 5.71$. (Points s and t split the boundary of P into two subchains; P is an *s-t street* if each point on one subchain is visible from some point on the opposite subchain.) Here, both P and the coordinates of t are unknown to the robot; the robot is equipped with a vision sensor, and we assume that the goal t is visually identifiable. Streets (as well as "star-shaped polygons"; see below) enjoy the special property that in the tree of shortest paths from s, left-turning paths and right-turning paths are grouped. Klein's strategy is based on the idea of minimizing the "local absolute detour," while moving from one point known to be on the shortest *s-t* path to another such point. Klein's analysis was improved by Icking [215], who proved a bound of $\pi/2+\sqrt{1+\pi^2/4} \approx 4.44$ on the competitive ratio. Kleinberg [242] gives a simpler strategy and analysis, achieving a competitive ratio of $\sqrt{4+\sqrt{8}} \approx 2.61$, and shows further that it achieves an optimal ratio of $\sqrt{2}$ in the case of rectilinear streets. López-Ortiz and Schuierer [261] present a strategy, similar to Klein's, having a substantially simpler analysis, resulting in a ratio of $\pi+1 \approx 4.14$; they show that a hybrid strategy based on this one achieves a ratio of $\frac{1}{2}\sqrt{\pi^2+4\pi+8} \approx 2.76$. López-Ortiz and Schuierer [260] have given a further improved strategy, using ideas similar to Kleinberg's (but with a substantially more complex analysis), achieving competitive ratio $\sqrt{1+(1+\pi/4)^2} \approx 2.05$. López-Ortiz and Schuierer [263] have given an extension of the original approach of Klein, to "continuous local absolute detour," that results in a competitive ratio of 2.03; further, by combining this approach with their earlier method ([260]), López-Ortiz and Schuierer obtain a hybrid strategy achieving competitive ratio of 1.73. Most recently, Semrau [359] has developed a strategy that results in a competitive ratio of $\pi/2 \approx 1.57$, which is getting very close to the theoretical lower bound of $\sqrt{2}$.

Open Problem 13 *Is there a strategy achieving a competitive ratio of $\sqrt{2}$ in streets?*

The results on searching in streets discussed above have assumed that the robot does *not* know the location of the target t. For such problems, it is easy to show that $\sqrt{2}$ is a lower bound on the competitive ratio (see Klein [242]). However, López-Ortiz and Schuierer [260] have shown a $\sqrt{2}$ lower bound on the competitive ratio for deterministic strategies, even if the coordinates of the target *are* known to the robot and the street is rectilinear. Thus, for rectilinear streets, knowledge of the target location does not assist the robot.

López-Ortiz and Schuierer [264] give a strategy with a constant competitive ratio (12.72) that finds a path to a target point in an unknown star-shaped polygon[3], even if the coordinates of the target point are unknown, and it is not necessarily on the polygon's boundary. They also prove a lower bound of 9 on any strategy that must find a path in an unknown star-shaped polygon to a target point whose coordinates are not specified. Star-shaped polygons, like streets, enjoy the property that the left-turning and right-turning paths in the shortest path tree rooted at s are grouped. Note too that a star-shaped polygon can be made into an s-t street, for any vertex s, by adding, if necessary, a vertex t such that the diagonal st intersects the kernel.[4]

Some results are also known for more general polygons than streets or star-shaped polygons. Datta and Icking [128] introduced the notion of a "generalized street;" for rectilinear generalized streets they give an algorithm with competitive ratio of $\sqrt{82} \approx 9.06$ (or 9, in the L_1 metric) and prove a lower bound of 9 on the competitive ratio of any strategy, assuming the target t is *not* known to the robot. (A simple polygon P is a *generalized street* ($\mathcal{G}$-street) with respect to two points s and t on its boundary if for any point p on the boundary of P, there exists a horizontal chord, whose endpoints are on different subchains of the boundary, such that p is weakly visible to the chord. The class of generalized streets strictly contains the class of streets.) López-Ortiz and Schuierer [260] show a lower bound of 9 even in the case that the coordinates of the target are known. Datta, Hipke, and Schuierer [127] define even more generalized notions of rectilinear streets ("HV-streets" and "θ-$\mathcal{G}$-streets"), for which they prove bounds in the L_1 metric: 14.5 is an upper and lower bound on the competitive ratio for HV-streets, while 19.97 is an upper bound for θ-$\mathcal{G}$-streets. (See also Schuierer [355] for lower bounds on the competitive ratio in θ-$\mathcal{G}$-streets.) López-Ortiz and Schuierer [262] give a competitive strategy (with ratio 80) in arbitrarily oriented (nonrectilinear) $\mathcal{G}$-streets. Kleinberg [242] considers searching in general rectilinear simple polygons, obtaining a strategy with competitive ratio $O(m)$, where m is the number of *essential cuts*, which may be much smaller than the number of vertices.

Streets have also been studied with respect to searching in link distance, instead of L_2 or L_1 length. Ghosh and Saluja [174] give a deterministic strategy for searching for an s-t path in a street, using at most $2m-1$ links, where m is the link distance from s to t. Further, they show that this bound is best possible for deterministic strategies in general streets. For rectilinear streets, a rectilinear link distance of $m+1$ is achievable, and this is best possible; here, m is the rectilinear link distance from s to t. Ghosh and Saluja observe that in general (non-street) simple polygons, no competitive ratio better than $n/4$ is possible, where n is the number of vertices.

On-line path problems arise also for objectives other than that of finding a path from a start s to a target point t. Icking and Klein [216] have given a competitive strategy for the problem of searching for the kernel of an unknown star-shaped polygon; the goal is for the robot to get to *some* point of the kernel. (A vision-equipped robot can recognize when it reaches a point in the kernel.) The competitive ratio is based on the distance from s to the point of the kernel that is closest to s. Icking and Klein [216] obtain a ratio of 5.48 (which they have subsequently improved slightly); they also prove a lower bound of $\sqrt{2}$ (which was subsequently increased to 1.48 by [264]). The best current bound is that of Lee et al. [255], who obtain a

[3] A polygon is *star-shaped* if there exists a point within it from which all other points of the polygon are visible.

[4] The *kernel* of a polygon P is the locus of points within P from which every point of P is visible. If the kernel is non-empty, then the polygon is star-shaped.

competitive ratio of $1 + 2\sqrt{2} \approx 3.829$. López-Ortiz and Schuierer [264] give a constant competitive ratio (of 46.35) for the on-line *recognition* of a star-shaped polygon, in which the robot must execute a path until it can prove or disprove the star-shapedness of P. They also show a lower bound of 9 on the competitive ratio of any such strategy.

In the *explore* (or *mapping*) version of the problem, our objective is to execute a path such that the robot can map out the entire space, by seeing every point of free space; see [135, 242]. (See also Section 7.4 on the "watchman route problem," which is the *off-line* version of the problem, to compute a shortest route that sees the entire space when the map is *given*.) In particular, Deng, Kameda, and Papadimitriou [135] have shown that no competitive strategy exists, in general, if there are an unbounded number of obstacles; however, if the number of obstacles is bounded, they obtain a competitive strategy with a constant competitive ratio ($O(k)$, where k is the number of obstacles). In particular, if P is a simple rectilinear polygon, there is a 2-competitive deterministic algorithm [135, 136], and a 5/4-competitive randomized algorithm [242] for the exploration problem. For general simple polygons, the competitive ratio of [135] is proved to be constant, but is only estimated to be in the thousands. A bound of 133 was later given by Hoffman et al. [211], and has recently been improved to $(18\sqrt{2} + 1) \leq 26.5$ by the same set of authors [212].

Kalyanasundaram and Pruhs [231] study the search and explore problems for a vision-equipped robot among a set of k disjoint convex obstacles having average aspect ratio[5] $\overline{\alpha}$. They obtain tight bounds on the competitive ratio for both problems: $\theta(\min\{k, \sqrt{k\overline{\alpha}}\})$. (In the mapping problem of [231], the robot is required to see all of the *boundary* of the work space, but not necessarily all of its interior; this is in contrast with the mapping problem of [135].) They also show that the natural greedy "nearest neighbor" heuristic for the search problem can be quite bad, showing an $\Omega(2^k)$ lower bound on the competitive ratio for that strategy. In the *visual traveling salesperson problem* (visual TSP), the robot's objective is to visit and traverse the boundary of every obstacle; this formulation is meant to model the fact that a robot (equipped with a vision sensor) may have to get close to an object in order to map it completely. For the visual TSP, Kalyanasundaram and Pruhs [232] give a 19-competitive algorithm, based on applying their 18-competitive algorithm for the "on-line TSP" in planar graphs to a type of "relative neighborhood graph" in the presence of obstacles. (See also [233].)

For other related work, without theoretical guarantees on the competitive ratio, but useful in autonomous vehicle navigation, we refer the reader to the book edited by Iyengar and Elfes [225], as well as [226, 281, 293, 330, 331]. Mitchell [281] has considered a model, based on a special case of the weighted region problem (Section 4.3), in which the robot gathers information, which it accumulates in a map, and at each step applies the best possible *local strategy*, assuming travel within known free space has a cost-per-unit-distance of 1, while travel in unexplored terrain has cost $\alpha > 1$ per unit distance.

For a recent survey of on-line searching and navigation algorithms, see Berman [60].

6 Shortest Paths in Higher Dimensions

We turn our attention now to the problem of computing shortest paths in higher dimensional geometric spaces. Most of the discussion will focus on three-dimensional spaces, since most effort has been devoted to this case. We begin with a few definitions.

A *polyhedral domain* is a connected subset, P, of $\Re^3$ whose boundary consists of a union of a finite number of triangles. (The definition is readily extended to d dimensions, where the boundary must consist of a union of "simplices.") The complement of P consists of connected (polyhedral) components, which are the *obstacles*. A polyhedral domain is *orthohedral* if each boundary facet is orthogonal to one of the coordinate axes. A polyhedral domain P is a (convex) *polytope* if it is the convex hull of its vertices.

A *polyhedral surface* is a connected union of a finite number of polygonal faces, with any two polygons intersecting in a common edge, a common vertex, or not at all, and each edge belonging to exactly two polygons.

[5]Here, the aspect ratio of a convex body is the ratio of the radius of the smallest circumscribing circle to the radius of a largest inscribed circle.

Throughout this section, n will denote the number of edges in a polyhedral domain or surface. Without loss of generality, we can assume that all faces of a polyhedral surface are triangles, since a polygon triangulation algorithm can be applied to decompose each polygonal face into triangles, introducing a total of $O(n)$ additional edges and faces.

6.1 Complexity

In three or more dimensions, most shortest-path problems are very difficult. The problem is difficult even in the most basic Euclidean shortest-path problem in a three-dimensional polyhedral domain P, and even if the obstacles are convex, or the domain P is simply connected. There are two sources of complexity, as we now discuss.

One difficulty arises from algebraic considerations. In general, the structure of a shortest path in a polyhedral domain need not lie on any kind of discrete graph. Shortest paths in a polyhedral domain will be polygonal, with bend points that generally lie *interior* to obstacle edges, obeying a simple "unfolding" property: The path must enter and leave at the same angle to the edge. It follows that any locally optimal subpath joining two consecutive obstacle vertices can be "unfolded" at each edge along its edge sequence, thereby obtaining a straight segment. (The *edge sequence* of a path is the ordered list of obstacle edges that are intersected by it.) Given an edge sequence, this local optimality property uniquely identifies a shortest path through that edge sequence. However, to compare the lengths of two paths, each one shortest with respect to two (different) edge sequences, requires exponentially many bits, since the algebraic numbers that describe the optimal path lengths may have exponential degree [50, 51].

A second difficulty arises from combinatorial considerations. The number of combinatorially distinct (i.e., having distinct edge sequences) shortest paths between two points may be exponential. Canny and Reif [82] have used this fact to prove that the shortest-path problem is NP-hard, even if the obstacles are simply a set of parallel triangles. While this result is strong evidence that we will not be able to solve the problem exactly in polynomial time, it does not rule out the possibility that we could construct a shortest path map in time proportional to its combinatorial size, which may be exponential in general, but far smaller in many practical cases.

Open Problem 14 *Can one compute a shortest path map for a polyhedral domain in output-sensitive time?*

Sharir and Schorr [362] gave a doubly exponential time ($2^{2^{O(n)}}$) exact algorithm, based on reducing to an algebraic decision problem in the theory of real closed fields. This result was improved by Reif and Storer [340], who give a singly exponential time algorithm (requiring $2^{n^{O(1)}}$ time and $n^{O(\log n)}$ space), based on the same theory, but using a more efficient reduction. Finally, Canny [80] has given a PSPACE algorithm, which applies not only to the shortest path problem in three dimensions, but also to the two-dimensional asteroid avoidance problem (see Section 4.8).

Given the difficulty of solving the general problem exactly, it is natural to consider approximation algorithms for the general case, or to consider special cases in which we can obtain polynomial bounds.

6.2 Special Cases

If the polyhedral domain P has only a small number, k, of convex obstacles, a shortest path can be found in $n^{O(k)}$ time, as shown by Sharir [360]. If the obstacles are known to be "vertical buildings" having only k different heights, then shortest paths can be found in time $O(n^{6k-1})$ [169]; however, it is not known if this version of the problem is NP-hard if k is allowed to be large. Both of these special cases have worst-case exponential algorithms; is there some nontrivial case of disjoint obstacles in three dimensions that is *not* hard to solve exactly? We have noted that Canny and Reif's hardness proof applies even to simple (convex) triangular "plates" that lie in parallel planes; however, their construction seems to rely on some edges of the triangles *not* being axis-parallel. This suggests an interesting question:

Open Problem 15 *What is the complexity of the Euclidean shortest-path problem in 3-space for obstacles that are disjoint aligned boxes? What about for disjoint (unit) spheres?*

If we require paths to stay on a polyhedral surface (i.e., the domain P is essentially 2-dimensional), then the unfolding property of optimal paths can be exploited to yield polynomial-time algorithms. This was first used by Sharir and Schorr [362] to obtain an $O(n^3 \log n)$-time algorithm for convex surfaces. Mitchell, Mount, and Papadimitriou [291] obtained an $O(n^2 \log n)$-time algorithm for general polyhedral surfaces, by developing a continuous Dijkstra method of propagating a shortest path map over the surface, taking advantage of the local optimality (unfolding) property. Chen and Han [99] have improved the time bound even further, obtaining an algorithm requiring $O(n^2)$ time and $O(n)$ space. (The algorithm of [99] relies on the nonoverlapping property of the "star unfolding", as shown by Aronov and O'Rourke [34]; see below.) These algorithms not only construct a shortest path map with respect to a single source, but can be used to construct a geodesic Voronoi diagram for multiple source points within the same time bound (where n now includes the number of source points).

One of the most interesting open problems in this area is to break the quadratic time barrier, even for the case of convex polytopes:

Open Problem 16 *Can one compute shortest paths on the surface of a convex polytope in $\Re^3$ in subquadratic time? In $O(n \log n)$ time?*

Several facts are known about the set of edge sequences corresponding to shortest paths on the surface of a *convex* polytope P in $\Re^3$. In particular, Mount [300] has shown that the worst-case number of distinct edge sequences that correspond to a shortest path between some pair of points is $\Theta(n^4)$. Further, Agarwal et al [3] have shown that the exact set of such sequences can be computed in time $O(n^6 \beta(n) \log n)$, where $\beta(n) = o(\log^* n)$. (A simpler $O(n^6)$ algorithm can compute a small superset of the sequences [3].) The number of *maximal* edge sequences for shortest paths is $\Theta(n^3)$, as shown by Schevon and O'Rourke [352]. Some of these results depend on a careful study of the "star unfolding" with respect to a point p on the boundary, ∂P, of P. The *star unfolding* is the (nonoverlapping [34]) cell complex obtained by subtracting from ∂P the shortest paths from p to vertices of P, and then "flattening" the resulting boundary.

Agarwal et al [3] have also shown that two-point queries can be answered in time $O((\sqrt{n}/m^{1/4}) \log n)$, after spending $O(n^6 m^{1+\delta})$ preprocessing time and storage, for any choice of $1 \leq m \leq n^2$, and $\delta > 0$. (If one query point always lies on an edge of the polytope, the algorithm can be improved to use $O(n^5 m^{1+\delta})$ preprocessing time and storage and guarantee $O((n/m)^{1/3} \log n)$ query time, for any choice of $1 \leq m \leq n$.) Further, the geodesic diameter is obtained in time $O(n^8 \log n)$, improving an earlier $O(n^{14} \log n)$ bound of O'Rourke and Schevon [311]. Chiang and Mitchell [107] show how two-point queries can be answered efficiently (even in optimal $O(\log n)$ time) on *nonconvex* polyhedral surfaces; however, the preprocessing and space complexities are even higher than in the convex case. Performing efficient two-point queries while using only a small polynomial amount of storage remains an open problem:

Open Problem 17 *How efficiently, and using what size data structure, can one preprocess a polyhedral surface for exact two-point queries? Can exact two-point queries be done in sublinear query time using subquadratic storage? What if the surface is convex?*

In the special case of terrain surfaces (polyhedral surfaces having at most one intersection point with any line parallel to the z-axis), de Berg and van Kreveld [131] have studied various optimal path problems, including some bicriteria versions, with constraints imposed on the maximum allowed altitude. They build a "height-level map," in time $O(n \log n)$, stored implicitly using $O(n)$ space, which enables $O(\log n)$ time queries to compute a shortest s-t path that stays below a given elevation z, or an s-t path having a minimum total ascent.

6.3 Approximation Algorithms

Papadimitriou [317] was the first to study the general problem from the point of view of approximations. He gave a fully polynomial approximation scheme that produces a path guaranteed to be no longer than

$(1+\epsilon)$ times the length of a shortest path. His algorithm requires time $O(n^4(L+\log(n/\epsilon))^2/\epsilon^2)$, where L is the number of bits necessary to represent the value of an integer coordinate of a vertex of P. Clarkson [119] gives an alternative method, requiring roughly $O(n^2 \log^{O(1)} n/\epsilon^4)$ time (the exact expression includes also a precision parameter that depends on the geometry of P).

Choi, Sellen, and Yap [114, 113] have re-examined closely the analysis of Papadimitriou and have addressed some inconsistencies found in the original algorithm. To this end, it is important to distinguish between the *bit* framework and the *algebraic* framework of studying the complexity of the problem. Almost all shortest path algorithms (and most computational geometry algorithms) assume an algebraic model of computation, in which the time complexity is measured in terms of the number of algebraic operations performed on real numbers. It is assumed that these operations are performed *exactly.* In the bit framework, though, time complexity is measured in terms of the number of boolean operations on bits, assuming the input is encoded with binary strings. Given the nature of current computer hardware, it is likely that the bit framework more accurately models actual computation times.

Choi, Sellen, and Yap [114] give upper bounds on the bit complexity of the approximate shortest-path problem. They have also introduced the important notion of "precision-sensitivity" in algorithms, where the goal is to write the complexity in terms of an implicit parameter, δ, that measures the implicit precision of the input instance [113]. For example, in the shortest-path problem, they define $\delta = (d_2 - d^*)/d^*$ to be the relative difference between the length d^* of an optimal path, and the length, d_2, of the second-shortest, locally optimal path; i.e., $d_2 > d^*$ is the length of a shortest path that uses an edge sequence distinct from any optimal edge sequence, but is closest in length to d^* among all such locally optimal paths. Provided that the optimal edge sequence is in some sense nondegenerate, one obtains an approximation algorithm that is polynomial in $1/\delta$ and the other parameters of the input, with only linear dependence on $1/\epsilon$.

Recently, Har-Peled [194] has shown how to compute an *approximate shortest path map* in polyhedral domains. In particular, he shows that, for a given source point s, and real parameter $0 < \epsilon \leq 1$, a subdivision of $\Re^3$ of size $O(n^2/\epsilon^{4+\delta})$ can be constructed in time roughly $O(n^4/\epsilon^6)$, so that for any point $t \in \Re^3$ a $(1+\epsilon)$-approximation of the length of a shortest s-t path can be reported in time $O(\log(n/\epsilon))$. His technique is to sprinkle a carefully selected set S of discrete points within P and to record with each point of S a "weight" that corresponds to the approximate shortest path distance from s to it; the approximate shortest path map is then given by the additive-weight Voronoi diagram of S.

In addition to approximation results for shortest paths in polyhedral domains, there have been a number of results on approximating shortest paths on polyhedral *surfaces.*

Hershberger and Suri [208] obtain a 2-approximation for a shortest s-t path on a convex polytope in time $O(n)$, using a relatively simple algorithm that considers the shortest path on the surface of the polytope's bounding box, between an appropriate pair of points. An extension to the algorithm allows one to compute a $2.38(1+\epsilon)$-approximate shortest path tree, SPT(s), in $O(n \log n)$ time. The method also results in a $2k$-approximation algorithm for shortest paths in a polyhedral domain consisting of k convex polytopes.

Agarwal et al. [6] extend the method of [208] by surrounding the input (convex) polytope with a tighter-fitting constant-size (depending on ϵ) bounding polytope, which approximately preserves shortest path distances. The result is that in time $O(n \log(1/\epsilon) + 1/\epsilon^3)$ one can compute a $(1+\epsilon)$-approximate shortest s-t path, for any $0 < \epsilon \leq 1$. (The approximate length of a shortest path can be reported in time $O(n + 1/\epsilon^3)$.) Har-Peled [193, 192] improves this result, obtaining results for the approximate two-point query version: He gives an $O(n)$-time algorithm to preprocess a convex polytope so that a two-point query can be answered in time $O((\log n)/\epsilon^{3/2} + 1/\epsilon^3)$, yielding the $(1+\epsilon)$-approximate shortest path distance, as well as a path having $O(1/\epsilon^{3/2})$ segments that avoids the interior of the input polytope. He also gives an $O(n+1/\epsilon^6)$-time algorithm to compute an approximate diameter of the polytope's surface, obtaining a pair of points on the surface whose shortest path distance is least $(1-\epsilon)$ times the diameter.

Varadarajan and Agarwal [382] have considered the problem of approximating shortest paths on general (nonconvex) polyhedral surfaces. They have obtained the first subquadratic-time algorithms for provably good approximating paths, computing a 13-approximation in $O(n^{5/3} \log^{5/3} n)$ time, or a 15-approximation in $O(n^{8/5} \log^{8/5} n)$ time. Their method is based on a partitioning of the surface into $O(n/r)$ patches, each having at most r faces, using a planar separator theorem. (The parameter r is chosen to be $n^{1/3}\log^{1/3} n$ or $n^{2/5}\log^{2/5} n$.) Then, on the boundary of each patch, a carefully selected set of points ("portals") is selected, and these are interconnected with a graph that approximates shortest paths within each patch. Finally, Dijkstra's algorithm is used to search for a shortest path in the resulting graph, which is proven to contain an approximately shortest path.

Open Problem 18 *Can one compute a $(1+\epsilon)$-approximate shortest path on a nonconvex polyhedral surface (or even on a terrain) in subquadratic time? Can one compute an $O(1)$-approximate shortest path in close to linear time?*

Har-Peled [194] has shown how to compute an approximate shortest path map on polyhedral surfaces, using techniques mentioned above. Given a source point and a parameter $0 < \epsilon \leq 1$, he constructs a subdivision of the surface of size $O((n/\epsilon)\log(1/\epsilon))$, so that a $(1+\epsilon)$-approximate shortest path query to any point t can be answered in time $O(\log(n/\epsilon))$, by locating t within the subdivision. The preprocessing time is $O(n^2 \log n + (n/\epsilon)\log(1/\epsilon)\log(n/\epsilon))$ for general surfaces, and $O((n/\epsilon^3)\log(1/\epsilon) + (n/\epsilon^{3/2})\log(1/\epsilon)\log n)$ for convex polytopes.

Finally, we mention some investigations on practical methods for computing nearly shortest paths on surfaces. By using the same methods that have been applied to the weighted region problem (Section 4.3) in subdivisions, Lanthier, Maheshwari, and Sack [246] and Mata and Mitchell [273] have shown that very simple algorithms based on searching a discrete graph (an "edge subdivision graph", or a "pathnet") produce paths that are remarkably close to optimal, and approach optimal as a parameter (δ, or $1/k$) approaches zero. The discrete graph can be constructed in advance, to assist in speeding two-point queries. Further, the path obtained can be postprocessed with a local optimality procedure that pulls the path "taut" within the sleeve of facets that it crosses, resulting in a solution even closer to optimal. Using a slightly different discrete graph than the edge subdivision graph of [246, 273], Aleksandrov et al. [11] give alternative time bounds that depend on other parameters related to the "fatness" of the triangular facets of a polyhedral surface. They place Steiner points along edges in a geometric progression, as in Papadimitriou [317]. This allows one to compute a $(1+\epsilon)$-approximate shortest path from s to t in time $O(Mn\log Mn + nM^2)$ (and space $O(nM^2)$), where $M = O(\frac{1}{\epsilon \sin\theta}\log\frac{\lambda}{h\epsilon})$, λ is the length of a longest edge, h is the minimum altitude of a triangular facet, θ is the smallest angle of any triangular facet, and $0 < \epsilon < \frac{2}{3}$. By searching a sparser subgraph, they have recently ([12]) improved the time bound to $O(Mn\log Mn)$.

6.4 Other Metrics

Link distance in a polyhedral domain in $\Re^d$ can be approximated (within factor 2) in polynomial time, by searching a weak visibility graph whose nodes correspond to simplices in a simplicial decomposition of the domain. The complexity of computing the exact link distance is open.

Open Problem 19 *How efficiently can link distance be computed in polyhedral domains in 3-space?*

For the case of orthohedral domains, and rectilinear (L_1) shortest paths, the shortest-path problem in $\Re^d$ becomes relatively easy to solve in polynomial time, since the "grid graph" induced by the facets of the domain serves as a path preserving graph that we can search for an optimal path. In $\Re^3$, we can do better than to use the $O(n^3)$ grid graph induced by $O(n)$ facets, as shown by Clarkson, Kapoor, and Vaidya [121]; an $O(n^2\log^2 n)$ size subgraph suffices for the case of n (possibly overlapping) axis-parallel boxes, allowing shortest paths to be found using Dijkstra's algorithm in time $O(n^2\log^3 n)$. More generally, for a set of obstacles given by n axis-aligned (not necessarily disjoint) boxes in $\Re^d$, de Berg et al. [132, 133] show that one can compute a data structure of size $O((n\log n)^{d-1})$, in $O(n^d \log n)$ preprocessing time, that supports fixed-source link distance queries in $O(\log^{d-1} n)$ time. Further, this result applies, within the same complexities, to the case of a *combined metric*, in which path cost is measured as a linear combination of L_1 length and the rectilinear link distance (see also Section 4.7).

In the case of axis-parallel *disjoint* box obstacles in $\Re^3$, Choi and Yap [115] have shown that rectilinear shortest paths can be computed in time $O(n^2 \log n)$. Also, for this same problem in higher dimensions, a recent structural result of Choi and Yap [117, 116] may help in devising very efficient algorithms: There always exists a coordinate direction such that *every* shortest path from s to t is monotone in this direction.

7 Other Network Optimization Problems

Until now, we have been considering problems of computing a shortest path from one point to another (or from one point to all others). We consider now some other network optimization problems, in which the objective is to compute a shortest path, cycle, tree, or other graph, subject to various types of constraints.

We focus primarily on two classes of problems: those of finding minimum-cost *trees* or *tours* that span some or all elements of a set S. We discuss the resulting "minimum spanning tree" and "traveling salesperson" problems in the next subsections, and then give more details of a general method of obtaining approximations to these problems. The subject of spanning trees and spanners is surveyed extensively in the chapter by Eppstein [150] in this handbook.

Other well-studied network optimization problems that we do not attempt to survey here include *minimum cost matching* (which has polynomial-time exact and approximate solutions; see [36, 379, 380, 389]) and *minimum weight triangulation (MWT)* (whose complexity status is still open, although constant-factor approximation algorithms exist for both the Steiner and non-Steiner versions; see Bern and Eppstein [64] and Levcopoulos and Krznaric [257]). We also refer the reader to the article of Smith and Winter [365], which surveys a large class of topological network design problems. Kalyanasundaram and Pruhs [233] survey *on-line* versions of standard network optimization problems.

7.1 Optimal Spanning Trees

Minimum Spanning Trees

A *minimum spanning tree (MST)* of a set of n points S is a tree of minimum total length whose nodes are the set S of n points, and whose edges are line segments joining pairs of points.

The (Euclidean) minimum spanning tree problem can be solved to optimality in the plane in time $O(n \log n)$, by appealing to the fact that the MST is a subgraph of the ($O(n)$-size) Delaunay diagram; after computation of the Delaunay diagram, results of Cheriton and Tarjan [104] can be applied to find the MST in only $O(n)$ additional time.

Proposition 4 *An edge in a Euclidean MST is Delaunay.*

The above proposition remains valid in $\Re^d$, for $d \geq 3$; however, the result does not lead directly to a subquadratic-time algorithm for MST in higher dimensions, since there can be $\Omega(n^2)$ Delaunay edges, even in $\Re^3$. However, geometry can be exploited to avoid examining the full set of $\binom{n}{2} = \Omega(n^2)$ weighted edges in the complete graph. Yao [391] was the first to compute an MST in $\Re^d$ in subquadratic time. His general method yields a time bound of $O(n^{2-\alpha_d}(\log n)^{1-\alpha_d})$, where α_d is a constant depending on the dimension d. His algorithm is based on partitioning the space around each point p into sufficiently small cones so that there is at most one MST edge incident on p per cone, with this edge linking p to its nearest neighbor within that cone. In Yao [391], $\alpha_d = 2^{-(d+1)}$, but this has improved as better data structures for nearest-neighbors have been developed. Agarwal et al. [5] give a randomized algorithm whose expected running time has $\alpha_d = \frac{2}{\lceil d/2 \rceil + 1} - \gamma$, for any fixed $\gamma > 0$. In three dimensions, their algorithm requires $O(n^{4/3} \log^{4/3} n)$ expected time. (See also Agarwal, Matoušek, and Suri [7], who study maximum spanning trees (Section 7.1); a variant of their somewhat simpler randomized algorithm applies also to minimum spanning trees.) These algorithms exploit the close relationship between the problem of computing an MST and that of computing a bichromatic closest pair of points between n red points and m blue points. Letting $T_d(n, m)$ denote the complexity of solving the bichromatic closest pair problem, Agarwal et al. [5] show that the Euclidean MST can be computed in time $O(T_d(n,n)\log^d n)$ (if $T_d(n,n) = o(n^{1+\epsilon})$), or in time $O(T_d(n,n))$, if $T_d(n,n)$ is superlinear. Since they give a randomized algorithm for the bichromatic closest pair, with expected time $O((nm)^{1-1/(\lceil d/2\rceil+1)+\epsilon})$, their result implies that the MST can be computed in expected time $O(n^{2-2/(\lceil d/2\rceil+1)+\epsilon})$. Callahan and Kosaraju [79] show a bound of $O(T_d(n,n)\log n)$, while Krznaric, Levcopoulos, and Nilsson [244], as well as Kapoor [235], obtain $O(T_d(n,n))$. These bounds hold for any L_p metric ($p \geq 1$); $O(T_d(n,n))$ is optimal in the algebraic computation tree model.

For some L_p metrics, more efficient algorithms are known. Agarwal et al. [5] give a deterministic algorithm requiring $O(n\log^d n)$ time for any metric having a polyhedral unit ball (e.g., L_1 and L_∞); see also Gabow, Bentley, and Tarjan [161]. In three dimensions, there is now an optimal $O(n\log n)$ time algorithm for the MST in the L_1 or L_∞ metric, due to Krznaric, Levcopoulos, and Nilsson [244] (improving an earlier $O(n\log n\log\log n)$ bound of [161]).

Clarkson [120] and Vaidya [378] have given algorithms that are particularly efficient for points that are independently and uniformly distributed in a unit cube in $\Re^d$. Their algorithms have expected time $O(n\alpha(cn,n))$, where c is a constant depending on dimension, and α is the very slowly growing inverse Ackermann function.

Several results are also known about approximation algorithms for the MST. Clarkson [118] gives an $O(n(\log n+\frac{1}{\epsilon}\log\delta))$ time ($O(n\log\delta)$ space) algorithm for a $(1+\epsilon)$-approximate Euclidean MST in $\Re^3$; he also gives results in higher dimensions for the L_1 metric ($O(n(\alpha(m,n)+\log^{d-1}(\frac{1}{\epsilon})\log\delta))$ time, $O(n(\log(\frac{1}{\epsilon})+\log\delta))$ space). Here, $m = O(n)$, and δ is a parameter that depends on the data: it is the ratio between the maximum and the minimum distance between pairs of points. Vaidya [377] gives a $(1+\epsilon)$-approximation for any L_p metric, requiring time $O(n(\log n)^{d+1}\epsilon^{-(d-1)})$, which he later improves to $O(\epsilon^{-d}n\log n)$ time [378]. Callahan and Kosaraju give an approximation, based on their "well-separated pair decomposition," for the Euclidean MST that requires time $O(n\log n + (\epsilon^{-d/2}\log\frac{1}{\epsilon})n)$.

Das, Kapoor, and Smid [126] have studied the problem of r-approximating the Euclidean MST, for *large* values of r: For $4 < r < n$, and for any $d \geq 1$, they prove a lower bound of $\Omega(n\log(n/r))$, in the algebraic tree model of computation, and prove that this is tight by exhibiting an algorithm having the same asymptotic time complexity. If the (non-algebraic) floor function and random access operations are permitted, then they obtain a $3\sqrt{d}n^{1-1/d}$-approximation algorithm requiring $O(n)$ time. For this more powerful model, Bern et al. [67] compute a $(1+\epsilon)$-approximate MST in the plane in time $O((1/\epsilon)n\log\log n)$ (time $O(n+\frac{n\log(1/\epsilon)}{\log n})$ in $\Re^1$).

The best lower bound for the (exact) MST problem is currently $\Omega(n\log n)$, in any fixed dimension $d \geq 1$, in the algebraic tree model of computation for a general input of unordered points. In contrast, the seemingly related *all-nearest-neighbors* problem can be solved in time $O(2^d n\log n)$, using the algorithm of Vaidya [381]. The all-nearest-neighbors problem is to compute the nearest neighbor for each of the n input points; given the MST, it is readily solved in $O(n)$ time, since the MST must include an edge linking each point to one of its nearest neighbors.

Open Problem 20 *Does there exist a near-linear time algorithm for Euclidean MST (or bichromatic nearest neighbors) in $\Re^d$, for $d \geq 3$?*

Maximum Spanning Trees

If instead of finding a *minimum* spanning tree, the objective is changed to ask for a *maximum*-length spanning tree on a set of points, the problem changes its nature. (Applications are given in [39].) While in graphs the same algorithms that find minimum spanning trees also can be used for computing maximum spanning trees, by negating edge lengths, the geometric version of the problem changes because it is not obvious how to find a small subgraph of the complete graph that is guaranteed to contain the maximum spanning tree. The natural generalization of the MST result might be to expect that the maximum spanning tree must appear as a subgraph of the (linear-size) *furthest-point* Delaunay diagram (see the chapter on Voronoi diagrams, by Aurenhammer and Klein [45]). However, this is not true in general, since the only points that are vertices of the furthest-point Delaunay diagram are the points on the convex hull; further, even if the input point set *is* in convex position, the maximum spanning tree need not lie on the furthest-point Delaunay diagram (see [296]). A different approach is taken by Monma et al. [299], who provide an optimal $O(n \log n)$-time algorithm for computing a maximum spanning tree of n points in the plane. They start with computing, in $O(n \log n)$ time, the *furthest neighbor* graph, joining each point to its furthest neighbor; the resulting graph is a forest, whose connected components are called *clusters*. They then show that the clusters can be cyclically ordered around their convex hull, allowing a maximum spanning tree to be computed by adding a longest edge between adjacent clusters. (If the input points are already in convex position, the algorithm requires only $O(n)$ time.) Subquadratic-time algorithms for higher dimensions are also known, based on efficient methods to compute bichromatic farthest neighbors. [7] give randomized algorithms with expected time $O(n^{4/3} \log^{7/3} n)$ in $\Re^3$, and $O(n^{2-\alpha_d})$ in $\Re^d$ $(d \geq 4)$, where $\alpha_d = \frac{2}{\lceil d/2 \rceil + 1 + \gamma}$, for any fixed $\gamma > 0$. They also give a simpler (deterministic) approximation algorithm, giving a tree at least $(1-\epsilon)$ times optimal, that requires $O(\epsilon^{(1-d)/2} n \log^2 n)$ time.

Minimum Steiner Spanning Trees

A *minimum Steiner spanning tree* (or simply *Steiner tree*) of S is a tree of minimum total length whose nodes are a *superset* of the given set S. Those nodes that are not points of S are generally called *Steiner points*. It turns out that allowing the flexibility of adding Steiner points in order to obtain a potentially shorter spanning tree makes the problem much more difficult. In fact, the Steiner tree problem is known to be NP-hard [163], even for points in the Euclidean plane.

The Steiner tree problem is in sharp contrast with the MST problem, which can be solved exactly in low-degree polynomial time. It is natural, therefore, to study how closely the MST solution approximates the Steiner tree. The supremum, over all point sets, of the ratio between the length of the MST and the length of the Steiner tree is known as the *Steiner ratio*[6]; it has been studied extensively in the last several years. A simple example (the three corners of an equilateral triangle) shows that the Euclidean Steiner ratio in the plane can be as high as $2/\sqrt{3}$. Gilbert and Pollak [175] conjectured that this ratio can in fact never be greater than $2/\sqrt{3}$. This conjecture was finally confirmed by a proof due to Du and Hwang [142]. (For the L_1 metric, the Steiner ratio in the plane is 3/2, and this is tight [214].)

Approximation algorithms have also been obtained for the Steiner tree problem. First, because of the Steiner ratio, the MST algorithms already give a $2/\sqrt{3}$-approximation for the Euclidean Steiner tree problem in the plane. However, in a series of results, starting with important work by Zelikovsky [392], improved approximation algorithms were obtained, for both graph versions and geometric versions of the problem. In the Euclidean plane, the approximation factor has been improved to just over 1.1 by Zelikovsky's "relative greedy" algorithm [393]. We refer the reader Bern and Eppstein [64] and Du and Hwang [143], for excellent surveys on these problems and the recent results. Finally, though, a PTAS was discovered by Arora [35] and Mitchell [289]. This result serves to separate the geometric versions of the problem from the "metric" version (in an arbitrary graph whose edge lengths satisfy the triangle inequality), since the metric version is known to be MAXSNP-hard (meaning that no PTAS exists, unless P=NP), even if all edge lengths are 1 or 2 [65].

A problem that arises in some applications in VLSI is that of computing a minimum-length rectilinear Steiner tree within a rectilinear polygon P, for a set of n sites on the boundary of the polygon. If P is rectilinear convex (any horizontal/vertical line intersects it in a connected set), having k vertices, Richards and Salowe [345] solve this problem exactly in time $O(k^4 n)$. For the same problem, Cheng, Lim, and Wu [102] give an $O(n^3)$ algorithm, and Cheng and Tang [103] give an $O(k^2 n)$ algorithm. If P is a general rectilinear polygon, then an exact solution requiring time $O(k^3 n)$ is given by Cheng [101], using a dynamic programming algorithm.

In the *on-line* version of the Steiner tree problem, the n points S appear one at a time, and the on-line algorithm must decide how to connect each successive point to the previously constructed Steiner tree. As in the case of on-line navigation problems, our interest is in establishing bounds on the *competitive ratio*,

[6]Many authors have defined the Steiner ratio to be the reciprocal of what we call the Steiner ratio. We follow the notation of Bern and Eppstein [64].

which is the supremum over all n-point sets S of the ratio between the weight of the connected graph constructed by the on-line algorithm and the weight of the minimum Steiner tree for S. As shown by Imase and Waxman [223], a natural greedy strategy results in an $O(\log n)$ competitive ratio, for any metric space: at step i, simply join the ith point v_i to the connected graph T_{i-1} built so far, by linking v_i to the point of T_{i-1} that is closest to it. In general metric spaces, Imase and Waxman also establish a lower bound of $\Omega(\log n)$ on the competitive ratio. However, their construction does not apply to Euclidean instances. For sets of points in the Euclidean plane (or even on a grid), Alon and Azar [16] are able to prove a lower bound of $\Omega(\log n/\log\log n)$ on the competitive ratio for the on-line Steiner tree problem, even for randomized on-line algorithms. (See also [233] for a survey of on-line network optimization problems.)

Group Steiner Tree Problem

In the *group Steiner tree problem* (also known as the "class Steiner problem," the "tree cover problem," or the "one-of-a-set Steiner problem"), we are given an undirected graph with edge weights, and a set of k subsets ("groups") of the graph's n vertices. The objective is to find a minimum-weight tree having at least one vertex from each group. Because of a reduction from set cover, it is NP-hard to approximate the group Steiner tree to a factor of $o(\log k)$; see [219, 240, 363, 270]. A $(k-1)$-approximation algorithm is given by Reich and Widmayer [335] and Ihler [218]. (See also Ihler, Reich, and Widmayer [221].) Slavík [363] gives an $O(\log k)$-approximation algorithm for the special case of an edge-weighted tree. Bateman et al. [58] give the first sublinear approximation factor for general graphs, with an approximation factor of $(1+\ln\frac{k}{2})\cdot\sqrt{k}$. Charikar et al. [87] give a k^ϵ-approximation algorithm that runs in polynomial time, as well as an $O(\log^2 n)$-approximation algorithm that runs in quasi-polynomial time. Garg, Konjevod, and Ravi [167] give a randomized $O(\log^3 n \log k)$-approximation algorithm for general graphs, which improves to $O(\log^2 n \log k)$ for a class of graphs that includes planar graphs, as well as graphs induced by a set of points in the Euclidean plane. Most recently, Charikar et al. [88] have derandomized the rounding scheme of [167], and obtained a deterministic $O(\log^2 n \log k \log\log n)$-approximation algorithm for general edge-weighted graphs.

Arkin et al. [22] obtained an $O(K \log k)$-approximation algorithm, where K is the maximum number of elements in a group; their algorithm is based on repeated applications of approximations to the k-MST problem (Section 7.1). Slavík [363] provides a $2K$-approximation algorithm, using an algorithm based on linear programming relaxations. (His approximation factor becomes $3K/2$ for the one-of-a-set TSP problem (Section 7.4), matching the Christofides factor of 3/2 when each group has size one; see Section 7.2.)

An outstanding open question is to determine if a constant-factor approximation algorithm exists for geometric instances of the problem; the hardness result for obtaining an $o(\log k)$-approximation does not apply to point sites in the plane.

Open Problem 21 *Is there an $O(1)$-approximation algorithm for the group Steiner problem on a set of points in the Euclidean plane?*

If the groups of points are in fact connected sets (e.g., polygons) in the plane, then an $O(\log k)$-approximation algorithm is given by Mata and Mitchell [272]. The special case in which the groups are intervals that lie on two parallel lines has a polynomial-time algorithm by Ihler [220].

The group Steiner problem is closely related to the one-of-a-set traveling salesperson problem (TSP), and the TSP with neighborhoods, which are discussed below in Section 7.4.

k-Minimum Spanning Trees

A *k-minimum spanning tree (k-MST)* is a minimum-length tree that spans some subset of $k \le n$ points of S. The fact that the particular subset of k points is not specified, but must be selected by the algorithm, makes the problem much more difficult than the usual MST problem (the case $k=n$). In fact, the problem is NP-hard, even for points in the Euclidean plane; see [158, 333]. A series of approximation results have been obtained for this problem. Ravi et al. [333] give an approximation algorithm with ratio $O(k^{1/4})$, which was improved to a factor of $O(\log k)$ by Garg and Hochbaum [166] and Mata and Mitchell [272]. Eppstein [149] has improved the approximation ratio to $O(\log k/\log\log n)$, and has given general techniques to improve the running times (as a function of n) of existing algorithms; further, he shows that the exact k-MST problem can be solved in time $2^{O(k\log k)}n+O(n\log n)$, which is simply $O(n\log n)$ for fixed k. Blum et al. [72] obtained the first $O(1)$-approximation; this was greatly simplified with the $2\sqrt{2}$-approximation of Mitchell [285, 290]. Ultimately, a PTAS was given by Arora [35] and Mitchell [289]. More details will be given below.

In general graphs having nonnegative edge weights, the current best approximation algorithm is a 3-approximation by Garg [165], which applies also to the "rooted" case (in which the tree is required to include a given node); this has been improved to a 2.5-approximation, by Arya and Ramesh [37], if the tree is not "rooted."

7.2 Traveling Salesperson Problem

In the *traveling salesperson problem (TSP)*, we are given a set S of n points ("sites") and are asked to find a shortest cycle ("tour") that visits every point of S. (There is a variant of the problem in which one wants a shortest *path* that visits S.) The TSP is a classical problem in combinatorial optimization, and has been studied extensively in many forms, including geometric instances; see [59, 250, 343, 230]. The problem is NP-hard, as shown by Papadimitriou [316], even for points in the Euclidean plane.

The TSP has a simple approximation algorithm based on "doubling" the minimum spanning tree. Since an optimal tour spans all of the sites (and is converted into a spanning tree by deleting any one edge), it must be at least as long as the minimum spanning tree; thus, by doubling the spanning tree (and shortcutting in order to obtain a tour visiting each site exactly once), we obtain a tour that is at most twice the length of the optimal TSP tour. This 2-approximation algorithm has been improved to yield a factor of 1.5 by Christofides; instead of doubling the minimum spanning tree, this method augments the tree with a minimum-weight matching on the set of odd-degree vertices in the tree. Since the resulting graph, after augmentation, is connected and has even degree, it has an Euler cycle, which is taken as the approximating tour. The approximation factor is 1.5 since a minimum-weight matching is at most 0.5 times the length of an optimal TSP tour.

For general metric spaces, the 1.5-approximation factor remains the best that is known. Until very recently, this was also the best known factor for geometric instances of the TSP. However, there are now polynomial-time approximation schemes for geometric versions of the TSP; more details are given below.

Das, Kapoor, and Smid [126] have studied the problem of r-approximating the Euclidean TSP, for large values of r: For $8 < r < n$, and for any $d \geq 1$, they prove a lower bound of $\Omega(n\log(n/r))$, in the algebraic tree model of computation, and prove that this is tight by exhibiting a matching asymptotic upper bound. (If the floor function and random access operations are permitted, then they obtain a $6\sqrt{d}n^{1-1/d}$-approximation algorithm requiring $O(n)$ time. In this more powerful model, Bern et al. [67] compute a $(2+\epsilon)$-approximate TSP in the plane in time $O((1/\epsilon)n\log\log n)$.)

An important class of heuristics for the TSP are *insertion methods*, in which sites are added one by one to an existing tour: At the ith stage, site v_i is inserted by deleting that edge (u,w) of T_{i-1} (the tour constructed on sites $\{v_1,\ldots,v_{i-1}\}$) which minimizes $d(u,v_i)+d(v_i,w)-d(u,w)$, and replacing it with the edges (u,v_i) and (v_i,w). (The initial tour T_1 is a self-loop of length zero through site v_1.) Various insertion methods are possible based on the choice of ordering of the sites for insertion. In a landmark paper, Rosenkrantz, Stearns, and Lewis [349] show that an arbitrary order of insertion of the sites gives a $(\lceil\log n\rceil+1)$-approximation of the TSP, in arbitrary metric spaces. Further, they showed that *nearest insertion* and *cheapest insertion* lead to a 2-approximation. It remained open for some time whether or not an insertion order exists that does not achieve a constant-factor approximation. Independently, Azar [47] and Bafna, Kalyanasundaram, and Pruhs [49] showed that indeed an insertion order exists that has worst-case factor $\Omega(\log n/\log\log n)$, even for instances in the Euclidean plane. Furthermore, Azar shows that the worst-case factor for *random insertion* (add the sites in random order) is $\Omega(\log\log n/\log\log\log n)$, also for points in the Euclidean plane. One of the best insertion methods in practice is *furthest insertion*, in which the site furthest from the existing tour is added at each stage; for this method, a 2.43 lower bound is known on the approximation factor for points in the plane (see Hurkens [213]).

7.3 Approximation Schemes

We now briefly survey some recent progress on approximation algorithms for geometric network optimization, which has led to polynomial-time approximation schemes (PTAS's) for several of these problems, including the TSP, Steiner tree, and k-MST problems. Many of these results are in a current state of flux, being simplified and improved. In order to keep abreast of latest developments, we encourage the reader to refer to web pages (and personal email) with the authors.

In early 1996, Arora [35] and Mitchell [289] gave PTAS's for a class of geometric optimization problems in the plane, which included the TSP, Steiner tree, and k-MST. Both of these methods were based on methods of approximating an optimal solution with one that comes from a special class of networks, and then applying dynamic programming to optimize over that class of networks. Both methods led to algorithms with running times $n^{O(1/\epsilon)}$, to obtain a $(1+\epsilon)$-approximation. The method of Mitchell [289] was based on exactly the same method as he used in earlier work ([285]) to obtain a very simple 2-approximation for the rectilinear k-MST; the only change necessary was to observe that if one replaced the "1" by an "m" in the definition of "guillotine subdivision", then the approximation factor became $(1 + 1/m)$ instead of $(1 + 1/1) = 2$. The method was also based on earlier work on "division trees" introduced by Blum, Chalasani, and Vempala [72, 290], and the guillotine *rectangular* subdivision methods of Mata and Mitchell [272].

During the last year, there have been several improvements to the original PTAS results. Mitchell [287] has reduced the running time of his method to $O(n^{O(1)})$, using a relatively minor modification to the earlier method. Arora [36] has obtained a randomized algorithm, based on a clever use of quadtrees and an improved new structure theorem, with expected running time that is *nearly linear*: $O(n \log^{O(1/\epsilon)} n)$. Further, Arora's method applies to higher dimensional versions of the problem, with an expected running time in $\Re^d$ of $O(n(\log n)^{(O(\frac{d}{\epsilon}))^{d-1}})$. The randomized algorithms can be derandomized at the cost of an extra factor of n^d in the running times.

In a very recent improvement to these results, Rao and Smith [332] obtain an $O(n \log n)$ time deterministic algorithm for any fixed ϵ and any fixed dimension d. They introduce a remarkable generalization of the notion of t-spanners – the "t-banyan" – which approximates to within factor t the interconnection cost (allowing Steiner points) for subsets of sites of *any* cardinality (not just 2 sites, as in the case of t-spanners). They prove that for any fixed $\epsilon > 0$ and $d \geq 1$, there exists a $(1+\epsilon)$-banyan having $O(n)$ vertices and $O(n)$ edges, computable in $O(n \log n)$ time.

Trevisan [375] has shown that approximation schemes in $\Re^d$ must have time bounds that are doubly exponential in d; he proves that it is NP-hard to obtain a $(1+\epsilon)$-approximation in $\Re^{O(\log n)}$, for some $\epsilon > 0$.

We should remark that, while these results are of considerable theoretical interest, it is not yet known if they hold any practical implications. The "constants" hidden in the big-Oh notation are quite large (exponential) as functions of $(1/\epsilon)$ and d.

7.4 TSP Variants and Related Geometric Problems

k-TSP, Quota-Driven TSP

The k-TSP, like the k-MST, takes as input an additional integer parameter, k, and requires that one compute a minimum-length tour that visits some subset of k sites. Optionally, a site is specified as a *root* that is required to be visited. For the graph version of the k-TSP, with edge weights obeying triangle inequality, Garg's method [165] for the k-MST yields a 3-approximation for both the rooted and unrooted version; the improvement of Arya and Ramesh [37] does not apply to the k-TSP.

A related problem is the *quota-driven salesperson* problem, in which each site has an associated integral value, w_i, and a salesperson has a given integer quota, R. The objective is to find a shortest possible cycle having the sum of the values for the sites visited is at least R. A k-TSP approximation algorithm gives also applies to this problem, since each site can be replicated w_i times; the running time is then polynomial in n and R. Another related problem is the *prize-collecting salesperson* problem, as studied by Balas [52] (see also [69]). It differs from the quota-driven salesperson problem, in that, in addition to "values" w_i, there are non-negative penalties associated with each site, and the objective function is now to minimize the sum of the distances traveled *plus* the sum of the penalties on the points *not* visited, subject to satisfying the quota R. As discussed in [46], an approximation algorithm follows from concatenating a cycle obtained for the quota-driven salesperson, with the 2-approximation cycle given by the algorithm of Goemans and Williamson [176] (which considers the effect of penalties, but does not use the quota constraint).

Orienteering Problem

In the *orienteering problem* (also known as the "bank robber" problem, or the "generalized TSP"), the traveling salesperson is allowed to travel at most a distance B, and has the objective to maximize the *number* of sites that he can visit, subject to the distance constraint. We can distinguish between the "rooted" and "unrooted" versions of the problem, depending on whether or not there is a specified site where the traveler starts. This resource-constrained optimization problem is, in a sense, dual to the problem of minimizing the length of a cycle, subject to meeting a quota on the number of sites visited (the k-TSP) or the sum of the values of the sites visited (the quota-driven salesperson problem).

For the unrooted orienteering problem, Awerbuch et al. [46] give a method for obtaining a $2c$-approximation algorithm, where c is the approximation factor for the k-TSP. For geometric instances, this, together with PTAS results, implies a $(2+\epsilon)$-approximation algorithm for the unrooted case. The first results on the rooted case have recently been given by Arkin et al. [28], who obtain a 2-approximation for both the rooted and unrooted cases, for geometric instances of the problem. Their methods rely on recent results on m-guillotine subdivisions. (It remains an open question whether or not the rooted orienteering problem has any approximation algorithm in general graphs.)

MAX TSP

The *MAX TSP* changes the objective in the ordinary TSP from that of minimizing to that of *maximizing* the length of a tour that visits every point of S. For the MAX TSP in graphs, the problem is easily seen (from Hamiltonian cycle) to be NP-complete, even if edge lengths obey the triangle inequality; however, a 5/7-approximation algorithm has recently been obtained [195, 243].

For geometric instances of the MAX TSP, Alon et al. [17] provide a constant-factor approximation algorithm for MAX TSP, as well as the problem of computing a longest *non*crossing tour. Barvinok [57] has obtained a PTAS for the MAX TSP, computing a tour whose length is guaranteed to be at least $(1-\epsilon)$ times optimal, for any fixed $\epsilon > 0$; his algorithm applies in any fixed dimension d, for L_p metrics, and more general metrics too. Most recently, Barvinok et al. [56] have resolved the complexity of the MAX TSP for geometric instances of the problem in metrics defined by a convex polytope, in any fixed dimension d; they provide a *polynomial-time* algorithm to solve the problem exactly in time $O(n^{f-2}\log n)$, where f is the number of facets defining the polytope. For example, for the L_1 or L_∞ metric in the plane, their algorithm requires $O(n^2 \log n)$ time. (This has been improved recently to an optimal, $O(n)$-time, algorithm of Fekete [154], which applies to the planar L_1 and L_∞ problems, using special structure of the problem in the plane.) Their approach is to reduce the problem to solving a set of maximum weight b-matching problems (which are actually transportation problems) on an appropriate (bipartite) graph. This remarkable result is one of the rare cases in which a TSP problem can be solved exactly in polynomial time. The complexity of the *Euclidean* MAX TSP has been open. Very recently, Fekete [154] was able to prove that the Euclidean MAX TSP is NP-hardness in dimension three or greater. The complexity remains open in the Euclidean plane:

Open Problem 22 *What is the complexity of the MAX TSP in the Euclidean plane? What if the tour is required to be noncrossing?*

We should remark, though, that there is an issue about the model of computation here since, for both the TSP and MAX TSP, we do not know if the problem lies in NP for the Euclidean metric. In particular, for sites having rational coordinates, in order to compare the length of a tour to a given rational number, we must evaluate a sum of n square roots to sufficient precision; the best known bound on the number of bits of precision in order to guarantee a correct answer is exponential in n. It may be that the Euclidean MAX

TSP can be solved in polynomial time if we assume that arithmetic operations (including square roots and comparisons) can be done in constant time, but the algorithm may require exponential time on a standard Turing machine.

Bottleneck and Maximum Scatter TSP

In the *bottleneck TSP*, the goal is to obtain a tour minimizing the length of the longest edge in the tour. In graphs, the problem is NP-complete [250]. If the edge lengths do not satisfy the triangle inequality, then no constant factor approximation algorithm can exist, unless P=NP. If the edge lengths do satisfy the triangle inequality, then Parker and Rardin [321] have given a 2-approximation algorithm and shown that this is best possible (unless P=NP). For the geometric version of the problem, it is easy to show that the problem is NP-hard, from the fact that Hamiltonian cycle in grid graphs is hard [250].

In the *maximum scatter TSP*, the goal is to obtain a tour *maximizing* the length of the *shortest* edge in the tour. (Such problems arise, e.g., in sequencing rivet operations.) Arkin et al. [23] have studied the problem in graphs, showing that the problem is NP-complete, that the general problem has no polynomial-time constant-factor approximation algorithm (unless P=NP), and that if edge lengths obey the triangle inequality there is a 2-approximation (a tour whose shortest edge has length at least one half that of optimal), which is best possible. However, it is not yet known if the *geometric* version of the problem is hard. Also, while the factor 2 approximation algorithms are best possible in graphs whose edge lengths obey the triangle inequality, the current approximation algorithm results for bottleneck TSP and maximum scatter TSP do not exploit geometric structure, which may lead to an improved factor:

Open Problem 23 *What is the complexity of the maximum scatter TSP for points in the plane? Can one obtain approximation factors better than 2 for geometric instances of bottleneck TSP or maximum scatter TSP?*

Minimum Latency Problem

In the *minimum latency problem* (MLT), also known as the *deliveryman problem* and the *traveling repairman problem*, the goal is to find a tour on S that minimizes the sum of the "latencies" of all points, where the *latency* of a point p is the length of the tour from a given starting point to the point p. (Thus, the latency of a point measures how long a job at that point must wait before being served by the repairman/deliveryman that is traveling along the tour.) For the problem in graphs, Blum et al. [71] have given a 128-approximation algorithm; this has been improved by Goemans and Kleinberg [177], who obtain a factor of 29. By a direct application of Theorem 2 of [71], which states that a c-approximation for the k-MST implies an $8c$-approximation for the MLP, we see that recent PTAS results on the geometric k-MST imply an $(8+\epsilon)$-approximation for geometric instances of MLP. It is an interesting open problem to improve on this factor:

Open Problem 24 *Is there a PTAS for the minimum latency problem on a set of points in the Euclidean plane?*

Area Optimization Problems

In the *min-area TSP* (resp., *max-area TSP*), the goal is to determine a cycle on a given set S of points such that the cycle defines a simple polygon of minimum (resp., maximum) area. Fekete [153] (in part together with Pulleyblank [155]) has studied these problems extensively. He has shown that both the min-area and max-area TSP problems are NP-complete.

For the max-area TSP, Fekete gives a (1/2)-approximation algorithm, showing how, in $O(n \log n)$ time, one can obtain a cycle surrounding area that is at least half that of the convex hull of S. Further, he shows that it is NP-complete to decide if one can obtain a simple polygon whose area is more than $(2/3+\epsilon)$ times that of the convex hull.

For the min-area TSP, Fekete conjectures that no polynomial-time approximation algorithm exists (unless P=NP); he bases this conjecture on evidence suggested by a potentially related result of his: The min-area disjoint triangle matching problem (to determine a minimum-area set of disjoint triangles on $3n$ points) has no approximation algorithm (unless P=NP).

Open Problem 25 *Is there a polynomial-time approximation algorithm for the min-area TSP?*

Angular-Metric TSP and Angle-Restricted Tours

In the *angular-metric TSP*, the goal is to determine a cycle on a given set S of points such that the sum of the direction changes at each vertex (point of S) is minimized. Aggarwal et al. [9] have proven that this problem is NP-complete.

In the *angle-restricted tour* (ART) problem, one is given a set A of allowable angles and asks if a tour exists on the set S such that every angle between consecutive edges of the tour lies in the set A; such a tour is called an *A-tour.* (This problem is related to that of generating a simple polygon, given a sequence of angles; see Culberson and Rawlins [124].) Fekete [152] and Fekete and Woeginger [156] have studied the class of ART problems, showing that (1) if $|S| \neq 4$ and $A = [0, \pi]$, then an A-tour always exists for any finite S; (2) if $A = \{-\pi/2, \pi/2\}$, then, based on results of O'Rourke [309], there exists a polynomial-time algorithm to detect if S admits an A-tour; (3) if $A = \{-\pi/2, \pi/2, \pi\}$, $\{-\pi/2, \pi\}$. or $\{\pi, \pi/2\}$, then it is NP-complete to decide if S admits an A-tour; and (4) if $A = (-\pi/2, \pi/2)$ (the "acute" case) or $A = (-\pi, -\pi/2) \cup (\pi/2, \pi]$ (the "obtuse" case), there are arbitrarily large points sets S that do *not* admit an A-tour (an "acute tour" or an "obtuse tour"). Their work suggests a number of open questions, including:

Open Problem 26 *What is the computational complexity of deciding whether a point set has an acute (or obtuse) tour?*

TSP with Neighborhoods

In the *TSP with neighborhoods*, the goal is to find a shortest tour that visits at least one point in each of a set of k (possibly overlapping) *neighborhoods.* The neighborhoods may be connected sets (e.g., disks or polygons), or possibly disconnected sets (e.g., pairs of discrete points, or sets of disjoint polygons). Since it is a generalization of TSP, the problem is clearly NP-hard.

When the neighborhoods are connected and "well behaved" (e.g., disks, or having roughly equal-length and parallel diameter segments), Arkin and Hassin [26] have obtained $O(1)$-approximation algorithms, with running time $O(n + k \log k)$, where n is the total complexity of the k neighborhoods. Further, they prove a form of "combination lemma" that allows one to consider unions of sets of well-behaved neighborhoods; the resulting approximation factor is given by the sum of the approximation factors obtained for each class individually. For the general case of connected polygonal neighborhoods, Mata and Mitchell [272] obtained an $O(\log k)$-approximation algorithm, based on "guillotine rectangular subdivisions," with time bound $O(n^5)$. Gudmundsson and Levcopoulos [184] have recently obtained a faster method, which, for any fixed $\epsilon > 0$, is guaranteed to perform at least one of the following two tasks (although one does not know in advance which one will be accomplished): (1) it outputs in time $O(n + k \log k)$ a tour with length at most $O(\log k)$ times optimal; or (2) it outputs a tour with length at most $(1 + \epsilon)$ times optimal, in time $O(n^3)$ (if $\epsilon \leq 3$) or $O(n^2 \log n)$ (if $\epsilon > 3$). However, no polynomial-time method guaranteeing a constant factor approximation is known for general neighborhoods.

If the neighborhoods are disconnected, then the problem seems to be even more difficult. The problem then is called the "one-of-a-set TSP" or the "group TSP," referring to the fact that the tour is required only to visit one point from each set (group). It has also been called the *errand scheduling problem* (see Slavík [364]), since it models the problem of finding the best order in which to perform a set of errands, each of which can be performed at some subset of the nodes of an edge-weighted graph. The one-of-a-set TSP in graphs generalizes both the set cover problem and the TSP. It is a special case of the "traveling purchaser problem" [307], in which a traveler is required to purchase each item on a shopping list, by visiting an appropriate subset of sites, in hopes of minimizing the total cost of travel plus the amount paid for the items; each site has a given inventory of items, and the prices of items vary from site to site. Polylogarithmic approximation algorithms follow from the results known on the closely related "group Steiner tree" problem (Section 7.1). Further, since the problem generalizes set cover, we cannot hope for a better approximation ratio than $\Theta(\log k)$ in general graphs ([270]). However, for sets of points in the plane, no reduction from set cover is known; it is possible that there is a constant-factor approximation algorithm for general discrete sets of points in the plane. We note that a constant-factor approximation algorithm for the group Steiner tree problem implies a constant-factor approximation for the one-of-a-set TSP (just by doubling the Steiner tree to obtain a tour).

Open Problem 27 *Does the TSP with (connected) neighborhoods problem have a polynomial-time $O(1)$-approximation algorithm? What if the neighborhoods are not connected sets (e.g., if the neighborhoods are discrete sets of points)?*

Lawnmowing and Milling

In the *lawnmowing problem*, the goal is to find a shortest cycle (or path) for the motion of a disk (representing a "lawnmower") such that every point of a given (possibly disconnected) region R is covered by the disk at some position of the disk. It is easy to see that the problem is NP-hard, in general; R may be a set of n well separated points, making the problem that of a TSP with disjoint circular neighborhoods. However, Arkin, Fekete, and Mitchell [24, 25] have shown that the problem is NP-hard, even if R is simply connected (e.g., a simple polygon). Their proof also applies to the *milling problem*, which adds the constraint that the cutter stay within R, in a multiply-connected region. What is not yet known, though, is if the milling problem is hard when R is *simply* connected:

Open Problem 28 *Is the milling problem NP-hard for a region R that is simply connected?*

A recent result that is potentially related to this question is that of Umans and Lenhart [376], who have shown that one can determine Hamiltonicity of a "solid grid graph" (a grid graph induced by the points that lie inside a simply connected region) in polynomial time.

Approximation algorithms for the lawnmowing problem allow one to get within a constant factor of optimal [24, 25, 224]; the best current factor is $(3 + \epsilon)$, based on the algorithm of Arkin, Fekete, and Mitchell [25], together with recent PTAS results for TSP. They also give a 2.5-approximation algorithm for the milling problem; the approximation factor becomes 11/5 if R is a rectilinear simple polygon.

The model of the milling problem as discussed above is oversimplified. In practice, there are several other issues in the milling process to consider; these are discussed in detail in the book of Held [196], who introduced computational geometry techniques to the pocket machining problem. (See also the survey by Guyder [188] and two recent special issues (March'94 and November'94) of the journal *CAD* devoted to machining.) It is important in practice to avoid *re-milling* a region where the cutter has already been, as this can damage the finished surface; the model of [24, 25] allows for re-milling. The most common strategies used in practice are based on *contour-parallel* ("window-pane") milling, in which the cutter follows the boundary of the region and spirals inward, and *zig-zag* ("axis-parallel", "staircase") milling, in which the cutter sweeps out strips parallel to the coordinate axis. It both of these methods, if one is to avoid re-milling portions of the pocket, the cutting tool must be *retracted* and moved (rapidly) to a new location, where it can then resume cutting. In the case of zig-zag milling, Arkin, Held, and Smith [27] have considered the problem of finding tool paths that minimize the number of tool retractions. They prove that the problem is NP-hard in general, but give constant-factor approximation algorithms that are shown experimentally to perform well in practice.

Watchman Route Problem

In the *watchman route (path) problem*, the goal is to find a shortest possible cycle (path) within a polygonal domain P, such that every point of P is seen by some point of the cycle. This problem is seen to be closely related to the TSP with neighborhoods, since it can be thought of as a shortest-path/cycle problem in which we have the constraint that the path/cycle must visit the visibility region associated with each point of the domain.

The watchman route problem was first investigated by Chin and Ntafos, who give an $O(n)$-time algorithm if P is a rectilinear simple polygon [110], and claimed an $O(n^4)$-time algorithm if P is a simple polygon and we are given an *anchor* ("door") point, p, on the boundary of P, through which the cycle is required to go [112]. Their algorithms are based on identifying a set of ***essential cuts***, which are directed line segment chords (extensions of selected polygon edges) that any watchman route must visit, in the order that they appear about the polygon. Further, a locally shortest watchman route that visits an essential cut without crossing it must do so either at a point where the cut intersects another cut, or at a point where the route "reflects" on the cut, with equal incidence and reflection angles. Given a subsequence of "active" essential cuts, where reflection is to occur, the polygon can be "unrolled", resulting in a shortest path problem in a simple polygon, which is readily solved in linear time (Section 2). On the basis of these facts, Chin and Ntafos devise an incremental "adjustment" algorithm, using an "adjust at the first choice" rule, for searching for the combinatorial type of a shortest watchman route, anchored at p. Subsequently, Tan, Hirata, and Inagaki [373] claimed an improved time complexity of $O(n^3)$, based on showing a linear bound on the number of adjustments required. Then, Tan and Hirata [370] developed a divide-and-conquer algorithm, with claimed time complexity of $O(n^2)$ for this same anchored version of the watchman route problem. Building on these results, Carlsson, Jonsson, and Nilsson [84] (see the updated version in [301]) gave a method of removing the assumption that there is a prescribed anchor point through which the watchman must go; they claim an $O(n^4)$-time algorithm for computing an unrestricted ("floating") shortest watchman cycle, basing it on the $O(n^2)$ result of [370] for the anchored case.

Recent work of Hammar and Nilsson [189] has shown, however, that there is a flaw in the earlier works, starting with the paper of Chin and Ntafos [112], and including subsequent work [373, 370, 84, 301]. Chin and Ntafos had made an observation, which is not true in general, about a monotonicity in the directions of adjustments along cuts. Hammar and Nilsson give an example in which oscillations can occur in the optimization, causing the algorithm of Chin and Ntafos to exhibit exponential behavior. Hammar and Nilsson also propose a new adjustment rule to fix the problem; however, it too suffers from a similar problem. Tan, Hirata, and Inagaki [372] have proposed a new approach to fixing the problem, based on a dynamic programming optimization over "partial watchman routes" on a cut. Applying this fix to their earlier incremental algorithm ([373]) results in a time complexity of $O(n^4)$ for the anchored watchman route problem. (Applying the fix to the algorithm of Chin and Ntafos results in time $O(n^5)$.) To our knowledge, this is the current best time bound that is generally accepted to be correct. (The divide-and-conquer approach in [370] does not yet yield an error-free algorithm with an improved time bound.) The time complexity of the unrestricted (floating) watchman route algorithm ([84, 301]) also goes up by a quadratic factor, as a result of the fix, to $O(n^6)$.

In the case of a polygonal domain P with holes, the problem is easily seen to be NP-hard (from Euclidean TSP) [110]. Mata and Mitchell [272] obtain an $O(\log n)$-approximation algorithm, for a version that considers "rectilinear visibility", using dynamic programming on a class of guillotine rectangular subdivisions. In light of the recent PTAS results for TSP, based on guillotine subdivisions and their generalizations, it is interesting to ask if improved approximation results can now be improved:

Open Problem 29 *Does the watchman route problem in a polygonal domain have a polynomial-time $O(1)$-approximation algorithm? Is there a PTAS?*

For the watchman *path* problem in a simple polygon, Carlsson and Jonsson [83] have obtained the first polynomial-time algorithm. (The claimed time bound is $O(n^{12})$, but, since it too is based on previous work having errors pointed out by [189, 372], the actual time complexity of a corrected version is likely to have an additional quadratic factor.) The problem turns out to be considerably more difficult than the cycle version, since one must determine the order in which the path visits essential cuts. An interesting problem that is solved as part of this work is the *shortest postman path* problem, in which the goal is to find a shortest path that visits all vertices of P; [83] give an $O(n^3)$ algorithm.

The *robber route problem* [303] requires that the watchman see only a specified subset of the edges of P, while avoiding to be seen from a set of point "threats" within P; see also Gewali et al. [168].

There have been several other results on generalizations and variations of the watchman route problem, including watchman routes in "weak visibility" polygons [245], in "spiral" polygons [302], external to a simple polygon [305], external to two convex polygons [170], under limited visibility distance [304], and for multiple watchmen [85, 297, 301] (in restricted simple polygons). Also, minimum-link watchman tours have been studied. In simple polygons, Alsuwaiyel and Lee [18, 19] show that the problem is NP-hard and give an $O(1)$-approximation algorithm. In polygonal domains, Arkin, Mitchell, and Piatko [29] show the problem to be NP-hard (even for convex obstacles) and give an $O(\log n)$-approximation algorithm. We know of no results yet, though, for approximating watchman routes in three dimensions:

Open Problem 30 *Give an efficient approximation algorithm for watchman routes in polyhedral domains in 3-space.*

Zookeeper's Problem

The *zookeeper's problem* is to find a shortest cycle in a simple polygon P (the *zoo*), through a given vertex v (the *zookeeper's chair*), such that the cycle visits every one of a set of k disjoint convex polygons (*cages*), each sharing an edge with P, without entering any of the cages.

This problem is a special case of the TSP with neighborhoods problem, constrained within a simple polygon. The simple polygon constraint helps to simplify the problem; it implies that an optimal cycle must visit the cages in the same order that they appear on the boundary of P (otherwise, the cycle would self-intersect and could be shortened). This observation, together with the reflection principle, allowed Chin and Ntafos [111] to solve the problem with a procedure that searches for the combinatorial type (sequence of cage edges) of an optimal cycle by incremental updates, at each stage computing a shortest path in a 2-manifold that results from "unfolding" a sequence of cage edges (according to the reflection principle). By showing that there are only $O(n)$ updates, and each can be done in time $O(n)$, they achieve an $O(n^2)$-time algorithm. Hershberger and Snoeyink [204] have shown how each update can be done in time $O(\log^2 n)$, resulting in overall time $O(n \log^2 n)$, using the structure of hourglasses in simple polygons, and the shortest path query data structures of [185, 200]

Open Problem 31 *Can the zookeeper problem be solved in time $O(n)$?*

The related *safari route problem*, introduced by Ntafos [304], requires that the cycle visit all of the cages, but it allows travel through a "cage." If the cages are not attached to the boundary of a simple polygon P, then we get the (NP-hard) TSP with neighborhoods. However, for cages that are attached to P, Ntafos [304] obtains an $O(kn^2)$ algorithm. Tan and Hirata [371] improve the time bound to $O(n^2)$, while removing the constraint that the cycle pass through a given point v on the boundary of P.

Aquarium Keeper's Problem

The *aquarium keeper's problem* is to find a shortest cycle in a simple polygon P (the *aquarium*), such that the cycle touches *every* edge of P. Thus, the aquarium keeper's problem is a special case of the zookeeper's problem (and thus of the TSP with neighborhoods) in which there is a "cage" erected on *every* edge of P, and each cage consists simply of the edge itself. This extra structure allows an $O(n)$-time solution, as shown by Czyzowicz et al. [125], since there is no issue of obtaining the combinatorial type of the path. Carlsson and Jonsson [83] solve the *path* version of the problem in time $O(n^4)$; there is added complexity in searching for a shortest path, since we do not know the order in which a path must visit the edges of P.

ACKNOWLEDGEMENTS

I want thank several people who have contributed suggestions that have improved this survey: Pankaj Agarwal, Yi-Jen Chiang, Sariel Har-Peled, Christian Icking, Doug Ierardi, Andrei Iones, Peter Johansson, Rolf Klein, Kevin Klenk, Sylvain Lazard, Bengt Nilsson, Joe O'Rourke, Jörg Sack, Sven Schuierer, Jürgen Sellen, Chee Yap, and Sergey Zhukov.

References

[1] J. Abello and K. Kumar. Visibility graphs and oriented matroids. In R. Tamassia and I. G. Tollis, editors, *Graph Drawing (Proc. GD '94)*, volume 894 of *Lecture Notes Comput. Sci.*, pages 147–158. Springer-Verlag, 1995.

[2] P. Agarwal and M. Sharir. Arrangements. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, page ?? Elsevier Science Publishers B.V. North-Holland, Amsterdam, 1998.

[3] P. K. Agarwal, B. Aronov, J. O'Rourke, and C. A. Schevon. Star unfolding of a polytope with applications. *SIAM J. Comput.*, 26:1689–1713, 1997.

[4] P. K. Agarwal, T. Biedl, S. Lazard, S. Robbins, S. Suri, and S. Whitesides. Curvature-constrained shortest paths in a convex polygon. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, page to appear, 1998.

[5] P. K. Agarwal, H. Edelsbrunner, O. Schwarzkopf, and E. Welzl. Euclidean minimum spanning trees and bichromatic closest pairs. *Discrete Comput. Geom.*, 6(5):407–422, 1991.

[6] P. K. Agarwal, S. Har-Peled, M. Sharir, and K. R. Varadarajan. Approximate shortest paths on a convex polytope in three dimensions. *J. ACM*, 44:567–584, 1997.

[7] P. K. Agarwal, J. Matoušek, and S. Suri. Farthest neighbors, maximum spanning trees and related problems in higher dimensions. *Comput. Geom. Theory Appl.*, 1(4):189–201, 1992.

[8] P. K. Agarwal, P. Raghavan, and H. Tamaki. Motion planning for a steering-constrained robot through moderate obstacles. In *Proc. 27th Annu. ACM Sympos. Theory Comput.*, pages 343–352, 1995.

[9] A. Aggarwal, D. Coppersmith, S. Khanna, R. Motwani, and B. Schieber. The angular-metric traveling salesman problem. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 221–229, Jan. 1997.

[10] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, Englewood Cliffs, NJ, 1993.

[11] L. Aleksandrov, M. Lanthier, A. Maheshwari, and J.-R. Sack. An ϵ-approximation algorithm for weighted shortest path queries on polyhedral surfaces. In *Abstracts 14th European Workshop Comput. Geom.*, pages 19–21, 1998.

[12] L. Aleksandrov, M. Lanthier, A. Maheshwari, and J.-R. Sack. An ϵ-approximation algorithm for weighted shortest paths on polyhedral surfaces. In *Proc. 6th Scand. Workshop Algorithm Theory*, volume ?? of *Lecture Notes Comput. Sci.*, page to appear. Springer-Verlag, 1998.

[13] R. Alexander. *Construction of Optimal-Path Maps for Homogeneous-Cost-Region Path-Planning Problems.* Ph.D. thesis, Computer Science, U.S. Naval Postgraduate School, Monterey, CA, 1989.

[14] R. Alexander and N. Rowe. Geometrical principles for path planning by optimal-path-map construction for linear and polygonal homogeneous-region terrain. Technical report, Computer Science, U.S. Naval Postgraduate School, Monterey, CA, 1989.

[15] R. Alexander and N. Rowe. Path planning by optimal-path-map construction for homogeneous-cost two-dimensional regions. In *Proc. IEEE Internat. Conf. Robot. Autom.*, 1990.

[16] N. Alon and Y. Azar. On-line Steiner trees in the Euclidean plane. *Discrete Comput. Geom.*, 10:113–121, 1993.

[17] N. Alon, S. Rajagopalan, and S. Suri. Long non-crossing configurations in the plane. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 257–263, 1993.

[18] M. H. Alsuwaiyel and D. T. Lee. Minimal link visibility paths inside a simple polygon. *Comput. Geom. Theory Appl.*, 3(1):1–25, 1993.

[19] M. H. Alsuwaiyel and D. T. Lee. Finding an approximate minimum-link visibility path inside a simple polygon. *Inform. Process. Lett.*, 55(2):75–79, 1995.

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

[21] S. R. Arikati, D. Z. Chen, L. P. Chew, G. Das, M. H. M. Smid, and C. D. Zaroliagis. Planar spanners and approximate shortest path queries among obstacles in the plane. In J. Díaz and M. Serna, editors, *Algorithms—ESA '96, Fourth Annual European Symposium*, volume 1136 of *Lecture Notes Comput. Sci.*, pages 514–528, Barcelona, Spain, Sept. 1996. Springer-Verlag.

[22] E. M. Arkin, Y.-J. Chiang, J. S. B. Mitchell, and S. S. Skiena. A note on the group steiner problem. Manuscript, University at Stony Brook, 1997.

[23] E. M. Arkin, Y.-J. Chiang, J. S. B. Mitchell, S. S. Skiena, and T. Yang. On the maximum scatter TSP. In *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, pages 211–220, 1997.

[24] E. M. Arkin, S. P. Fekete, and J. S. B. Mitchell. The lawnmower problem. In *Proc. 5th Canad. Conf. Comput. Geom.*, pages 461–466, 1993.

[25] E. M. Arkin, S. P. Fekete, and J. S. B. Mitchell. Approximation algorithms for lawn mowing and milling. Technical report, Mathematisches Institut, Universität zu Köln, 1997.

[26] E. M. Arkin and R. Hassin. Approximation algorithms for the geometric covering salesman problem. *Discrete Appl. Math.*, 55:197–218, 1994.

[27] E. M. Arkin, M. Held, and C. L. Smith. Optimization problems related to zigzag pocket machining. In *Proc. 7th ACM-SIAM Sympos. Discrete Algorithms*, pages 419–428, 1996.

[28] E. M. Arkin, J. S. B. Mitchell, and G. Narasimhan. Resource-constrained geometric network optimization. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, page to appear, 1998.

[29] E. M. Arkin, J. S. B. Mitchell, and C. Piatko. Minimum-link watchman tours. Report, University at Stony Brook, 1994.

[30] E. M. Arkin, J. S. B. Mitchell, and C. D. Piatko. Bicriteria shortest path problems in the plane. In *Proc. 3rd Canad. Conf. Comput. Geom.*, pages 153–156, 1991.

[31] E. M. Arkin, J. S. B. Mitchell, and S. Suri. Logarithmic-time link path queries in a simple polygon. *Internat. J. Comput. Geom. Appl.*, 5(4):369–395, 1995.

[32] B. Aronov. On the geodesic Voronoi diagram of point sites in a simple polygon. *Algorithmica*, 4:109–140, 1989.

[33] B. Aronov, S. J. Fortune, and G. Wilfong. Furthest-site geodesic Voronoi diagram. *Discrete Comput. Geom.*, 9:217–255, 1993.

[34] B. Aronov and J. O'Rourke. Nonoverlap of the star unfolding. *Discrete Comput. Geom.*, 8:219–250, 1992.

[35] S. Arora. Polynomial time approximation schemes for Euclidean TSP and other geometric problems. In *Proc. 37th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 2–11, 1996.

[36] S. Arora. Nearly linear time approximation schemes for Euclidean TSP and other geometric problems. In *Proc. 38th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 554–563, 1997.

[37] S. Arya and H. Ramesh. A 2.5 factor approximation algorithm for the k-MST problem. *Inform. Process. Lett.*, 65(3):117–118, 1998.

[38] T. Asano, T. Asano, L. J. Guibas, J. Hershberger, and H. Imai. Visibility of disjoint polygons. *Algorithmica*, 1:49–63, 1986.

[39] T. Asano, B. K. Bhattacharya, J. M. Keil, and F. Yao. Clustering algorithms based on minimum and maximum spanning trees. In *Proc. 4th Annu. ACM Sympos. Comput. Geom.*, pages 252–257, 1988.

[40] T. Asano, D. Kirkpatrick, and C. K. Yap. d_1-optimal motion for a rod. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.*, pages 252–263, 1996.

[41] T. Asano and G. Toussaint. Computing the geodesic center of a simple polygon. In D. S. Johnson, editor, *Discrete Algorithms and Complexity*, Perspectives in Computing, pages 65–79. Academic Press, 1987.

[42] M. Atallah and D. Chen. On parallel rectilinear obstacle-avoiding paths. *Comput. Geom. Theory Appl.*, 3:307–313, 1993.

[43] M. J. Atallah. Parallel algorithms in computational geometry. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, page ?? Elsevier Science Publishers B.V. North-Holland, Amsterdam, 1998.

[44] M. J. Atallah and D. Z. Chen. Parallel rectilinear shortest paths with rectangular obstacles. *Comput. Geom. Theory Appl.*, 1:79–113, 1991.

[45] F. Aurenhammer and R. Klein. Voronoi diagrams. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, page ?? Elsevier Science Publishers B.V. North-Holland, Amsterdam, 1998.

[46] B. Awerbuch, Y. Azar, A. Blum, and S. Vempala. Improved approximation guarantees for minimum-weight k-trees and prize-collecting salesmen. In *Proc. 27th Annu. ACM Sympos. Theory Comput.*, pages 277–283, 1995.

[47] Y. Azar. Lower bounds for insertion methods for TSP. *Combinatorics, Probability and Computing*, 3:285–292, 1994.

[48] R. Baeza-Yates, J. Culberson, and G. Rawlins. Searching in the plane. *Inform. Comput.*, 106:234–252, 1993.

[49] V. Bafna, B. Kalyanasundaram, and K. Pruhs. Not all insertion methods yield constant approximate tours in the Euclidean plane. *Theoret. Comput. Sci.*, 125:345–353, 1994.

[50] C. Bajaj. The algebraic complexity of shortest paths in polyhedral spaces. In *Proc. 23rd Allerton Conf. Commun. Control Comput.*, pages 510–517, 1985.

[51] C. Bajaj. The algebraic degree of geometric optimization problems. *Discrete Comput. Geom.*, 3:177–191, 1988.

[52] E. Balas. The prize collecting traveling salesman problem. *Networks*, 19:621–636, 1989.

[53] E. Bar-Eli, P. Berman, A. Fiat, and P. Yan. Online navigation in a room. *J. Algorithms*, 17:319–341, 1994.

[54] R. Bar-Yehuda and B. Chazelle. Triangulating disjoint Jordan chains. *Internat. J. Comput. Geom. Appl.*, 4(4):475–481, 1994.

[55] J. Barraquand and J.-C. Latombe. Nonholonomic multi-body mobile robots: controllability and motion planning in the presence of obstacles. *Algorithmica*, 10:121–155, 1993.

[56] A. Barvinok, D. S. Johnson, G. J. Woeginger, and R. Woodroofe. The maximum traveling salesman problem under polyhedral norms. In *Sixth International Conference on Integer Programming and Combinatorial Optimization*, volume 1412 of *Lecture Notes Comput. Sci.*, pages 195–201, Rice University, Houston, TX, June 1998. Springer-Verlag.

[57] A. I. Barvinok. Two algorithmic results for the TSP. *Math. Oper. Res.*, 21:65–84, 1996.

[58] C. D. Bateman, C. S. Helvig, G. Robins, and A. Zelikovsky. Provably good routing tree construction with multi-port terminals. In *Proc. ACM/SIGDA International Symposium on Physical Design*, page ??, Apr. 1997.

[59] J. L. Bentley. Fast algorithms for geometric traveling salesman problems. *ORSA J. Comput.*, 4(4):387–411, 1992.

[60] P. Berman. On-line searching and navigation. In A. Fiat and G. Woeginger, editors, *Competitive Analysis of Algorithms.* Springer-Verlag, 1998.

[61] P. Berman, A. Blum, A. Fiat, H. Karloff, A. Rosen, and M. Saks. Randomized robot navigation algorithms. In *Proc. 7th ACM-SIAM Sympos. Discrete Algorithms*, pages 75–84, 1996.

[62] P. Berman and M. Karpinski. Randomized navigation to a wall through convex obstacles. Technical Report 85118-CS, Bonn University, 1994.

[63] M. Bern. Triangulations. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 22, pages 413–428. CRC Press LLC, Boca Raton, FL, 1997.

[64] M. Bern and D. Eppstein. Approximation algorithms for geometric problems. In D. S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*, pages 296–345. PWS Publishing Company, Boston, MA, 1997.

[65] M. Bern and P. Plassman. The Steiner problem with edge lengths 1 and 2. *Inform. Process. Lett.*, 32:171–176, 1989.

[66] M. Bern and P. Plassmann. Mesh generation. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, page ?? Elsevier Science Publishers B.V. North-Holland, Amsterdam, 1998.

[67] M. W. Bern, H. J. Karloff, P. Raghavan, and B. Schieber. Fast geometric approximation techniques and geometric embedding problems. *Theoret. Comput. Sci.*, 106(2):265–281, Dec. 1992.

[68] A. Bezdek. On optimal route planning evading cubes in the three space. *Beiträge zur Algebra und Geometrie*, ??:to appear, ??

[69] D. Bienstock, M. X. Goemans, D. Simchi-Levi, and D. Williamson. A note on the prize collecting traveling salesman problem. *Math. Prog.*, 59:413–420, 1993.

[70] A. Blum and P. Chalasani. An on-line algorithm for improving performance in navigation. In *Proc. 34th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 2–11, 1993.

[71] A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, and M. Sudan. The minimum latency problem. In *Proc. 26th Annu. ACM Sympos. Theory Comput. (STOC 94)*, pages 163–171, 1994.

[72] A. Blum, P. Chalasani, and S. Vempala. A constant-factor approximation for the k-MST problem in the plane. In *Proc. 27th Annu. ACM Sympos. Theory Comput.*, pages 294–302, 1995.

[73] A. Blum, P. Raghavan, and B. Schieber. Navigating in unfamiliar geometric terrain. *SIAM J. Comput.*, 26(1):110–137, Feb. 1997.

[74] M. Blum and D. Kozen. On the power of the compass (or, why mazes are easier to search than graphs). In *Proc. 19th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 132–142, 1978.

[75] J.-D. Boissonnat, A. Cérézo, and J. Leblond. Shortest paths of bounded curvature in the plane. *Internat. J. Intell. Syst.*, 10:1–16, 1994.

[76] J.-D. Boissonnat, J. Czyzowicz, O. Devillers, J.-M. Robert, and M. Yvinec. Convex tours of bounded curvature. In *Proc. 2nd Annu. European Sympos. Algorithms*, volume 855 of *Lecture Notes Comput. Sci.*, pages 254–265. Springer-Verlag, 1994.

[77] J.-D. Boissonnat and S. Lazard. A polynomial-time algorithm for computing a shortest path of bounded curvature amidst moderate obstacles. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.*, pages 242–251, 1996.

[78] P. Bose, W. Evans, D. Kirkpatrick, M. McAllister, and J. . Snoeyink. Approximating shortest paths in arrangements of lines. In *Proc. 8th Canad. Conf. Comput. Geom.*, pages 143–148. Carleton University Press, Ottawa, Canada, 1996.

[79] P. B. Callahan and S. R. Kosaraju. Faster algorithms for some geometric graph problems in higher dimensions. In *Proc. 4th ACM-SIAM Sympos. Discrete Algorithms*, pages 291–300, 1993.

[80] J. Canny. Some algebraic and geometric computations in PSPACE. In *Proc. 20th Annu. ACM Sympos. Theory Comput.*, pages 460–467, 1988.

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

[82] J. Canny and J. H. Reif. New lower bound techniques for robot motion planning problems. In *Proc. 28th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 49–60, 1987.

[83] S. Carlsson and H. Jonsson. Computing a shortest watchman path in a simple polygon in polynomial-time. In *Proc. 4th Workshop Algorithms Data Struct.*, volume 955 of *Lecture Notes Comput. Sci.*, pages 122–134. Springer-Verlag, 1995.

[84] S. Carlsson, H. Jonsson, and B. J. Nilsson. Finding the shortest watchman route in a simple polygon. In *Proc. 4th Annu. Internat. Sympos. Algorithms Comput.*, volume 762 of *Lecture Notes Comput. Sci.*, pages 58–67. Springer-Verlag, 1993.

[85] S. Carlsson, B. J. Nilsson, and S. Ntafos. Optimum guard covers and m-watchmen routes for restricted polygons. *Internat. J. Comput. Geom. Appl.*, 3:85–105, 1993.

[86] K. F. Chan and T. W. Lam. An on-line algorithm for navigating in unknown environment. *Internat. J. Comput. Geom. Appl.*, 3:227–244, 1993.

[87] M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li. Approximation algorithms for directed Steiner problems. In *Proc. 9th ACM-SIAM Sympos. Discrete Algorithms*, page to appear, 1998.

[88] M. Charikar, C. Chekuri, A. Goel, and S. Guha. Rounding via tree: Deterministic approximation algorithms for group Steiner trees and k-median. In *Proc. 30th Annu. ACM Sympos. Theory Comput.*, page to appear, 1998.

[89] B. Chazelle. A theorem on polygon cutting with applications. In *Proc. 23rd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 339–349, 1982.

[90] B. Chazelle. An algorithm for segment-dragging and its implementation. *Algorithmica*, 3:205–221, 1988.

[91] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, 6:485–524, 1991.

[92] B. Chazelle, H. Edelsbrunner, M. Grigni, L. J. Guibas, J. Hershberger, M. Sharir, and J. Snoeyink. Ray shooting in polygons using geodesic triangulations. In *Proc. 18th Internat. Colloq. Automata Lang. Program.*, volume 510 of *Lecture Notes Comput. Sci.*, pages 661–673. Springer-Verlag, 1991.

[93] D. Z. Chen. On the all-pairs Euclidean short path problem. In *Proc. 6th ACM-SIAM Sympos. Discrete Algorithms*, pages 292–301, 1995.

[94] D. Z. Chen, O. Daescu, and K. S. Klenk. On geometric path query problems. In *Proc. 5th Workshop Algorithms Data Struct.*, volume 1272 of *Lecture Notes Comput. Sci.*, pages 248–257. Springer-Verlag, 1997.

[95] D. Z. Chen, G. Das, and M. Smid. Lower bounds for computing geometric spanners and approximate shortest paths. In *Proc. 8th Canad. Conf. Comput. Geom.*, pages 155–160, 1996.

[96] D. Z. Chen and K. S. Klenk. Rectilinear short path queries among rectangular obstacles. *Inform. Process. Lett.*, 57:313–319, 1996.

[97] D. Z. Chen, K. S. Klenk, and H.-Y. T. Tu. Shortest path queries among weighted obstacles in the rectilinear plane. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 370–379, 1995.

[98] D. Z. Chen, K. S. Klenk, and H.-Y. T. Tu. Shortest path queries among weighted obstacles in the rectilinear plane. Manuscript, Dept. Comput. Sci., Purdue Univ., West Lafayette, IN, 1996.

[99] J. Chen and Y. Han. Shortest paths on a polyhedron. In *Proc. 6th Annu. ACM Sympos. Comput. Geom.*, pages 360–369, 1990.

[100] Y.-B. Chen and D. Ierardi. Time-optimal trajectories of a rod in the plane subject to velocity constraints. *Algorithmica*, 18(2):165–197, June 1997.

[101] S.-W. Cheng. The Steiner tree problem for terminals on the boundary of a rectilinear polygon. In *Proc. of the DIMACS Workshop on Network Design: Connectivity and Facilities Location*, Apr. 1997.

[102] S.-W. Cheng, A. Lim, and C.-T. Wu. Optimal rectilinear Steiner tree for extremal point sets. In *Proc. 4th Annu. Internat. Sympos. Algorithms Comput.*, volume 762 of *Lecture Notes Comput. Sci.*, pages 523–532. Springer-Verlag, 1993.

[103] S.-W. Cheng and C.-K. Tang. A fast algorithm for computing optimal rectilinear Steiner trees for extremal point sets. In *Proc. 6th Annu. Internat. Sympos. Algorithms Comput.*, volume 1004 of *Lecture Notes Comput. Sci.*, pages 322–331. Springer-Verlag, 1995.

[104] D. Cheriton and R. E. Tarjan. Finding minimum spanning trees. *SIAM J. Comput.*, 5:724–742, 1976.

[105] L. P. Chew. Planning the shortest path for a disc in $O(n^2 \log n)$ time. In *Proc. 1st Annu. ACM Sympos. Comput. Geom.*, pages 214–220, 1985.

[106] L. P. Chew. There are planar graphs almost as good as the complete graph. *J. Comput. Syst. Sci.*, 39:205–219, 1989.

[107] Y.-J. Chiang and J. S. B. Mitchell. Two-point Euclidean shortest path queries in the plane. In *Abstracts 14th European Workshop Comput. Geom.*, pages 55–57, 1998.

[108] Y.-J. Chiang, F. P. Preparata, and R. Tamassia. A unified approach to dynamic point location, ray shooting, and shortest paths in planar maps. In *Proc. 4th ACM-SIAM Sympos. Discrete Algorithms*, pages 44–53, 1993.

[109] Y.-J. Chiang, F. P. Preparata, and R. Tamassia. A unified approach to dynamic point location, ray shooting, and shortest paths in planar maps. *SIAM J. Comput.*, 25:207–233, 1996.

[110] W. Chin and S. Ntafos. Optimum watchman routes. *Inform. Process. Lett.*, 28:39–44, 1988.

[111] W.-P. Chin and S. Ntafos. Optimum zookeeper routes. *Congr. Numer.*, 58:257–266, 1987. Combinatorics, Graph Theory and Computing. Proc. 20th South-East Conf., Boca Raton.

[112] W.-P. Chin and S. Ntafos. Watchman routes in simple polygons. *Discrete Comput. Geom.*, 6(1):9–31, 1991.

[113] J. Choi, J. Sellen, and C.-K. Yap. Precision-sensitive Euclidean shortest path in 3-space. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 350–359, 1995.

[114] J. Choi, J. Sellen, and C. K. Yap. Approximate Euclidean shortest paths in 3-space. *Internat. J. Comput. Geom. Appl.*, 7(4):271–295, Aug. 1997.

[115] J. Choi and C.-K. Yap. Rectilinear geodesics in 3-space. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 380–389, 1995.

[116] J. Choi and C.-K. Yap. Monotonicity of rectilinear geodesics in d-space. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.*, pages 339–348, 1996.

[117] J. S. Choi. *Geodesic problems in high dimensions.* PhD thesis, Courant Institute, New York University, New York, June 1995.

[118] K. L. Clarkson. Fast expected time and approximation algorithms for geometric minimum spanning trees. In *Proc. 16th Annu. ACM Sympos. Theory Comput.*, pages 342–348, 1984.

[119] K. L. Clarkson. Approximation algorithms for shortest path motion planning. In *Proc. 19th Annu. ACM Sympos. Theory Comput.*, pages 56–65, 1987.

[120] K. L. Clarkson. An algorithm for geometric minimum spanning trees requiring nearly linear expected time. *Algorithmica*, 4:461–469, 1989.

[121] K. L. Clarkson, S. Kapoor, and P. M. Vaidya. Rectilinear shortest paths through polygonal obstacles in $O(n(\log n)^2)$ time. In *Proc. 3rd Annu. ACM Sympos. Comput. Geom.*, pages 251–257, 1987.

[122] R. Cole and A. Siegel. River routing every which way, but loose. In *Proc. 25th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 65–73, 1984.

[123] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press, Cambridge, MA, 1990.

[124] J. Culberson and G. J. E. Rawlins. Turtlegons: generating simple polygons from sequences of angles. In *Proc. 1st Annu. ACM Sympos. Comput. Geom.*, pages 305–310, 1985.

[125] J. Czyzowicz, P. Egyed, H. Everett, D. Rappaport, T. Shermer, D. Souvaine, G. Toussaint, and J. Urrutia. The aquarium keeper's problem. In *Proc. 2nd ACM-SIAM Sympos. Discrete Algorithms*, pages 459–464, Jan. 1991.

[126] G. Das, S. Kapoor, and M. Smid. On the complexity of approximating Euclidean traveling salesman tours and minimum spanning trees. *Algorithmica*, 19:447–460, 1997.

[127] A. Datta, C. A. Hipke, and S. Schuierer. Competitive searching in polygons – Beyond generalised streets. In *Proc. 6th Annu. Internat. Sympos. Algorithms Comput.*, volume 1004 of *Lecture Notes Comput. Sci.*, pages 32–41. Springer-Verlag, 1995.

[128] A. Datta and C. Icking. Competitive searching in a generalized street. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 175–182, 1994.

[129] A. Datta and K. Krithivasan. Path planning with local information. In *Proc. Conf. Found. Softw. Tech. Theoret. Comput. Sci.*, volume 338 of *Lecture Notes Comput. Sci.*, pages 108–121, New Delhi, India, Dec. 1988. Springer-Verlag.

[130] M. de Berg. On rectilinear link distance. *Comput. Geom. Theory Appl.*, 1(1):13–34, July 1991.

[131] M. de Berg and M. van Kreveld. Trekking in the alps without freezing or getting tired. *Algorithmica*, 18:306–323, 1997.

[132] M. de Berg, M. van Kreveld, and B. J. Nilsson. Shortest path queries in rectilinear worlds of higher dimension. In *Proc. 7th Annu. ACM Sympos. Comput. Geom.*, pages 51–60, 1991.

[133] M. de Berg, M. van Kreveld, B. J. Nilsson, and M. H. Overmars. Shortest path queries in rectilinear worlds. *Internat. J. Comput. Geom. Appl.*, 2(3):287–309, 1992.

[134] L. De Floriani and E. Puppo. Applications in geographic information systems. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, page ?? Elsevier Science Publishers B.V. North-Holland, Amsterdam, 1998.

[135] X. Deng, T. Kameda, and C. Papadimitriou. How to learn an unknown environment. In *Proc. 32nd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 298–303, 1991.

[136] X. Deng, T. Kameda, and C. H. Papadimitriou. How to learn an unknown environment I: the rectilinear case. Technical Report CS-93-04, Department of Computer Science, York University, Canada, 1993.

[137] G. Desaulniers. On shortest paths for a car-like robot maneuvering around obstacles. Les Cahiers du GERAD G-94-35, Ecole des Hautes Etudes Commerciales, Montreal, Canada, 1994.

[138] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[139] B. Donald and P. Xavier. Provably good approximation algorithms for optimal kinodynamic planning for cartesian robots and open chain manipulators. *Algorithmica*, 14(6):480–530, 1995.

[140] B. R. Donald, P. Xavier, J. Canny, and J. Reif. Kinodynamic motion planning. *J. ACM*, 40(5):1048–1066, Nov. 1993.

[141] J. R. Driscoll, H. N. Gabow, R. Shrairaman, and R. E. Tarjan. Relaxed heaps: An alternative to Fibonacci heaps with applications to parallel computation. *Commun. ACM*, 31:1343–1354, 1988.

[142] D.-Z. Du and F. K. Hwang. A proof of Gilbert-Pollak's conjecture on the Steiner ratio. *Algorithmica*, 7:121–135, 1992.

[143] D.-Z. Du and F. K. Hwang. The state of art on Steiner ratio problems. In D.-Z. Du and F. K. Hwang, editors, *Computing in Euclidean Geometry*, volume 1 of *Lecture Notes Series on Computing*, pages 163–191. World Scientific, Singapore, 1992.

[144] L. E. Dubins. On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents. *Amer. J. Math.*, 79:497–516, 1957.

[145] P. Eades, X. Lin, and N. C. Wormald. Performance guarantees for motion planning with temporal uncertainty. *Australian Computer Journal*, 25(1):21–28, 1993.

[146] A. Efrat and S. Har-Peled. Fly cheaply: On the minimum fuel-consumption problem. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, page to appear, 1998.

[147] H. ElGindy and M. T. Goodrich. Parallel algorithms for shortest path problems in polygons. *Visual Comput.*, 3:371–378, 1988.

[148] H. ElGindy and P. Mitra. Orthogonal shortest route queries among axis parallel rectangular obstacles. *Internat. J. Comput. Geom. Appl.*, 4:3–24, 1994.

[149] D. Eppstein. Faster geometric k-point MST approximation. *Comput. Geom. Theory Appl.*, 8(5):231–240, Oct. 1997.

[150] D. Eppstein. Spanning trees and spanners. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, page ?? Elsevier Science Publishers B.V. North-Holland, Amsterdam, 1998.

[151] D. Eppstein and D. Hart. Shortest paths in an arrangement with k line orientations. Manuscript, University of California, Irvine, 1997.

[152] S. P. Fekete. *Geometry and the Travelling Salesman Problem.* Ph.D. thesis, Department of Combinatorics and Optimization, University of Waterloo, Waterloo, ON, 1992.

[153] S. P. Fekete. Area optimization of simple polygons. Technical Report 97-256, Mathematisches Institut, Universität zu Köln, 1997.

[154] S. P. Fekete. Simplicity and hardness of the maximum traveling salesman problem under geometric distances. Manuscript (submitted), Mathematisches Institut, Universität zu Köln, 1998.

[155] S. P. Fekete and W. R. Pulleyblank. Area optimization of simple polygons. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 173–182, 1993.

[156] S. P. Fekete and G. J. Woeginger. Angle-restricted tours in the plane. *Comput. Geom. Theory Appl.*, 8(4):195–218, 1997.

[157] A. Ferreira and J. G. Peters. Finding smallest paths in rectilinear polygons on a hypercube multiprocessor. In *Proc. 3rd Canad. Conf. Comput. Geom.*, pages 162–165, Aug. 1991.

[158] M. Fischetti, H. W. Hamacher, K. Jørnsten, and F. Maffioli. Weighted k-cardinality trees: Complexity and polyhedral structure. *Networks*, 24:11–21, 1994.

[159] S. Fortune and G. Wilfong. Planning constrained motion. *Annals of Math. and AI*, 3:21–82, 1991.

[160] M. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization problems. *J. ACM*, 34:596–615, 1987.

[161] H. N. Gabow, J. L. Bentley, and R. E. Tarjan. Scaling and related techniques for geometry problems. In *Proc. 16th Annu. ACM Sympos. Theory Comput.*, pages 135–143, 1984.

[162] S. Gao, M. Jerrum, M. Kaufmann, K. Mehlhorn, W. Rülling, and C. Storb. On continuous homotopic one layer routing. In *Proc. 4th Annu. ACM Sympos. Comput. Geom.*, pages 392–402, 1988.

[163] M. R. Garey, R. L. Graham, and D. S. Johnson. The complexity of computing Steiner minimal trees. *SIAM J. Appl. Math.*, 32:835–859, 1977.

[164] 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.

[165] N. Garg. A 3-approximation for the minimum tree spanning k vertices. In *37th Annual Symposium on Foundations of Computer Science*, pages 302–309, Burlington, Vermont, October 14-16 1996.

[166] N. Garg and D. S. Hochbaum. An $O(\log k)$ approximation algorithm for the k minimum spanning tree problem in the plane. In *Proc. 26th Annu. ACM Sympos. Theory Comput.*, pages 432–438, 1994.

[167] N. Garg, G. Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the group Steiner problem. In *Proc. 9th ACM-SIAM Sympos. Discrete Algorithms*, pages 253–259, 1998.

[168] L. Gewali, A. Meng, J. S. B. Mitchell, and S. Ntafos. Path planning in $0/1/\infty$ weighted regions with applications. *ORSA J. Comput.*, 2(3):253–272, Summer 1990.

[169] L. Gewali, S. Ntafos, and I. G. Tollis. Path planning in the presence of vertical obstacles. Technical report, Computer Science, University of Texas at Dallas, 1989.

[170] L. P. Gewali and S. Ntafos. Watchman routes in the presence of a pair of convex polygons. In *Proc. 7th Canad. Conf. Comput. Geom.*, pages 127–132, 1995.

[171] S. Ghosh. Visibility. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, page ?? Elsevier Science Publishers B.V. North-Holland, Amsterdam, 1998.

[172] S. K. Ghosh. On recognizing and characterizing visibility graphs of simple polygons. *Discrete Comput. Geom.*, 17:143–162, 1997.

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

[174] S. K. Ghosh and S. Saluja. Optimal on-line algorithms for walking with minimum number of turns in unknown streets. *Comput. Geom. Theory Appl.*, 8(5):241–266, Oct. 1997.

[175] E. N. Gilbert and H. O. Pollak. Steiner minimal trees. *SIAM J. Appl. Math.*, 16(1):1–29, 1968.

[176] M. Goemans and D. Williamson. General approximation technique for constrained forest problems. In *Proc. 3rd ACM-SIAM Sympos. Discrete Algorithms (SODA '92)*, pages 307–315, 1992.

[177] M. X. Goemans and J. M. Kleinberg. An improved approximation ratio for the minimum latency problem. In *Proc. 7th ACM-SIAM Sympos. Discrete Algorithms (SODA '96)*, pages 152–158, 1996.

[178] J. E. Goodman and J. O'Rourke, editors. *Handbook of Discrete and Computational Geometry.* CRC Press LLC, Boca Raton, FL, 1997.

[179] M. T. Goodrich. Parallel algorithms in geometry. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 36, pages 669–682. CRC Press LLC, Boca Raton, FL, 1997.

[180] M. T. Goodrich. Geometric data structures. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, page ?? Elsevier Science Publishers B.V. North-Holland, Amsterdam, 1998.

[181] M. T. Goodrich, S. Shauck, and S. Guha. Parallel methods for visibility and shortest path problems in simple polygons. *Algorithmica*, 8:461–486, 1992.

[182] M. T. Goodrich, S. Shauck, and S. Guha. Addendum to "parallel methods for visibility and shortest path problems in simple polygons". *Algorithmica*, 9:515–516, 1993.

[183] M. T. Goodrich and R. Tamassia. Dynamic ray shooting and shortest paths via balanced geodesic triangulations. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 318–327, 1993.

[184] J. Gudmundsson and C. Levcopoulos. A fast approximation algorithm for TSP with neighborhoods. Technical Report LU-CS-TR:97-195, Dept. of Comp. Sci., Lund University, 1997.

[185] L. J. Guibas and J. Hershberger. Optimal shortest path queries in a simple polygon. *J. Comput. Syst. Sci.*, 39:126–152, 1989.

[186] 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.

[187] L. J. Guibas, J. E. Hershberger, J. S. B. Mitchell, and J. S. Snoeyink. Approximating polygons and subdivisions with minimum link paths. *Internat. J. Comput. Geom. Appl.*, 3(4):383–415, Dec. 1993.

[188] M. K. Guyder. Automating the optimization of $2\frac{1}{2}$-axis milling. In F. Kimura and A. Rolstadås, editors, *Proc. Computer Applications in Production and Engineering*, page (supplement). North-Holland, Oct. 1989.

[189] M. Hammar and B. J. Nilsson. Concerning the time bounds of existing shortest watchman route algorithms. In *Proc. 11th International Symposium on Fundamentals of Computation Theory*, volume 1279 of *Lecture Notes Comput. Sci.*, pages 210–221, Krakow, Poland, 1-3 September 1997. Springer-Verlag.

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

[191] 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.

[192] S. Har-Peled. Approximate shortest paths and geodesic diameters on convex polytopes in three dimensions. *Discrete Comput. Geom.*, ??(??):to appear, ??

[193] S. Har-Peled. Approximate shortest paths and geodesic diameters on convex polytopes in three dimensions. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 359–365, 1997.

[194] S. Har-Peled. Constructing approximate shortest path maps in three dimensions. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, page to appear, 1998.

[195] R. Hassin and S. Rubinstein. An approximation algorithm for the maximum traveling salesman problem. Manuscript, submitted, Tel Aviv University, Tel Aviv, Israel, 1997.

[196] M. Held. *On the Computational Geometry of Pocket Machining*, volume 500 of *Lecture Notes Comput. Sci.* Springer-Verlag, June 1991.

[197] M. Henig. The shortest path problem with two objective functions. *European J. Oper. Res.*, 25:281–291, 1985.

[198] M. R. Henzinger, P. Klein, and S. Rao. Faster shortest-path algorithms for planar graphs. *J. Comput. Syst. Sci.*, 55:3–23, 1997.

[199] J. Hershberger. Finding the visibility graph of a simple polygon in time proportional to its size. *Algorithmica*, 4:141–155, 1989.

[200] J. Hershberger. A new data structure for shortest path queries in a simple polygon. *Inform. Process. Lett.*, 38:231–235, 1991.

[201] J. Hershberger. Optimal parallel algorithms for triangulated simple polygons. In *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, pages 33–42, 1992.

[202] J. Hershberger and L. J. Guibas. An $O(n^2)$ shortest path algorithm for a non-rotating convex body. *J. Algorithms*, 9:18–46, 1988.

[203] J. Hershberger and J. Snoeyink. Computing minimum length paths of a given homotopy class. *Comput. Geom. Theory Appl.*, 4:63–98, 1994.

[204] J. Hershberger and J. Snoeyink. An efficient solution to the zookeeper's problem. In *Proc. 6th Canad. Conf. Comput. Geom.*, pages 104–109, 1994.

[205] J. Hershberger and S. Suri. Efficient computation of Euclidean shortest paths in the plane. In *Proc. 34th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 508–517, 1993.

[206] J. Hershberger and S. Suri. An optimal algorithm for Euclidean shortest paths in the plane. Manuscript, Washington University, 1995.

[207] J. Hershberger and S. Suri. A pedestrian approach to ray shooting: Shoot a ray, take a walk. *J. Algorithms*, 18:403–431, 1995.

[208] J. Hershberger and S. Suri. Practical methods for approximating shortest paths on a convex polytope in $\Re^3$. In *Proc. 6th ACM-SIAM Sympos. Discrete Algorithms*, pages 447–456, 1995.

[209] J. Hershberger and S. Suri. Matrix searching with the shortest path metric. *SIAM J. Comput.*, 26(6):1612–1634, Dec. 1997.

[210] D. Hochbaum, editor. *Approximation Problems for NP-Complete Problems.* PWS Publishing Company, Boston, MA, 1997.

[211] F. Hoffmann, C. Icking, R. Klein, and K. Kriegel. A competitive strategy for learning a polygon. In *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, pages 166–174, 1997.

[212] F. Hoffmann, C. Icking, R. Klein, and K. Kriegel. The polygon exploration problem: A new strategy and a new analysis technique. In *Proc. 1998 Workshop Algorithmic Found. Robot.*, page to appear, 1998.

[213] C. Hurkens. Nasty TSP instances for farthest insertion. In *Proc. 2nd IPCO Conference Integer Programming and Combinatorial Optimization*, page ??, 1992.

[214] F. K. Hwang. On Steiner minimal trees with rectilinear distance. *SIAM J. Appl. Math.*, 30(1):104–114, 1976.

[215] C. Icking. *Motion and Visibility in Simple Polygons.* PhD thesis, FernUniversität Hagen, 1994.

[216] C. Icking and R. Klein. Searching for the kernel of a polygon: A competitive strategy. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 258–266, 1995.

[217] C. Icking, G. Rote, E. Welzl, and C. Yap. Shortest paths for line segments. *Algorithmica*, 10:182–200, 1993.

[218] E. Ihler. Bounds on the quality of approximate solutions to the group Steiner problem. In *Graph-Theoret. Concepts Comput. Sci.: Proc. Internat. Workshop*, volume 484 of *Lecture Notes Comput. Sci.*, pages 109–118, Berlin, Germany, 1991. Springer-Verlag.

[219] E. Ihler. The complexity of approximating the class Steiner tree problem. In *Graph Theoretic Concepts in Computer Science: Proc. Internat. Workshop WG '91*, volume 570 of *Lecture Notes Comput. Sci.*, pages 85–96, Berlin, Germany, 1992. Springer-Verlag.

[220] E. Ihler. The rectilinear Steiner tree problem for intervals on two parallel lines. *Math. Program. Ser. B*, 63:281–296, 1994.

[221] E. Ihler, G. Reich, and P. Widmayer. On shortest networks for classes of points in the plane. In *Proc. Computational Geometry: Methods, Algorithms and Applications*, volume 553 of *Lecture Notes Comput. Sci.*, pages 103–111. Springer-Verlag, 1991.

[222] 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.

[223] M. Imase and B. M. Waxman. Dynamic Steiner tree problem. *SIAM J. Discrete Math.*, 4:369–384, 1991.

[224] K. Iwano, P. Raghavan, and H. Tamaki. The traveling cameraman problem, with applications to automatic optical inspection. In *Proc. 5th Annu. Internat. Sympos. Algorithms Comput.*, volume 834 of *Lecture Notes Comput. Sci.*, pages 29–37. Springer-Verlag, 1994.

[225] S. S. Iyengar and A. Elfes, editors. *Autonomous Mobile Robots: Perception, Mapping, and Navigation.* IEEE Computer Society Press, Los Alamitos, CA, 1991.

[226] S. S. Iyengar, C. C. Jorgensen, S. V. N. Rao, and C. R. Weisbin. Robot navigation algorithms using learned spatial graphs. *Robotica*, 4:93–100, 1986.

[227] P. Jacobs and J. Canny. Planning smooth paths for mobile robots. In *Proc. IEEE Internat. Conf. Robot. Autom.*, pages 2–7, 1989.

[228] P. Jacobs and J. Canny. Planning smooth paths for mobile robots. In Z. Li and J. F. Canny, editors, *Nonholonomic Motion Planning*, pages 271–342. Kluwer Academic Pubishers, Norwell, MA, 1992.

[229] P. Johansson. On a weighted distance model for injection moulding. Linköping Studies in Science and Technology, Thesis No. 604 LiU-TEK-LIC-1997:05, Division of Applied Mathematics, Linköping University, Linköping, Sweden, Feb. 1997.

[230] M. Jünger, G. Reinelt, and G. Rinaldi. The traveling salesman problem. In M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors, *Network Models*, Handbook of Operations Research/Management Science, pages 225–330. Elsevier Science, Amsterdam, 1995.

[231] B. Kalyanasundaram and K. Pruhs. A competitive analysis of algorithms for searching unknown scenes. *Comput. Geom. Theory Appl.*, 3:139–155, 1993.

[232] B. Kalyanasundaram and K. Pruhs. Constructing competitive tours from local information. *Theoret. Comput. Sci.*, 130:125–138, 1994.

[233] B. Kalyanasundaram and K. Pruhs. Online network optimization problems. In A. Fiat and G. Woeginger, editors, *Competitive Analysis of Algorithms.* Springer-Verlag, 1998.

[234] M.-Y. Kao, J. H. Reif, and S. R. Tate. Searching in an unknown environment: An optimal randomized algorithm for the cow-path problem. In *Proc. 4th ACM-SIAM Sympos. Discrete Algorithms*, pages 441–447, 1993.

[235] S. Kapoor. Approximate geographic neighbor tree with applications. Manuscript, IIT, New Delhi, 1997.

[236] S. Kapoor and S. N. Maheshwari. Efficient algorithms for Euclidean shortest path and visibility problems with polygonal obstacles. In *Proc. 4th Annu. ACM Sympos. Comput. Geom.*, pages 172–182, 1988.

[237] S. Kapoor, S. N. Maheshwari, and J. S. B. Mitchell. An efficient algorithm for Euclidean shortest paths among polygonal obstacles in the plane. *Discrete Comput. Geom.*, 18:377–383, 1997.

[238] M. Kindl, M. Shing, and N. Rowe. A stochastic approach to the weighted-region problem, I: The design of the path annealing algorithm. Technical report, Computer Science, U.S. Naval Postgraduate School, Monterey, CA, 1991.

[239] M. Kindl, M. Shing, and N. Rowe. A stochastic approach to the weighted-region problem, II: Performance enhancement techniques and experimental results. Technical report, Computer Science, U.S. Naval Postgraduate School, Monterey, CA, 1991.

[240] P. N. Klein and R. Ravi. A nearly best-possible approximation algorithm for node-weighted Steiner tree. *J. Algorithms*, 19:104–115, 1995.

[241] R. Klein. Walking an unknown street with bounded detour. *Comput. Geom. Theory Appl.*, 1:325–351, 1992.

[242] J. M. Kleinberg. On-line search in a simple polygon. In *Proc. 5th ACM-SIAM Sympos. Discrete Algorithms*, pages 8–15, 1994.

[243] S. Kosaraju, J. Park, and C. Stein. Long tours and short superstrings. In *Proc. 35th Annu. IEEE Sympos. Found. Comput. Sci. (FOCS 94)*, 1994.

[244] D. Krznaric, C. Levcopoulos, and B. J. Nilsson. Minimum spanning trees in d dimensions. In *Proc. 5th Annu. European Sympos. Algorithms*, volume 1284 of *Lecture Notes Comput. Sci.*, pages 341–349. Springer-Verlag, 1997.

[245] P. Kumar and C. Veni Madhavan. Shortest watchman tours in weak visibility polygons. In *Proc. 5th Canad. Conf. Comput. Geom.*, pages 91–96, 1993.

[246] M. Lanthier, A. Maheshwari, and J.-R. Sack. Approximating weighted shortest paths on polyhedral surfaces. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 274–283, 1997.

[247] J.-C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, 1991.

[248] J. P. Laumond. Feasible trajectories for mobile robots with kinematic and environment constraints. In L. O. Hertzberger and F. C. A. Groen, editors, *Conference on Intelligent Autonomous Systems (Amsterdam, the Netherlands, December 8–11, 1986)*, pages 346–354. Elsevier Science Publishers, Dec. 1986.

[249] J.-P. Laumond, P. Jacobs, M. Taix, and R. M. Murray. A motion planner for nonholonomic mobile robots. *IEEE Trans. Robot. Autom.*, 10(5):577–593, 1994.

[250] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, editors. *The Traveling Salesman Problem.* Wiley, New York, NY, 1985.

[251] D. T. Lee. Proximity and reachability in the plane. Report R-831, Dept. Elect. Engrg., Univ. Illinois, Urbana, IL, 1978.

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

[253] D. T. Lee, C. D. Yang, and T. H. Chen. Shortest rectilinear paths among weighted obstacles. *Internat. J. Comput. Geom. Appl.*, 1(2):109–124, 1991.

[254] D. T. Lee, C. D. Yang, and C. K. Wong. Rectilinear paths among rectilinear obstacles. *Discrete Appl. Math.*, 70:185–215, 1996.

[255] J.-H. Lee, C.-S. Shin, J.-H. Kim, S. Shin, and K.-Y. Chwa. New competitive strategies for searching in unknown star-shaped polygons. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 427–429, 1997.

[256] C. E. Leiserson and F. M. Maley. Algorithms for routing and testing routability of planar VLSI layouts. In *Proc. 17th Annu. ACM Sympos. Theory Comput.*, pages 69–78, 1985.

[257] C. Levcopoulos and D. Krznaric. Quasi-greedy triangulations approximating the minimum weight triangulation. In *Proc. 7th ACM-SIAM Sympos. Discrete Algorithms*, pages 392–401, 1996.

[258] Z. Li and J. F. Canny, editors. *Nonholonomic Motion Planning.* Kluwer Academic Pubishers, Norwell, MA, 1992.

[259] A. Lingas, A. Maheshwari, and J.-R. Sack. Parallel algorithms for rectilinear link distance problems. *Algorithmica*, 14:261–289, 1995.

[260] A. López-Ortiz and S. Schuierer. Going home through an unknown street. In *Proc. 4th Workshop Algorithms Data Struct.*, volume 955 of *Lecture Notes Comput. Sci.*, pages 135–146. Springer-Verlag, 1995.

[261] A. López-Ortiz and S. Schuierer. Simple, efficient and robust strategies to traverse streets. In *Proc. 7th Canad. Conf. Comput. Geom.*, pages 217–222, 1995.

[262] A. López-Ortiz and S. Schuierer. Generalized streets revisited. In J. Diaz and M. Serna, editors, *Proc. 4th Annu. European Sympos. Algorithms*, volume 1136 of *Lecture Notes Comput. Sci.*, pages 546–558. Springer-Verlag, 1996.

[263] A. López-Ortiz and S. Schuierer. Walking streets faster. In *Proc. 5th Scand. Workshop Algorithm Theory*, volume 1097 of *Lecture Notes Comput. Sci.*, pages 345–356. Springer-Verlag, 1996.

[264] A. López-Ortiz and S. Schuierer. Position-independent near optimal searching and on-line recognition in star polygons. In F. Dehne, A. Rau-Chaplin, J.-R. Sack, and R. Tamassia, editors, *Algorithms and Data Structures, 5th International Workshop*, volume 1272 of *Lecture Notes Comput. Sci.*, pages 284–296, Halifax, Nova Scotia, Canada, Aug. 1997. Springer-Verlag.

[265] V. J. Lumelsky. Algorithmic and complexity issues of robot motion in an uncertain environment. *J. Complexity*, 3:146–182, 1987.

[266] V. J. Lumelsky. Algorithmic issues of sensor-based robot motion planning. In *Proc. 26th IEEE Conf. Decision Control*, pages 1796–1801, 1987.

[267] V. J. Lumelsky. A comparative study on the path length performance of maze-searching and robot motion planning algorithms. *IEEE Trans. Robot. Autom.*, 7(1):57–66, 1991.

[268] V. J. Lumelsky and A. A. Stepanov. Dynamic path planning for a mobile automaton with limited information on the environment. *IEEE Trans. Autom. Control*, AC-31:1058–1063, 1986.

[269] V. J. Lumelsky and A. A. Stepanov. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2:403–430, 1987.

[270] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *J. ACM*, 41:960–982, 1994.

[271] A. Maheshwari and J.-R. Sack. Link distance problems. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, page ?? Elsevier Science Publishers B.V. North-Holland, Amsterdam, 1998.

[272] C. Mata and J. S. B. Mitchell. Approximation algorithms for geometric tour and network design problems. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 360–369, 1995.

[273] C. Mata and J. S. B. Mitchell. A new algorithm for computing shortest paths in weighted planar subdivisions. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 264–273, 1997.

[274] K. M. McDonald and J. G. Peters. Smallest paths in simple rectilinear polygons. *IEEE Trans. on Computer Aided Design*, 11:864–875, 1992.

[275] A. Mei and Y. Igarashi. An efficient strategy for robot navigation in unknown environment. *Inform. Process. Lett.*, 52:51–56, 1994.

[276] A. Mei and Y. Igarashi. A robot navigation strategy in unknown environment and its efficiency. *IEICE Trans. Fundamentals Electronics, Comm. and Comput. Sci.*, E77-A(7):1157–1162, July 1994.

[277] B. Mirtich and J. Canny. Using skeletons for nonholonomic path planning among obstacles. In *Proc. 9th IEEE Internat. Conf. Robot. Autom.*, pages 2533–2540, 1992.

[278] J. S. B. Mitchell. *Planning shortest paths.* Ph.D. thesis, Stanford Univ., Stanford, CA, 1986.

[279] J. S. B. Mitchell. An optimal algorithm for shortest rectilinear paths among obstacles. In *Abstracts 1st Canad. Conf. Comput. Geom.*, page 22, 1989.

[280] J. S. B. Mitchell. On maximum flows in polyhedral domains. *J. Comput. Syst. Sci.*, 40:88–123, 1990.

[281] J. S. B. Mitchell. An algorithmic approach to some problems in terrain navigation. In S. S. Iyengar and A. Elfes, editors, *Autonomous Mobile Robots: Perception, Mapping, and Navigation*, pages 408–427. IEEE Computer Society Press, Los Alamitos, CA, 1991.

[282] J. S. B. Mitchell. A new algorithm for shortest paths among obstacles in the plane. *Ann. Math. Artif. Intell.*, 3:83–106, 1991.

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

[284] J. S. B. Mitchell. Shortest paths among obstacles in the plane. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 308–317, 1993.

[285] J. S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple new method for the geometric k-MST problem. In *Proc. 7th ACM-SIAM Sympos. Discrete Algorithms*, pages 402–408, 1996.

[286] J. S. B. Mitchell. Shortest paths among obstacles in the plane. *Internat. J. Comput. Geom. Appl.*, 6:309–332, 1996.

[287] J. S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: Part III – Faster polynomial-time approximation schemes for geometric network optimization. Manuscript, University at Stony Brook, 1997.

[288] J. S. B. Mitchell. Shortest paths and networks. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 24, pages 445–466. CRC Press LLC, Boca Raton, FL, 1997.

[289] J. S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM J. Comput.*, ??:??, 1998.

[290] J. S. B. Mitchell, A. Blum, P. Chalasani, and S. Vempala. A constant-factor approximation algorithm for the geometric k-MST problem in the plane. *SIAM J. Comput.*, ??:To appear, 1998.

[291] J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou. The discrete geodesic problem. *SIAM J. Comput.*, 16:647–668, 1987.

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

[293] J. S. B. Mitchell, D. W. Payton, and D. M. Keirsey. Planning and reasoning for autonomous vehicle control. *Internat. J. Intell. Syst.*, II:129–198, 1987.

[294] J. S. B. Mitchell, C. Piatko, and E. M. Arkin. Computing a shortest k-link path in a polygon. In *Proc. 33rd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 573–582, 1992.

[295] J. S. B. Mitchell and S. Suri. Separation and approximation of polyhedral objects. *Comput. Geom. Theory Appl.*, 5:95–114, 1995.

[296] J. S. B. Mitchell and S. Suri. A survey of computational geometry. In M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors, *Network Models*, Handbook of Operations Research/Management Science, pages 425–479. Elsevier Science, Amsterdam, 1995.

[297] J. S. B. Mitchell and E. L. Wynters. Watchman routes for multiple guards. In *Proc. 3rd Canad. Conf. Comput. Geom.*, pages 126–129, 1991.

[298] P. Mitra and B. Bhattacharya. Efficient approximate shortest-path queries among isothetic rectangular obstacles. In *Proc. 3rd Workshop Algorithms Data Struct.*, volume 709 of *Lecture Notes Comput. Sci.*, pages 518–529. Springer-Verlag, 1993.

[299] C. Monma, M. Paterson, S. Suri, and F. Yao. Computing Euclidean maximum spanning trees. *Algorithmica*, 5:407–419, 1990.

[300] D. M. Mount. The number of shortest paths on the surface of a polyhedron. *SIAM J. Comput.*, 19:593–611, 1990.

[301] B. J. Nilsson. *Guarding Art Galleries — Methods for Mobile Guards.* PhD thesis, Lund University, 1995.

[302] B. J. Nilsson and D. Wood. Optimum watchmen routes in spiral polygons. In *Proc. 2nd Canad. Conf. Comput. Geom.*, pages 269–272, 1990.

[303] S. Ntafos. The robber route problem. *Inform. Process. Lett.*, 34(2):59–63, Mar. 1990.

[304] S. Ntafos. Watchman routes under limited visibility. *Comput. Geom. Theory Appl.*, 1(3):149–170, 1992.

[305] S. Ntafos and L. Gewali. External watchman routes. *Visual Comput.*, 10:474–483, 1994.

[306] C. Ó'Dúnlaing. Motion-planning with inertial constraints. *Algorithmica*, 2:431–475, 1987.

[307] H. L. Ong. Approximate algorithms for the traveling purchaser problem. *Oper. Res. Lett.*, 1:201–205, 1982.

[308] J. O'Rourke. Finding a shortest ladder path: a special case. IMA Preprint Series 353, Inst. Math. Appl., Univ. Minnesota, Minneapolis, MN, 1987.

[309] J. O'Rourke. Uniqueness of orthogonal connect-the-dots. In G. T. Toussaint, editor, *Computational Morphology*, pages 97–104. North-Holland, Amsterdam, Netherlands, 1988.

[310] J. O'Rourke. Visibility. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 25, pages 467–480. CRC Press LLC, Boca Raton, FL, 1997.

[311] J. O'Rourke and C. Schevon. Computing the geodesic diameter of a 3-polytope. In *Proc. 5th Annu. ACM Sympos. Comput. Geom.*, pages 370–379, 1989.

[312] J. O'Rourke and I. Streinu. Vertex-edge pseudo-visibility graphs: Characterization and recognition. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 119–128, 1997.

[313] M. H. Overmars and E. Welzl. New methods for computing visibility graphs. In *Proc. 4th Annu. ACM Sympos. Comput. Geom.*, pages 164–171, 1988.

[314] N. Papadakis and A. Perakis. Minimal time vessel routing in a time-dependent environment. *Transp. Sci.*, 23(4):266–276, 1989.

[315] N. Papadakis and A. Perakis. Deterministic minimal time vessel routing. *Oper. Res.*, 38(3):426–438, 1990.

[316] C. H. Papadimitriou. The Euclidean traveling salesman problem is NP-complete. *Theoret. Comput. Sci.*, 4:237–244, 1977.

[317] C. H. Papadimitriou. An algorithm for shortest-path motion in three dimensions. *Inform. Process. Lett.*, 20:259–263, 1985.

[318] C. H. Papadimitriou and E. B. Silverberg. Optimal piecewise linear motion of an object among obstacles. *Algorithmica*, 2:523–539, 1987.

[319] C. H. Papadimitriou and M. Yannakakis. Shortest paths without a map. *Theoret. Comput. Sci.*, 84(1):127–150, 1991.

[320] E. Papadopoulou and D. T. Lee. Efficient computation of the geodesic Voronoi diagram of points in a simple polygon. In P. G. Spirakis, editor, *Algorithms—ESA '95, Third Annual European Symposium*, volume 979 of *Lecture Notes Comput. Sci.*, pages 238–251, Corfu, Greece, Sept. 1995. Springer-Verlag.

[321] R. Parker and R. Rardin. Guaranteed performance heuristics for the bottleneck travelling salesman problem. *Operations Research Letters*, 2(6):269–272, 1984.

[322] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley, Reading, MA, 1984.

[323] M. Pocchiola and G. Vegter. The visibility complex. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 328–337, 1993.

[324] M. Pocchiola and G. Vegter. Computing the visibility graph via pseudo-triangulations. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 248–257, 1995.

[325] M. Pocchiola and G. Vegter. Topologically sweeping visibility complexes via pseudo-triangulations. *Discrete Comput. Geom.*, 16:419–453, Dec. 1996.

[326] R. Pollack, M. Sharir, and G. Rote. Computing of the geodesic center of a simple polygon. *Discrete Comput. Geom.*, 4:611–626, 1989.

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

[328] R. Raman. Recent results on the single-source shortest paths problem. *SIGACT News*, 28(2):81–87, 1997.

[329] N. S. V. Rao. Algorithmic framework for learned robot navigation in unknown terrain. *IEEE Computer*, 22:37–43, 1989.

[330] N. S. V. Rao, S. S. Iyengar, C. C. Jorgensen, and C. R. Weisbin. Robot navigation in an unexplored terrain. *Journal of Robotic Systems*, 3(4):389–407, 1986.

[331] N. S. V. Rao, S. S. Iyengar, B. J. Oommen, and R. L. Kashyap. On terrain model acquisition by a point robot amidst polyhedral obstacles. *Internat. J. Robot. Autom.*, 4(4):450–455, 1988.

[332] S. B. Rao and W. D. Smith. Improved approximation schemes for traveling salesman tours. In *Proc. 30th Annu. ACM Sympos. Theory Comput.*, page to appear, 1998.

[333] R. Ravi, R. Sundaram, M. V. Marathe, D. J. Rosenkrantz, and S. S. Ravi. Spanning trees short and small. In *Proc. 5th ACM-SIAM Sympos. Discrete Algorithms*, pages 546–555, 1994.

[334] J. A. Reeds and L. A. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2), 1990.

[335] G. Reich and P. Widmayer. Beyond Steiner's problem: A VLSI oriented generalization. In *Proc. 15th Internat. Workshop Graph-Theoret. Concepts Comput. Sci.*, volume 411 of *Lecture Notes Comput. Sci.*, pages 196–210, Heidelberg, NY, 1989. Springer-Verlag.

[336] J. Reif and S. Sen. Parallel computational geometry: An approach using randomization. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, page ?? Elsevier Science Publishers B.V. North-Holland, Amsterdam, 1998.

[337] J. Reif and M. Sharir. Motion planning in the presence of moving obstacles. *J. ACM*, 41(4):764–790, July 1994.

[338] J. Reif and H. Wang. Non-uniform discretization for kinodynamic motion planning and its applications. In J.-P. Laumond and M. Overmars, editors, *Algorithms for Robotic Motion and Manipulation*, pages 97–112, Wellesley, MA, 1997. A.K. Peters. Proc. 1996 Workshop on the Algorithmic Foundations of Robotics, Toulouse, France, July 1996.

[339] J. Reif and H. Wang. The complexity of the two dimensional curvature-constrained shortest-path problem. In *Proc. 1998 Workshop Algorithmic Found. Robot.*, page to appear, 1998.

[340] J. H. Reif and J. A. Storer. A single-exponential upper bound for finding shortest paths in three dimensions. *J. ACM*, 41(5):1013–1019, 1994.

[341] J. H. Reif and S. R. Tate. Approximate kinodynamic planning using L_2-norm dynamic bounds. *Comput. Math. Appl.*, 27(5):29–44, 1994.

[342] J. H. Reif and H. Wang. On-line navigation through weighted regions. Technical report, Dept. Computer Science, Duke University, 1993.

[343] G. Reinelt. Fast heuristics for large geometric traveling salesman problems. *ORSA J. Comput.*, 4:206–217, 1992.

[344] D. B. Reister and F. G. Pin. Time-optimal trajectories for mobile robots with two independently driven wheels. *Internat. J. Robot. Res.*, 13:38–54, 1994.

[345] D. S. Richards and J. S. Salowe. A linear-time algorithm to construct a rectilinear Steiner minimal tree for k-extremal point sets. *Algorithmica*, 7:247–276, 1992.

[346] S. Rivière. Topologically sweeping the visibility complex of polygonal scenes. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages C36–C37, 1995.

[347] H. Rohnert. A new algorithm for shortest paths avoiding convex polygonal obstacles. Report A86/02, Fachber. Inform., Univ. Saarlandes, Saarbrücken, West Germany, 1986.

[348] H. Rohnert. Shortest paths in the plane with convex polygonal obstacles. *Inform. Process. Lett.*, 23:71–76, 1986.

[349] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis. An analysis of several heuristics for the traveling salesman problem. *SIAM J. Comput.*, 6:563–581, 1977.

[350] N. C. Rowe. Obtaining optimal mobile-robot paths with non-smooth anisotropic cost functions using qualitative-state reasoning. *Internat. J. Robot. Res.*, 16(3):375–399, June 1997.

[351] J. S. Salowe. Constructing multidimensional spanner graphs. *Internat. J. Comput. Geom. Appl.*, 1(2):99–107, 1991.

[352] C. Schevon and J. O'Rourke. The number of maximal edge sequences on a convex polytope. In *Proc. 26th Allerton Conf. Commun. Control Comput.*, pages 49–57, Oct. 1988.

[353] S. Schirra. Robustness issues. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, page ?? Elsevier Science Publishers B.V. North-Holland, Amsterdam, 1998.

[354] S. Schuierer. An optimal data structure for shortest rectilinear path queries in a simple rectilinear polygon. *Internat. J. Comput. Geom. Appl.*, 6:205–226, 1996.

[355] S. Schuierer. Lower bounds in on-line geometric searching. In *11th International Symposium on Fundamentals of Computation Theory*, volume 1279 of *Lecture Notes Comput. Sci.*, pages 429–440. Springer-Verlag, Krakow, Poland, 1-3 September 1997.

[356] J. Sellen. Direction weighted shortest path planning. In *Proc. IEEE Internat. Conf. Robot. Autom.*, pages 1970–1975, 1995.

[357] J. Sellen. Planning paths of minimal curvature. In *Proc. IEEE Internat. Conf. Robot. Autom.*, pages 1976–1982, 1995.

[358] J. Sellen. Approximation and decision algorithms for curvature-constrained path planning: A state-space approach. In *Proc. 1998 Workshop Algorithmic Found. Robot.*, page to appear, 1998.

[359] I. Semrau. Analyse und experimentelle Untersuchung von Strategien zum Finden eines Ziels in Straßenpolygonen. Master's thesis (Diplomarbeit), Fernuniversität Hagen, July 1996.

[360] M. Sharir. On shortest paths amidst convex polyhedra. *SIAM J. Comput.*, 16:561–572, 1987.

[361] M. Sharir. A note on the Papadimitriou-Silverberg algorithm for planning optimal piecewise linear motion of a ladder. *Inform. Process. Lett.*, 32:187–190, 1989.

[362] M. Sharir and A. Schorr. On shortest paths in polyhedral spaces. *SIAM J. Comput.*, 15:193–215, 1986.

[363] P. Slavík. The errand scheduling problem. Technical report 97-2, Department of Computer Science, SUNY, Buffalo NY, March 14 1997.

[364] P. Slavík. On the approximation of the generalized traveling salesman problem. Manuscript, submitted, Department of Computer Science, SUNY, Buffalo NY, 1998.

[365] J. M. Smith and P. Winter. Computational geometry and topological network design. In D.-Z. Du and F. K. Hwang, editors, *Computing in Euclidean Geometry*, volume 1 of *Lecture Notes Series on Computing*, pages 287–385. World Scientific, Singapore, 1992.

[366] S. Suri. Computing geodesic furthest neighbors in simple polygons. *J. Comput. Syst. Sci.*, 39:220–235, 1989.

[367] H. J. Sussman. Shortest 3-dimensional paths with a prescribed curvature bound. In *Proc. 34th IEEE Conf. Decision Control*, pages 3306–3311, 1995.

[368] H. J. Sussmann and G. Tang. Shortest paths for the Reeds-Shepp car: A worked out example of the use of geometric techniques in nonlinear optimal control. Research Report SYCON-91-10, Rutgers University, New Brunswick, NJ, 1991.

[369] P. Švestka and M. H. Overmars. Motion planning for car-like robots using a probabilistic learning approach. *Internat. J. Robot. Res.*, 16(2):119–143, Apr. 1997.

[370] X. Tan and T. Hirata. Constructing shortest watchman routes by divide-and-conquer. In *Proc. 4th Annu. Internat. Sympos. Algorithms Comput.*, volume 762 of *Lecture Notes Comput. Sci.*, pages 68–77. Springer-Verlag, 1993.

[371] X. Tan and T. Hirata. Shortest safari routes in simple polygon. In *Proc. 5th Annu. Internat. Sympos. Algorithms Comput.*, volume 834 of *Lecture Notes Comput. Sci.*, pages 523–531, Beijing, 1994. Springer-Verlag.

[372] X. Tan, T. Hirata, and Y. Inagaki. Corrigendum to 'An incremental algorithm for constructing shortest watchman routes'. Manuscript (submitted to internat. j. comput. geom. appl.), Tokai University, Japan, 1998.

[373] X. H. Tan, T. Hirata, and Y. Inagaki. An incremental algorithm for constructing shortest watchman routes. *Internat. J. Comput. Geom. Appl.*, 3(4):351–365, 1993.

[374] M. Thorup. Undirected single source shortest path in linear time. In *Proc. 38th Annu. IEEE Sympos. Found. Comput. Sci.*, page To appear, 1997.

[375] L. Trevisan. When Hamming meets Euclid: The approximability of geometric TSP and MST. In *Proc. 29th Annu. ACM Sympos. Theory Comput.*, pages 21–29, 1997.

[376] C. Umans and W. Lenhart. Hamiltonian cycles in solid grid graphs. In *Proc. 38th Annu. IEEE Sympos. Found. Comput. Sci.*, page To appear, 1997.

[377] P. M. Vaidya. A fast approximation for minimum spanning trees in k-dimensional space. In *Proc. 25th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 403–407, 1984.

[378] P. M. Vaidya. Minimum spanning trees in k-dimensional space. *SIAM J. Comput.*, 17:572–582, 1988.

[379] P. M. Vaidya. Approximate minimum weight matching on points in k-dimensional space. *Algorithmica*, 4:569–583, 1989.

[380] P. M. Vaidya. Geometry helps in matching. *SIAM J. Comput.*, 18:1201–1225, 1989.

[381] P. M. Vaidya. An $O(n \log n)$ algorithm for the all-nearest-neighbors problem. *Discrete Comput. Geom.*, 4:101–115, 1989.

[382] K. R. Varadarajan and P. Agarwal. Approximating shortest paths on an nonconvex polyhedron. In *Proc. 38th Annu. IEEE Sympos. Found. Comput. Sci.*, page To appear, 1997.

[383] H. Wang and P. K. Agarwal. Approximation algorithms for curvature constrained shortest paths. In *Proc. 7th ACM-SIAM Sympos. Discrete Algorithms*, pages 409–418, 1996.

[384] W. Warntz. Transportation, social physics, and the law of refraction. *The Professional Geographer*, 9(4):2–7, 1957.

[385] E. Welzl. Constructing the visibility graph for n line segments in $O(n^2)$ time. *Inform. Process. Lett.*, 20:167–171, 1985.

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

[387] G. Wilfong. Motion planning for an autonomous vehicle. In *Proc. IEEE Internat. Conf. Robot. Autom.*, pages 529–533, 1988.

[388] G. Wilfong. Shortest paths for autonomous vehicles. In *Proc. 6th IEEE Internat. Conf. Robot. Autom.*, pages 15–20, 1989.

[389] D. P. Williamson and M. X. Goemans. Computational experience with an approximation algorithm on large-scale Euclidean matching instances. *INFORMS J. Comput.*, 8(1):29–40, 1996.

[390] C. D. Yang, D. T. Lee, and C. K. Wong. Rectilinear paths problems among rectilinear obstacles revisited. *SIAM J. Comput.*, 24:457–472, 1995.

[391] A. C. Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM J. Comput.*, 11:721–736, 1982.

[392] A. Z. Zelikovsky. An 11/6-approximation algorithm for the network Steiner problem. *Algorithmica*, 9:463–470, 1993.

[393] A. Z. Zelikovsky. Better approximation bounds for the network and Euclidean Steiner tree problems. Technical Report CS-96-06, University of Virginia, Charlottesville, VA, 1996.