Algorithms for Optimizing Search Schedules in a Polygon

by

Stephen Bahun

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2008

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

In the area of motion planning, considerable work has been done on guarding problems, where "guards", modelled as points, must guard a polygonal space from "intruders". Different variants of this problem involve varying a number of factors. The guards performing the search may vary in terms of their number, their mobility, and their range of vision. The model of intruders may or may not allow them to move. The polygon being searched may have a specified starting point, a specified ending point, or neither of these. The typical question asked about one of these problems is whether or not certain polygons can be searched under a particular guarding paradigm defined by the types of guards and intruders.

In this thesis, we focus on two cases of a chain of guards searching a room (polygon with a specific starting point) for mobile intruders. The intruders must never be allowed to escape through the door undetected. In the case of the two guard problem, the guards must start at the door point and move in opposite directions along the boundary of the polygon, never crossing the door point. At all times, the guards must be able to see each other. The search is complete once both guards occupy the same spot elsewhere on the polygon. In the case of a chain of three guards, consecutive guards in the chain must always be visible. Again, the search starts at the door point, and the outer guards of the chain must move from the door in opposite directions. These outer guards must always remain on the boundary of the polygon. The search is complete once the chain lies entirely on a portion of the polygon boundary not containing the door point.

Determining whether a polygon can be searched is a problem in the area of visibility in polygons; further to that, our work is related to the area of planning algorithms. We look for ways to find optimal schedules that minimize the distance or time required to complete the search. This is done by finding shortest paths in visibility diagrams that indicate valid positions for the guards. In the case of the two-guard room search, we are able to find the shortest distance schedule and the quickest schedule. The shortest distance schedule is found in $O(n^2)$ time by solving an L_1 shortest path problem among curved obstacles in two dimensions. The quickest search schedule is found in $O(n^4)$ time by solving an L_∞ shortest path problem among curved obstacles in two dimensions. For the chain of three guards, a search schedule minimizing the total distance travelled by the outer guards is found in $O(n^6)$ time by solving an L_1 shortest path problem among curved obstacles in two dimensions.

Acknowledgements

I would like to thank my supervisor, Anna Lubiw, for her patience, encouragement, and invaluable input during this endeavour.

Contents

List of Figures

Chapter 1

Introduction

The idea of guarding the interior of a polygon was first introduced as the art gallery problem by Klee, with the first published result in [9]. The problem requires selecting a finite number of points (guards with 360° vision) in a polygon such that every point in the polygon is visible from at least one of the guards. Chvatal determines that the number of guards necessary for any polygon with n vertices is at most $\lfloor n/3 \rfloor$. In general however, determining the exact minimum number of guards for a polygon is known to be NP-complete [23]. There have since been many variants and refinements to polygon search problems, including a book devoted to art gallery problems [32] and much more recent work.

A more complicated problem is obtained by considering mobile guards, in which case the possible paths of guards must be considered. The term "watchman" is used for the case of a single mobile guard with vision in all directions at all times [8]. The guard can move anywhere inside the polygon, and the polygon is searched once the guard has seen every point in it. For the *watchman route* problem, the shortest possible cycle that searches the polygon is desired. The best known algorithm to calculate the shortest watchman route runs in time $O(n^3 \log n)$ for a tour with a fixed starting point [13].

The examples discussed so far simply require that every point in the polygon be seen by a guard. If the goal is instead to try to detect mobile intruders, the problem becomes more complex. The problem of searching for mobile intruders inside a polygon was first introduced in [39]. In this case, the guard must follow a search schedule that ensures no intruders are in the room, under the assumption that the intruders can move during the search. A guard that can see in all directions at all times is called an ∞-searcher. More generally, if the guard's vision is limited to the beams from k moveable flashlights, it is called a *k-searcher*. These new aspects of the problem make it even more complex.

Lee et al. [25] consider the case of a 1-searcher in a *room* (polygon with a door point on the boundary). The door point is the starting location for the search, and the 1-searcher must ensure that no intruders in the room may leave through the door undetected. Note that with a 1-searcher, it is imperative that the guard

move along the boundary of the polygon to ensure that the intruder cannot sneak behind the guard. Lee et al. [25] characterize 1-searchable rooms, give an $O(n \log n)$ time algorithm for determining 1-searchability of a room, and give an $O(n^2)$ time algorithm for generating a search schedule (for searchable rooms).

In the variant known as the two-guard room search problem, the flashlight beam of a 1-searcher is replaced by a second guard moving along the boundary of the polygon. Both guards start at the door and must be able to see each other at all times during the search. This problem was first studied by Park et al. [35], and they give an $O(n \log n)$ time algorithm to determine if a room can be searched. This is improved to $O(n)$ by Bhattacharya et al. in [4].

Efrat et al. study a more general version of the two-guard search approach where a chain of n guards is used to search a polygon, where consecutive guards must maintain mutual visibility [14]. Their work focuses on finding the number of guards required to search a given room, and they develop an algorithm that finds the minimum number of guards in $O(n^2)$ time for n-vertex polygons.

The work on mobile intruders and limited visibility of guards has concentrated on identifying which polygons can be searched. The question of optimality of the search path(s) has not usually been addressed. There are several possible optimization criteria. One is to minimize the sum of the distances travelled by all the guards. Another is to minimize the total time spent during the search, assuming some maximum travel speed for the guards. For a single guard, these measures are the same, but when the scenario involves more than just the movement of a single guard, they are different, as we show in Chapter 3.

The main goal of this thesis is to find optimal search schedules for guards searching for mobile intruders. We first look at the problem of two guards searching a room. For any room that can be searched, we find a search schedule that minimizes the total distance that the guards need to travel, and we also find a search schedule that minimizes the amount of time required for the guards to search the room. Our algorithms find the shortest distance schedule in $O(n^2)$ time and the quickest schedule in $O(n^4)$ time. For a chain of three guards, our algorithm finds a schedule minimizing the distance travelled by the outer guards in $O(n^6)$ time.

Our main idea for finding optimal search schedules is to model the problem as a shortest path problem in a *visibility obstruction diagram* [22]. These diagrams indicate which points on the boundary of the polygon represent valid configurations for the guard(s). Visibility obstruction diagrams have been used previously to determine if a polygon is searchable ([22, 14, 4]), but they have not been used before for optimization.

This thesis is organized as follows.

In chapter 2, we go into more detail about variants of the problems discussed above and provide some preliminary background information that is necessary for the details of the thesis.

In chapter 3, we use a form of visibility diagram for two guards searching a room

that contains information such that any path from the starting point in the diagram (both guards at the door) to an ending line (both guards meeting again somewhere on the boundary) corresponds directly to a search schedule for the guards. We find shortest paths in this 2-dimensional visibility diagram corresponding to optimal search schedules for the room. Specifically, the shortest L_1 path from the starting point to the ending line yields a shortest distance search schedule, and the shortest such L_∞ path yields a quickest search schedule.

In chapter 4, we look at the problem of a chain of 3 guards searching a room. In this case, the space of all valid configurations of the guards is 4-dimensional. Because of this, it is not obvious how to solve the resulting search problem in a reasonable running time. Instead, we use a similar approach in searching a 2-dimensional *link diagram* to find a search schedule.

Chapter 2

Background Information

This chapter contains information useful for understanding the results contained within this thesis. In Section 2.1, we first fill in more detail of the background that was given in the introduction, following developments in polygon guarding from the basic art gallery theorem to our specific guarding problem with mobile intruders. Compared with the basic Art Gallery result, our problem involves mobility (of guards and intruders), and also optimization of the search path. It thus falls within the general area of Motion Planning, and more specifically Planning Algorithms. We give an overview of these areas in Section 2.2. The heart of our algorithms involve shortest path computations; we survey results on shortest paths in Section 2.3. Finally, in Section 2.4 we discuss some related aspects of visibility in polygons.

2.1 Previous Work

Consider a scenario in which a polygonal room contains valuables that must be kept safely inside the room. *Guards*, or entities with some form of vision, will be used to detect intruders that may attempt to remove items from the room. We will consider different problems that use varying types of guards, and several aspects of the problem will be considered when determining how to effectively guard a room. It is firstly necessary to know whether or not a specific room can be guarded at all, given certain assumptions about the guards and potential intruders. For any room that can be guarded, the guards need to be told where to go and possibly where to look. There may be a limited number of guards available, and it may be important to determine the minimum number of guards that are needed to guard a particular room. It may also be important to find a shortest route to clear a room using mobile guards. Different ideas about what a guard is capable of (such as whether or not a guard can move) have been considered, and different problems result from these different guarding paradigms.

In the art gallery problem, guards are not allowed to move, but every guard has 360° vision. The goal is to select locations in the polygon for the guards to

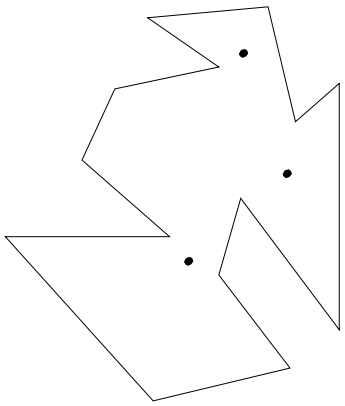

Figure 2.1: The Art Gallery problem - three points guarding the polygon

stand so that every point in the polygon is visible to at least one of the guards. The problem of finding the worst case minimum number of guards needed for an n-vertex polygon was first posed by Klee. Figure 2.1 gives an example of a polygon and guard locations. Chvatal determines that this number is $\lfloor n/3 \rfloor$ [9] and Fisk gives a much simpler proof [15]. These results show that this number holds even when guards are restricted to being placed only at vertices of the polygon. In general however, determining the exact minimum number of guards for a given input polygon is known to be NP-complete [23], even when restricting the possible positions of the guards in this way.

A variant of the art gallery problem with stationary guards allows the guards to see a conical range restricted by some fixed angle. Toth [42] first showed that if the guards have 180° vision and can be placed anywhere inside the polygon, the number of guards always sufficient to guard any n-vertex polygon is still $\lfloor n/3 \rfloor$. When guards have vision in the range of $[90°, 180°)$, up to $2n/3 - 1$ guards may be needed [43]. When the angle of vision for the guards is in $[45°, 60°)$, up to $2\lfloor \frac{n-1}{2} \rfloor$ guards may be needed [44].

The problem of guarding a polygon gets more complicated when the guards may move. The next scenario we will discuss still assumes that a guard can see in all directions at all times, but instead of placing stationary guards, there will be a single moving guard. In the problem of a watchman in a polygon [8], the goal is to find a path inside the polygon for the guard to follow such that every point on the boundary of the polygon is visible from at least one point on the path. Thus a single guard travelling along this path would detect a stationary intruder inside of the polygon. For the *watchman route* problem, the shortest possible cycle that searches the polygon is desired. The best known algorithm to calculate the shortest watchman route runs in time $O(n^3 \log n)$ for a tour with a fixed starting point [13]. The same result gives an upper bound of $O(n^4 \log n)$ on the time complexity for the version of the problem without a fixed point. Figure 2.2 gives an example of a polygon and a watchman route that searches the polygon.

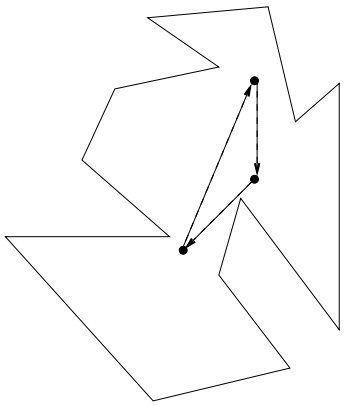

Figure 2.2: Watchman route problem - this route guards the polygon

Having discussed mobile guards, it is natural to consider that any intruders in the room may also be mobile. The next problem we discuss is one in which a single guard must search the polygon to ensure that there are no mobile intruders. No assumptions are made about relative speeds of the guards and the intruders – to handle worst-case scenarios, intruders are assumed to be arbitrarily fast. Suzuki and Yamashita first posed this problem [39]. A guard with 360° vision at all times is called an ∞-searcher in this model. Suzuki and Yamashita also consider that the guard may not be able to see in all directions at all times. Their version of a guard with limited visibility is a guard with k flashlight beams that it can shine in different directions over time. The guard sees along these k rays emitted from his position, thus an intruder crossing one of these light beams would be detected. Such a guard is called a k-searcher; thus a guard with a single flashlight beam is called a 1-searcher. A polygon that can be searched by a k-searcher is said to be k-searchable.

These new aspects of the problem make it more complex. Firstly, note that since a 1-searcher has only one ray of vision and the intruder moves arbitrarily quickly, the 1-searcher must move along the boundary of the polygon to prevent any intruders from moving behind the guard. Consider the polygon in figure 2.3. A guard with a single flashlight starting at point A can search the $BAHG$ region as it moves to B, but if it then moves to C, the flashlight will skip over the edge GF entirely. So an intruder that was hiding near vertex F could move into the $BAHG$ region undetected while the guard is on the edge BC. We say that the $BAHG$ region is *contaminated*. As a result, the guard would need to search the $BAHG$ region again before the polygon is searched completely.

In [39], characterizations are given of certain polygons that can or cannot be searched by guards with certain numbers of flashlights. It was proven in [34] that having k flashlights is no more powerful than having 2, i.e., that any k-searchable polygon is 2-searchable. A path for a 2-searcher can be constructed in $O(n^2)$ time [34]. Checking the 1-searchability of a polygon can be done in $O(n \log n)$ time [33] and a schedule can be constructed in $O(n^2)$ time [22]. There is no mention of

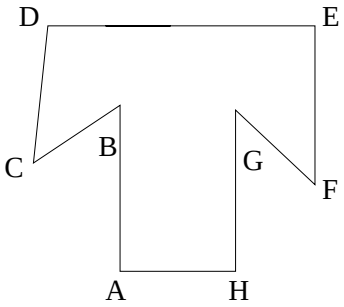

Figure 2.3: Intruder can move undetected from unsearched to searched region

optimality of the search schedules that are found.

Another direction of research that people have pursued is to fix the starting point of the guards' search and require that the searched portion of the polygon always include this start point. This models having a "door" in the room by which the searchers enter and through which any intruders attempt to escape undetected. Lee et al. [25] consider the case of a 1-searcher in a polygon with a door (a *room*). Since intruders can move arbitrarily fast, the search must begin at the door. The goal is to find a search schedule that does not allow intruders to leave through the door undetected and eventually finds any intruders that are in the room. In [25], they characterize 1-searchable rooms, give an $O(n \log n)$ time algorithm for determining 1-searchability of a room, and give an $O(n^2)$ time algorithm for generating a search schedule (for searchable rooms). No concern is shown for finding an optimal search schedule.

A variant known as the two-guard room search problem takes a different approach to the idea of guards searching a room. Imagine the guards as two robots with a light beam between them. As long as the robots are mutually visible, the beam between the robots can remain intact. If an intruder were to cross this line of sight between the robots, the light beam would be broken, and the robots would detect the intruder. In this case, both guards must remain along the boundary of the polygon, starting at the door, and the guards must be able to see each other at all times during the search. This problem was first studied by Park et al. [35], and they give an $O(n \log n)$ time algorithm to determine if a room can be searched. This is improved to $O(n)$ by Bhattacharya et al. in [4]. These works do not give algorithms to output search schedules, let alone optimal search schedules.

An earlier variant of the two-guard room search problem considers the case of searching a polygon starting at a door point and ending at a pre-determined boundary point. In this version, called searching a *corridor* or *street*, the guards may not cross either the starting or the ending point. For this problem, Icking and Klein [20] find a search schedule of minimum total length in time $O(n \log n + k)$ where $k \in O(n^2)$ is the size of the output. Note that while this problem is very similar to the two-guard room search problem, it is not possible to reuse this algorithm to find a search schedule of minimum total length by testing all possible final meeting

points of the two guards, since there are rooms that can be searched only by allowing the guards to cross the final meeting point [35]. Their approach involves querying the polygon repeatedly while constructing the search schedule, while our approach makes use of a secondary data structure built from the polygon. This appears to be the first appearance of an optimal schedule for a search by multiple guards.

The two-guard approach can be extended to a chain of n guards with a light beam between each pair of consecutive guards; then each of these pairs must maintain mutual visibility at all times. This version is studied by Efrat et al. [14], and they consider searching a polygon (without a door point). Their work focuses on finding the number of guards required to search a given polygon. Tan [41] gives an algorithm that finds the minimum number of guards r^* required to search an n-vertex polygon in $O(n^2)$ time and an algorithm to generate a search schedule in $O(r^*n^2)$ time. In [40], Tan finds an $O(n \log n)$ time algorithm to determine if a street can be searched by a chain of three guards and generates a search schedule in $O(n \log n + m)$ time, where $m \in O(n^2)$ is the size of the schedule.

In contrast to the majority of the previous work in polygon search problems, this thesis focuses on optimizing search schedules. We examine problems involving a chain of two or three guards searching for a mobile intruder in a polygonal room with a door. For the case of two-guard room search, we develop an algorithm to find the minimum-distance search schedule for any polygon in $O(n^2)$ time and an algorithm to find the minimum-time search schedule for any polygon in $O(n^4)$ time. For the case of searching a room by a chain of 3 guards, we find a schedule that minimizes the distance travelled by the outer guards in $O(n^6)$ time.

2.2 Planning Algorithms

This section provides a very brief survey of planning algorithms and some details from the subarea of motion planning [21].

We consider a *plan* to be a sequence of actions that will accomplish a particular task from a specific starting state. The job of a *planning algorithm* is to create a desired plan to accomplish a task when given a specific starting state as its input. Consider the task of moving from the start of a maze to its exit. A specific sequence of movements that will solve a particular maze is a plan. An algorithm that will generate a solution to any given maze is a planning algorithm. This maze problem lies in the realm of *motion planning*, where the plan describes the movement of some object [5]. In the area of robotics, where motion planning problems tend to arise, more complicated problems can involve multiple objects, complex shapes, and online situations. Planning problems in other areas, such as artificial intelligence and control theory, consider a range of other situations like any general achievement of a goal state or manipulation of values in order to maintain a consistent desired output over time [21].

In polygon search problems using mobile guards, finding a way to search the

polygon is a planning problem. Given a polygon, possibly with a door point, an algorithm to find a search schedule for the guards to follow is a planning algorithm. Polygon search problems discussed in this work will make use of some ideas from the area of planning.

2.2.1 Configuration Space

For any particular motion planning problem, the set of all possible states is known as the *configuration space*. With objects moving in the continuous realm, the configuration space is continuous, giving an infinite number of configurations. For a simple scenario of a dot moving along a line segment, the configuration space is simply the entire line segment. If the dot can move in two dimensions (inside a polygon, for example), the configuration space is also 2-dimensional.

If we consider the case of a 1-searcher, where the guard and flashlight beam can move independently, each configuration no longer represents simply a point in space, but rather the combination of the location of the guard and the direction of the flashlight beam. In the case of a 1-searcher searching for a mobile intruder inside a polygon, where the guard may only move along the boundary of the polygon, we can consider the flashlight beam as also being located on the boundary of the polygon (where the beam first hits a wall). Thus the movement of the guard is 1-dimensional along the perimeter of the polygon, and the movement of the flashlight beam is 1-dimensional, also along the perimeter of the polygon; therefore the configuration space is 2-dimensional.

A path through the configuration space represents a plan of continuous states. Thus a plan to solve a particular problem corresponds to some path through the configuration space of that problem. However, not every state in the configuration space is a valid state for the problem being considered. These are manifest as obstacles in the configuration space. For example, in a polygon with a reflex vertex, there are points on the polygon boundary that cannot see every other point in the polygon. Taking a pair of points that are not mutually visible, it is not possible for a 1-searcher to shine its flashlight beam from one of these points to the other. Thus, in the configuration space for this kind of problem, the reflex vertex would cause an obstacle to appear. A continuous path through the configuration space that does not intersect any obstacles corresponds to a valid plan, and a path from a starting configuration to a desired state is a solution to the problem.

2.3 Shortest Path Problems

A common optimization problem with respect to paths is to find a shortest distance path between two endpoints; this is the type of problem that we need to solve for our results. In a graph, the shortest path from a node to any other node along edges of the graph is typically found using Dijkstra's algorithm [11]. In a geometric shortest

path problem, a path is not confined to edges. Instead, there are other restrictions on feasible paths, such as obstacles that must not be intersected or polygon edges that must not be crossed. For a survey on problems related to shortest paths, see [29].

2.3.1 Distance Metrics

The typical geometric shortest path problem concerns Euclidean space, where the distance between two points is equal to the length of the direct straight line segment between the two points. Shortest paths can be considered in different metrics where the distance between two points is defined in alternative ways. Consider the L_p metrics, where the distance between two points in 2-dimensional space is defined to be $((\Delta x)^p + (\Delta y)^p)^{1/p}$; Euclidean space is considered the L_2 metric. We will look at two more special values for p, giving us the L_1 and L_∞ metrics.

In the L_1 metric, the distance between 2 points in 2-dimensional space is $(\Delta x + \Delta y)$, or in other words, the sum of the horizontal and vertical distances between the two points. In the L_∞ metric, the distance between 2 points in 2-dimensional space is $max((\Delta x), (\Delta y))$, or in other words, the greater of the horizontal and vertical distances between the two points. These are related in that a unit disc is a square in both cases, though rotated 45° in the case of the L_1 metric where it resembles a diamond shape [24]. In general, we denote the length of a path π in the L_p metric as $|\pi|_p$.

A geometric path need not consist of line segments. Thus we must consider the length of a general curve when determining the length of a path. The length of a curve $C = C(t), t \in [0, T]$ in any metric is defined to be $\sup(\sum_{i=1}^{k} d(C(t_i), C(t_{i-1})))$ where the sup is taken over partitions $t_0, \ldots, t_k$ of $[0, T]$, as long as the supremum exists [16]. Thus, the distance along a path in any metric is defined in this way. This distance is finite for *rectifiable curves*, which are the only curves we are concerned with.

2.3.2 Algorithmic Approaches for Geometric Shortest Paths

We will examine two general approaches to solving geometric shortest path problems. Both approaches are based on ideas for solving shortest path problems in the simpler realm of graphs.

Graph Approaches

Shortest path problems on graphs tend to be less complicated to solve than geometric shortest path problems due to the discrete nature of graphs. Some geometric shortest path problems can be simplified by considering a finite subset of points that are particularly significant. For example, in the L_2 metric, it can be shown

that the shortest path between points amongst polygonal obstacles bends only at corners of obstacles. These points are represented by vertices in a new graph, and distances between pairs of points that can see each other are the weights on the edges between the corresponding pairs of vertices. This visibility graph can be computed in $O(n \log n + m)$ time [36], [37], [38]. Then the shortest path between the original points can be found by running Dijkstra's Algorithm on this graph in $O(n \log n + m)$ time. The same approach to solving the problem will work in other metrics, as long as an appropriate definition of mutually-visible points is used when constructing the visibility graph.

Continuous Dijkstra

Another approach to the problem would be to try to find the shortest path directly, without having to create the entire visibility graph. The technique, known as *Continuous Dijkstra* [30], uses an approach similar to the way Dijkstra's algorithm works on a graph. Starting from the source point, the algorithm considers all points that can be reached within a particular distance, which increases during the algorithm. When these points include an obstacle vertex, the subsequent calculation of all points at a particular (slightly greater) distance from the source may depend on travelling along a path around this obstacle. This is one kind of *event* that may occur during the calculation, and there are other types as well. Eventually, the set of points will include the destination point, and the shortest path to get there is reconstructed based on events that occurred along the way. The running time of the algorithm depends on the efficiency of determining the occurrence of the events and making updates. Hershberger and Suri derive a running time of $O(n \log n)$ for finding shortest L_2 paths using the Continuous Dijkstra approach [19].

2.4 Visibility in Polygons

An interesting aspect of polygons concerns how pairs of points are visible to one another within the polygon. Clearly, polygon search problems are related to visibility within polygons. Ghosh [17] provides a thorough examination of visibility algorithms on polygons. In this section, we briefly give some explanations of a few concepts relevant to our work.

An *LR-visible* polygon is a polygon with two points s and t such that every point on the clockwise boundary from s to t is visible from some point on the clockwise boundary from t to s, and every point on the clockwise boundary from t to s is visible from some point on the clockwise boundary from s to t [17]. LR-visibility has been used as a factor in determining the searchability of polygons under several guarding paradigms [45], [4].

For any polygon P, let ∂P denote the boundary of the polygon. When we discuss mutual visibility of a pair of points, we are using the notion of *weak* visibility; if

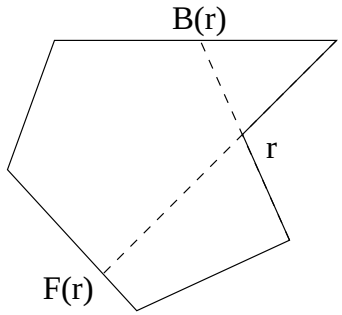

Figure 2.4: Backward and forward extension points of a reflex vertex

two points on ∂P are mutually visible, then the line segment connecting them lies inside P and does not intersect the interior of any disjoint edges. Thus, two points on the same edge of P are mutually visible.

In a convex polygon, all points of the polygon see each other. More interesting polygons have reflex vertices, which prevent certain polygon points from seeing each other. For any reflex vertex r, we define the *forward* and *backward extension points*, $F(r)$ and $B(r)$, to be the points where two rays extending inward from the edges incident with r first intersect the boundary of the polygon. See Figure 2.4 for an example. These points are notable since each one separates a boundary region that can see all points along an edge incident with r from one that cannot see any of those points besides r.

LaValle et al. [22] introduce a visual representation of the visibility between all pairs of points on the boundary of a polygon. This idea of a visibility obstruction diagram has been very useful in solving problems related to polygon search and is vital to the results in this thesis. We describe the relevant diagrams more specifically in Section 3.4 and Section 4.3. Also, see [46] for an extensive examination into this area.

Chapter 3

Room Search by Two Guards

3.1 Introduction

Let (P, d) denote a *room*, which is a simple polygon P with a designated point d on its boundary, called a *door*. The two guards are allowed to walk along the boundary of the room. They both start at d and initially go in opposite directions. Neither one may cross d at any time. The goal is for them to maintain their mutual visibility at all times and meet again somewhere else on ∂P, at which point the whole room has been searched for a mobile intruder. This problem, a variation on searching for a mobile intruder inside a polygon by a 1-searcher [39], was first studied by Park et al. [35]. For more background, see section 2.1.

In this chapter we will present an algorithm that will find optimal search schedules, if they exist, using two different notions of optimality: the shortest distance travelled and the shortest length of time required to search the room. The shortest distance schedule is found in $O(n^2)$ time and the quickest schedule is found in $O(n^4)$ time. This algorithm makes use of a search space that is a visibility diagram describing the valid positions for the two guards to maintain mutual visibility. The concept was first discussed in [22] and later modified by Zhang [46]. To maintain relevance to the current problem, we use the latter version, which assumes that two points on a single polygon edge are mutually visible. Our main lemma is that minimizing the distance (resp. time) of the room search schedule corresponds to minimizing the length of a path in the search space under the L_1 (resp. L_∞) metric.

In Section 3.2, we briefly look at the relationship between searching with a single guard with one flashlight and searching with two guards. In Section 3.3, we show how optimizing distance and optimizing time may result in different search schedules. In Section 3.4, we describe the visibility diagram we will use in our algorithms. In Section 3.5, we discuss the relationship between paths in this diagram and schedules for searching the room. In Section 3.6, we describe how to contruct the diagram and analyze the runtime of this procedure. Then in Section 3.7, we give the details of the algorithms to find schedules minimizing overall distance and time of a search schedule.

3.2 Two Guards vs. a 1-Searcher

The two guard problem is very similar to another room search variant [25] involving a *1-searcher*. A 1-searcher is a single guard with a flashlight who must remain on the boundary of the polygon and may move the direction of the the flashlight beam. The goal is to search the entire polygon by the flashlight beam in such a way as to ensure that no intruder could have possibly reached the door undetected by the guard and flashlight, and that no intruder inside the room remains undetected at the end of the search.

It is easy to see that any search schedule for two guards maintaining mutual visibility yields a search schedule for a single guard with a flashlight. The motion of one of the guards corresponds to the motion of the 1-searcher. The motion of the other guard corresponds to the movement of the light beam along edges of the polygon. Since there is mutual visibility between the two guards at every point in time, the corresponding locations of the guard and flashlight beam are feasible. Also, since intruders have nowhere to hide in the two-guard version, there is no way for intruders to avoid the 1-searcher.

The converse fails: not every search schedule for a 1-searcher corresponds to a search schedule for two guards. This is because a continuous change in the angle of the flashlight may correspond to a discontinuous change in the position of the projection of the light onto polygon edges. Zhang demonstrates an example showing this [46].

3.3 Shortest Distance Schedule vs. Quickest Schedule

Minimizing the distance and minimizing the time required for searching a room may result in different search schedules. Figure 3.1 shows an example of a room, with a door at 0, where the two notions of optimal schedule give two different search schedules. The shortest distance is achieved if one guard travels from 0 to 2 while the other guard waits at 5. The resulting total distance travelled at the end of the schedule is equal to 24 (the perimeter of the polygon). This schedule takes at least 19 time units, since the guard travelling from 0 to 2 travels a distance of at least 19. The quickest search schedule involves one guard travelling along 0, 5, 4, 3, and 2 while the other guard must go from 0 to b and back to a to maintain mutual visibility. This requires backtracking from b to a, which results in a total distance travelled that is greater than 24, but it takes less than 15 time units to complete the entire search.

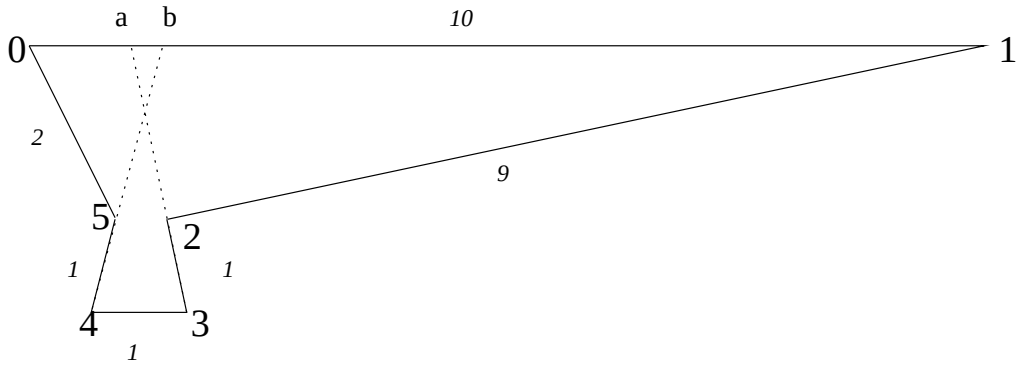

Figure 3.1: Shortest distance schedule is different from quickest schedule

3.4 V-Diagram

For any room (P, d), the *visibility diagram* [22] encodes, for any pair of points on the boundary of P, whether or not the points are mutually visible. We modify a visibility diagram from [4] and call it the *V-diagram*; it contains the minimum amount of relevant information for the room search problem. The diagram's boundary is a right triangle with the left and top sides equal in length to the perimeter of P. With the top left corner $(0, 0)$ representing the door d, the x (resp. y) coordinate represents the distance along ∂P from the door in the clockwise (resp. counterclockwise) direction. Then we can associate any point (x, y) in the diagram with positions of the two guards on ∂P.

To represent the mutual visibility of pairs of points, any point (x, y) in the V-diagram is shaded iff the corresponding positions of the two guards are not mutually visible in the polygon. Figure 3.2 shows an example of this, with vertex labels on the axes marking off distances from the door to the vertex. Shaded regions make up *obstacles*. Each obstacle in the V-diagram is a union of *barriers*, where a barrier represents the guard positions blocked by one reflex vertex. A barrier has a horizontal and vertical side determined by the two edges adjacent to the reflex vertex [46]. The remainder of the barrier boundary is defined by the pairs of points on the polygon that are connected by a line that also goes through the reflex vertex. For example, obstacle A in Figure 3.2 is composed of two barriers, associated with reflex vertices 9 and 10. The box in the figure shows the portion of the barrier where vertex 9 would obstruct the view between guards on edges (6,7) and (0,13).

Any point on the diagonal corresponds to a meeting point of the two guards. Then any path π in the V-diagram from the top-left corner (the door) to the diagonal (the *goal*) that does not cross any shaded areas (obstacles) corresponds to a valid search schedule s of P; we say that $s = S(\pi)$. See Figure 3.3 for an example of a search schedule corresponding to a path in the V-Diagram.

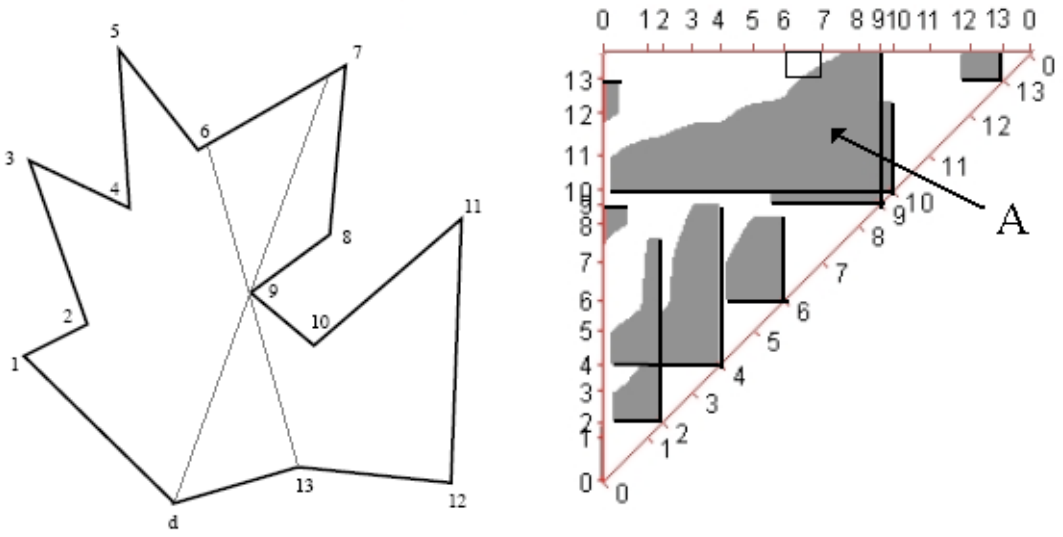

Figure 3.2: Room (door at 0) and its V-diagram from [4] (obstacles shaded)

3.5 Correspondence Between Paths in the V-diagram and Search Schedules

The V-diagram can be used to construct optimal schedules for searching the room. We first describe a relationship between the length of a path in the V-diagram and the distance/time of the corresponding room search schedule. We will assume that each guard can travel independently at varying speeds in the range [0, 1] both forwards and backwards (without crossing the door point). For any search schedule s, let $D(s)$ denote the distance travelled by the guards during s, and let $T(s)$ denote the time required for s. Recall that $|\pi|_p$ denotes the length of a path π in the L_p metric.

Lemma 3.5.1 *Let π be a path in the V-diagram for a room and $s = S(\pi)$ be the corresponding search schedule of the room. Then $|\pi|_1 = D(s)$ and $|\pi|_\infty = T(s)$.*

In order to prove this lemma we must define $D(s)$ and $T(s)$ more precisely. Let us first recall how the length of a curve is defined. A distance metric $d(a, b)$ defines the distance between a pair of points a and b. This is the length of the line segment from a to b. As mentioned in Section 2.3.1, the length of a general curve $\pi = \pi(t), t \in [0, T]$ is defined to be $\sup(\sum_{i=1}^{k} d(\pi(t_i), \pi(t_{i-1})))$ where the sup is taken over partitions $t_0, \ldots, t_k$ of $[0, T]$ [16]. The sup exists for curves with finite lengths, or *rectifiable* curves, and these are the only ones with which we are concerned.

We define $T(s)$ and $D(s)$ in a similar fashion, first for straight line segments and then using a sup.

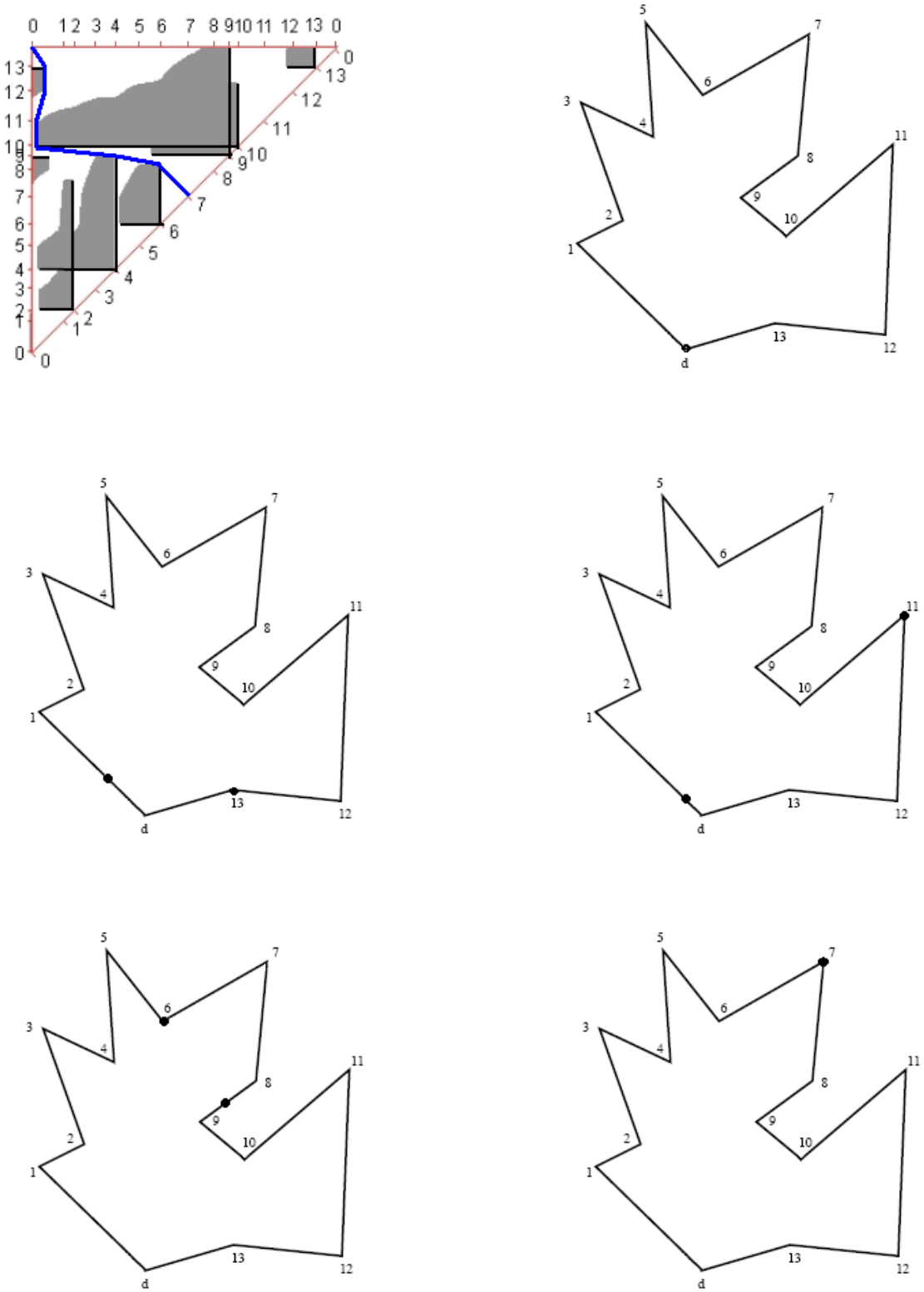

Figure 3.3: V-Diagram path and snapshots of corresponding search schedule

Proof. [Proof of Lemma 3.5.1] By the above clarification of the definition of $D(s)$ and $T(s)$ it suffices to prove the result for a path π that is a straight line segment.

In the L_1 metric, $|\pi|_1 = |a-b|_1 = |(a-b)_x| + |(a-b)_y|$. This represents the sum of the distances travelled by the two guards, so $\pi_1 = D(s)$.

In the L_∞ metric, $|\pi|_\infty = |a-b|_\infty = \max(|(a-b)_x|, |(a-b)_y|)$. Since both guards have the same maximum speed of 1, assume that the guard travelling the greatest distance $\max(|(a-b)_x|, |(a-b)_y|) = |\pi|_\infty$ travels at speed 1 to minimize the time spent. Then the time it takes that guard to travel that distance is $|\pi|_\infty$, and the other guard can easily travel the shorter distance during that amount of time. Thus it takes $|\pi|_\infty$ time for both guards to travel along π. We then have $|\pi|_\infty = T(s)$.

□

Corollary 3.5.2 *The search schedule minimizing the total distance travelled by the two guards corresponds to the shortest path in the L_1 metric from the door to the goal in the V-diagram. The search schedule minimizing the time (i.e., the quickest search schedule) corresponds to the shortest path in the L_∞ metric from the door to the goal in the V-diagram.*

3.6 V-diagram Construction

We now discuss the nature of the V-diagram and how it is constructed.

Theorem 3.6.1 *The border of each obstacle in the V-diagram is piecewise hyperbolic.*

We begin by proving an intermediate result.

Lemma 3.6.2 *When a single reflex vertex obstructs visibility between two edges in the polygon, the curve on the corresponding region of a shaded portion of the V-diagram is hyperbolic.*

Proof. Refer to Figure 3.4. We will assign coordinates to the relevant part of the polygon. Without loss of generality, assume that one of the endpoints of one of the edges is $(0,0)$, and the direction along that side is represented by the vector $(0,1)$. Then let $R = (r_1, r_2)$ be the reflex vertex, let (a_1, a_2) be one endpoint of the other edge, and let (v_1, v_2) be the normalized vector representing the direction along the other edge. Let $C = (0,0) + s(0,1)$ be any point along the first edge and let $D = (a_1, a_2) + t(v_1, v_2)$ be any point along the second edge. The barrier of the shaded region has a curve corresponding to the points along the edges where the visibility line along the edges goes through R (that is, when CR and DR are

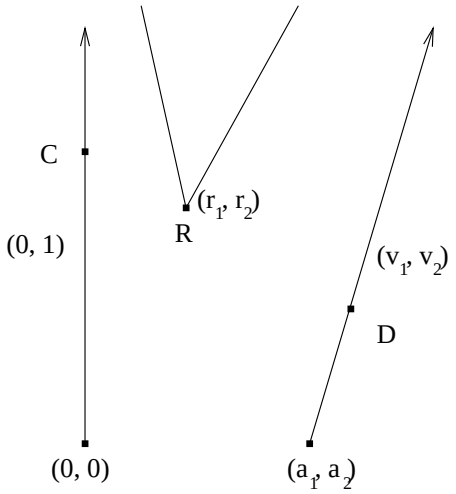

Figure 3.4: Movement of mutually visible guards with a reflex vertex

parallel). Then by treating the points as points in three dimensions, we can use the vector cross product to decide that two points along the edges give a point on this curve whenever

$$
\begin{aligned}
& CR \times DR = 0 \\
\iff & (((0,0) + s(0,1)) - (r_1, r_2)) \times ((a_1, a_2) + t(v_1, v_2) - (r_1, r_2)) = 0 \\
\iff & (-r_1, s - r_2) \times (a_1 + tv_1 - r_1, a_2 + tv_2 - r_2) = 0 \\
\iff & -r_1 a_2 - r_1 t v_2 + s a_1 + s t v_1 - s r_1 + r_2 a_1 + r_2 t v_1 = 0 \\
\iff & -s(a_2 + t v_2 - r_2) = -r_1 a_2 - r_1 t v_2 + r_2 a_1 + r_2 t v_1 \\
\iff & s = \frac{r_1 a_2 + r_1 t v_2 - r_2 a_1 - r_2 t v_1}{a_1 + t v_1 - r_1}
\end{aligned}
$$

Note that this expression is a quotient of two terms that are linear in t. Thus the relationship between the movement along one edge and the movement of the projection of the line of sight along another edge of the polygon is given by the equation of a hyperbolic curve.

□

We now prove Theorem 3.6.1.

Proof. From the above lemma, we know that a single reflex vertex obstructing visibility between two edges of the polygon corresponds to a hyperbolic portion of a barrier in the V-diagram. This is true for any pair of edges with mutual visibility obstructed by the reflex vertex, so each barrier is made up of hyperbolic pieces. Each obstacle is a union of barriers, so each obstacle is piecewise hyperbolic. □

To construct the V-diagram we need to accurately describe all of the obstacles. To do this, we must, for each reflex vertex r:

1. Find the backward and forward extension points, $B(r)$ and $F(r)$ (recall from Section 2.4). This takes $O(n)$ time.

2. Starting with the line through r and $B(r)$ and ending with the line through r and $F(r)$, sweep the line in a clockwise rotation about r. During each portion of the sweep in which the same 2 polygon edges intersect the sweep line, find the hyperbolic curve of the V-diagram representing the pairs of points on the polygon that have a connecting line through r. For each new polygon edge encountered, a new hyperbolic portion is determined. This takes $O(n)$ time.

Therefore, with $O(n)$ reflex vertices, the exact visibility diagram can be described in $O(n^2)$ time.

3.7 Algorithms for Optimal Schedules

We have now reduced the problem of finding optimal search schedules with respect to distance and time to the problem of finding shortest paths in the L_1 and L_∞ metrics among curved obstacles in the plane that are piecewise hyperbolic.

We note the following two properties of L_1 and L_∞ shortest paths:

1. Between any two points there is a shortest L_1 path that is rectilinear. This remains true in the presence of curved obstacles, except in the situation – that doesn't arise for us – where the shortest path travels between two abutting curved objects.

2. Recall from Section 2.3.1, the L_1 and L_∞ norms are related by a linear mapping; in particular we can find shortest L_∞ paths by rotating the plane and its obstacles by 45° and scaling, and then finding shortest L_1 paths.

Thus it suffices to find shortest L_1 paths, either among the original obstacles, or among the obstacles rotated by 45°.

There is considerable work on finding shortest L_1 paths among polygonal obstacles in the plane [7, 28]. There is also work on "curvilinear" computational geometry [12] which has led to shortest path algorithms among "splinegons" [27]. However, there appears to be no solution in the literature to finding shortest L_1 paths among curved obstacles. Recall the two basic approaches to solving this problem described in Section 2.3.2. The continuous Dijkstra approach is used by Mitchell [28] for polygonal obstacles. Alternatively, the problem may be modelled as a graph shortest path problem [10]. The former approach will likely lead to a more efficient solution, but we will simply claim a polynomial time algorithm via modelling the problem on a graph.

As vertices of the graph, we will use the local x or y extreme points of the obstacles, with the understanding that if a whole segment of an obstacle boundary is extreme (i.e. is horizontal or vertical) then we only take the endpoints of the segment.

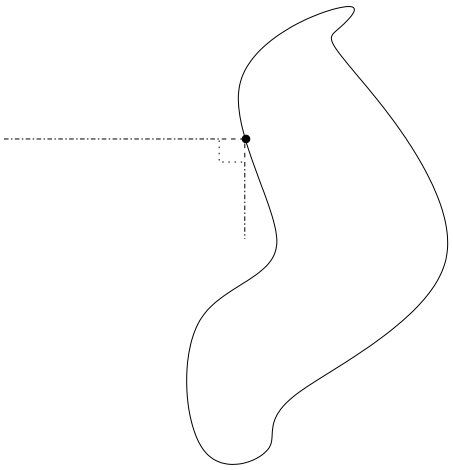

Figure 3.5: Intersection of path and obstacle

Lemma 3.7.1 *Between any two points there exists a shortest path among obstacles in the L_1 metric such that the path intersects obstacle boundaries only at local x or y extreme points of the obstacles, and such that the portion of the path between two consecutive such points is monotone in x and y.*

Proof. By note (1) at the beginning of section 3.7, there is a shortest path that is rectilinear. Suppose π is a shortest rectilinear path that contains obstacle intersection points that are not local extreme points. Let $q = \pi(t)$ be the first such point on π.

If π does not change direction at q then q is a local extreme point. Otherwise π makes a right angle turn at q, and we can alter the path as shown in Figure 3.5; the dotted path turns before q, avoiding it, and then rejoins the original path. The revised path has the same length, and if the detour is small enough, the revised path intersects no obstacles. Applying this inductively yields the desired path.

Now suppose that the path is not monotone between two consecutive extreme points. Then, assuming the path does not backtrack on itself, somewhere on the path there are 3 consecutive segments such that the first and third go in opposite directions (w would be the second of these segments in Figure 3.6). Since there are no extreme points on the second segment, it can be translated so as to shorten the lengths of the first and third segments without intersecting any obstacles (w can be shifted down without hitting any obstacles). Thus the path can be shortened until an obstacle is hit, which happens at an extreme point. So any such non-monotone subpath between consecutive extreme points can be shortened.

□

This lemma justifies creating a graph whose vertices are the extreme points of obstacles and whose edges correspond to monotone paths between pairs of points. We will in fact use a subset of these edges, not because it reduces the quadratic

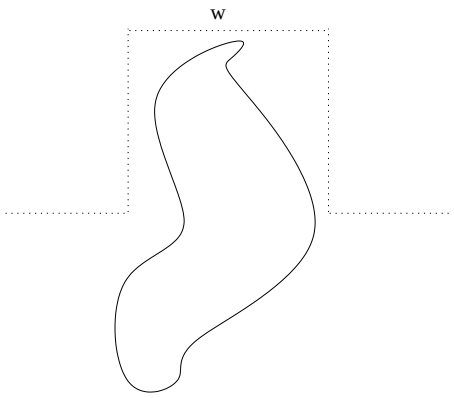

Figure 3.6: Shortest L_1 subpaths between extreme points

number of edges, but because it simplifies finding the edges. If there is a monotone path between two points then there is a *lowest* monotone path, the lower envelope of all monotone paths. A lowest monotone path is *minimal* if it does not go through an extreme point of an obstacle except at its endpoints. Observe that a non-minimal path is a concatenation of minimal paths.

This justifies restricting the edges of the graph to minimal lowest monotone paths between pairs of points. The weight of an edge is the L_1 distance between the endpoints.

We add one additional vertex g to represent the goal line in the V-diagram. In the case where the goal line is 45° from horizontal (i.e., when finding the shortest distance search schedule), the vertex is treated as if it were located in the V-diagram at the point mirrored by d through the goal line. The edges and edge weights to g are defined as those in the rest of the graph. Thus, g is equidistant in the L_1 metric from all points on the goal line (that distance being the perimeter of P), so any shortest path to g from the starting point in the V-diagram (where the path goes through the goal line) contains a shortest path to the goal line.

In the case where the goal line is horizontal (i.e., the rotated case when finding a quickest search schedule), we use a more direct approach, since there is no point that is equidistant from all points on the goal line. We add the vertex g to the graph, and we add an edge between it and an extreme point iff the vertical path between the extreme point and the goal line does not cross any obstacles. To determine this for any particular vertex, we need only to determine if any obstacles lie directly below the extreme point. The edge weight is the distance between the point and the goal line.

Let N be the number of vertices of the graph. Since there are $O(n)$ barriers each with $O(n)$ extreme points, thus N is $O(n^2)$. The graph has $O(N^2)$ edges. In the case where the obstacles are not rotated by 45° (the original L_1 case) we obtain a tighter bound.

Claim 3.7.2 *There are $O(n)$ vertices in the graph in the L_1 case.*

Proof. We show that each of the $O(n)$ barriers in the V-diagram of a simple polygon contains $O(1)$ extreme points. The boundary of each barrier consists of one horizontal segment, one vertical segment, and a curve joining them that is composed of hyperbolic sections (Theorem 3.6.1). The horizontal and vertical portions define 3 extreme points. We claim that the remaining curve is weakly monotone in x and y, which implies that there are no extreme points along it, with the possible exception of 2 points having the same extreme values as the 2 of the 3 previously-mentioned extreme points.

Let v be the reflex vertex corresponding to the barrier. An x coordinate of a point P on the curve of the barrier's boundary corresponds to the position of one guard on the polygon boundary. There is a unique line through the guard's position and vertex v. Continuing this line on the other side of v, it will intersect the polygon again. If the intersection occurs at an edge of the polygon that is collinear with the line, then all points along that edge determine y coordinates for that x coordinate on the barrier; this corresponds to a vertical portion of the obstacle boundary. Otherwise, we are concerned with the first time the line intersects the polygon boundary again to determine a single y coordinate; no additional points of intersection between the line and the polygon boundary are actually able to see the reflex vertex. Thus the curve defined in this way gives no local maxima or minima.

□

We now show how to construct the graph. Find the extreme points of the obstacles, and the *pieces* of obstacle boundaries between extreme points. Note that these pieces are monotone. For each extreme point shoot rays downward and to the left and the right stopping when we hit the first obstacle boundary piece encountered. We claim that this can all be done in $O(N \log N)$ time using plane sweeps in the x and y directions.

Consider two extreme points ρ and ψ, with ρ higher. Suppose ψ is to the right of ρ.

Claim 3.7.3 *There is a minimal lowest monotone path between ρ and ψ iff (1) the ray down from ρ meets the ray left from ψ (i.e. before either encounters an obstacle), or (2) the two rays meet the same piece of obstacle boundary.*

Proof. ($\Leftarrow$) If the ray down from ρ meets the ray left from ψ, then these rays define a minimal lowest monotone path between the two points.

If the two rays do not meet each other, but they meet the same piece of obstacle boundary, then these rays and the piece of obstacle boundary (which has no extreme points on it) define a minimal lowest monotone path between the points.

($\Rightarrow$) Suppose there is a minimal lowest monotone path μ between ρ and ψ. Because the path is a lowest monotone path, no point on it can be pushed downward, so each point on it lies on the ray down from ρ or on the ray left from ψ, or on an obstacle

boundary. Furthermore, monotonicity implies that the path cannot leave and return to either of the two rays. Thus the path travels down from ρ, then possibly along obstacle boundaries, then right to ψ. If the path does not encounter obstacle boundaries, condition (1) is satisfied. Otherwise, because the path is minimal, it does not encounter an extreme point along the obstacle boundary. Thus the ray down from ρ and the ray left from ψ meet the same piece of obstacle boundary and condition (2) is satisfied.

□

An analogous result holds in case ψ is to the left of ρ. We can test intersection of rays during the plane sweep. It remains to identify edges arising from condition (2). For any piece of obstacle boundary b, let U be the set of extreme points whose downward ray hits b, let R be the set of extreme points whose leftward ray hits b, and let L be the set of extreme points whose rightward ray hits b. Note that R or L is empty. We add an edge between any pair $\rho \in U$ and $\psi \in R$ if ρ is above and to the left of ψ. We add an edge between any pair $\rho \in U$ and $\psi \in L$ if ρ is above and to the right of ψ. The number of edges is $O(N^2)$ and we can output them in that time.

Note that both cases of adding g to the graph are taken care of easily. When adding g directly to the V-diagram before creating the graph, the graph creation procedure ensures that the proper edges incident with g are added, along with their proper edge weights. In the other case, determining if an extreme point lies directly above an obstacle is already taken care of when finding the first obstacle (if any) that is encountered by shooting a ray downward from the extreme point.

On the constructed graph, Dijkstra's algorithm finds a shortest path from the door vertex to g in $O(N^2)$ time. From this we can recover a shortest path in the V-diagram and hence an optimal room search schedule. Thus, we find a shortest distance schedule in $O(n^2)$ time and a quickest search schedule in $O(n^4)$ time.

Chapter 4

Room Search by a Chain of Three Guards

4.1 Introduction to Problem

In this chapter, we explore a generalization of the two-guard room search problem where, instead of two guards, we have a chain of k guards. This problem was first considered in [14] for the case of a polygon without a door point. We look at the version with a door point d, and with just 3 guards. Let (P, d) denote a *room* as defined earlier. All three guards are initially at the door point d. Two guards, the first and last one on the chain, are allowed to walk along the boundary of the room, starting at d, initially going in opposite directions. Neither one may cross d at any time. The rest of the k guards (in our case, the only other guard) may move inside the polygon. At all times, the chain of guards must be *connected* such that every pair of consecutive guards maintains mutual visibility. The goal is for the entire room to be swept by the chain of guards; this happens when the chain lies along a contiguous portion of the polygon boundary not containing d.

Our goal is to solve the same problems as we did for two-guard room search. In particular, we will explore the relationship between this problem and the search problem for a 2-searcher (one guard with 2 flashlights). We would also like to find search schedules that minimize the sum of the distances travelled by the 3 guards, or the total time spent in the search. When optimizing search schedules for two guards, our approach of searching the V-diagram required finding a shortest path in the 2-dimensional configuration space of the problem. Using the same approach to find the search schedule that minimizes the total distance travelled by all guards or the time required for the search would require solving shortest path problems in the 4-dimensional configuration space. While there are results for some shortest path problems among obstacles in three dimensions, they tend to be NP-hard [31]. We expect that it is hard to find shortest paths among obstacles in 4 dimensions, so for now, we will restrict ourselves to finding a schedule that minimizes the total distance travelled by the two guards on the border of the polygon, as it is a manageable

problem to solve. Since this will involve searching through a diagram that does not model the entire configuration space, it is necessary to show that a path through this space, which will define motion for the outer guards, leads to a valid search schedule for the chain of 3 guards. We prove this in detail in Section 4.4.

Efrat et al. [14] find the minimum number of guards needed to search a polygon in this way, and they introduce a *link diagram* that slightly resembles the V-diagram for two guards. The shape of the diagram is the same in that each point in the diagram corresponds to a pair of points on ∂P. Instead of having shaded regions for pairs of guard positions where the two guards cannot see each other, each point in the diagram is labelled with the *link distance* between the corresponding two points on ∂P. The link distance of a pair of points in a polygon is the minimum number of edges in a path in the polygon joining the two points. If the points are mutually visible, the link distance is 1; if they are not mutually visible, but there is a 2-edge path between the points that lies inside P, the link distance is 2. Each region of points with the same link distance is grouped together. A path in the link diagram defines motion for the outer guards, which implies a search schedule for a chain of guards. If the starting and ending points of the path correspond to meeting points of the outer guards, and the corresponding distance travelled by the guards is equal to the perimeter of the polygon, then a search schedule for the entire polygon exists. If the path goes through regions with link distance at most $k-1$, then k guards are needed to perform that search. Efrat et al. [14] use this approach to find the minimum number of guards required to search the polygon by an algorithm with running time $O(n^3)$. Tan [41] describes a faster $O(n^2)$ time algorithm to find this number.

In this chapter, we find schedules to minimize the distance travelled by the outer guards when searching a polygonal room. In Section 4.2, we describe the difference between a chain of three guards and a single guard with 2 flashlights (i.e., a 2-searcher). In Section 4.3, we discuss the modified version of the link diagram that we need to solve our problem. In Section 4.4, we show how paths in the link diagram correspond to search schedules for the guards. Finally, in Section 4.5, we give the algorithm for finding a search schedule minimizing the distance travelled by the outer guards.

4.2 Three Guards vs. A 2-Searcher

In this section, we compare the power of a chain of three guards versus a 2-searcher (a guard with two flashlights). This section is not necessary for understanding the rest of the chapter. Recall from Section 3.2 that a 1-searcher (i.e., a searcher with a flashlight) can search any room that can be searched by two guards, but the converse is not true [46]. We prove the analogous result that a 2-searcher can search any room that can be searched by a chain of three guards, but that the converse is false.

Claim 4.2.1 *Any room that can be searched by a chain of three guards can be searched by a single guard with two flashlights.*

Proof. Suppose a room is searchable by a chain of 3 guards. We will convert the search schedule into a valid search schedule for a 2-searcher. The position of the 2-searcher guard corresponds to the position of the inner guard of the chain of 3. The positions of the flashlight beams correspond to the positions of the other two guards of the chain of 3. Since there is mutual visibility between adjacent guards in the chain, there is a line of sight for each of the flashlight beams to reach the positions of the corresponding guards on the edges of the room. Thus, at any point in time, the positioning of the chain of three guards in the search defines a valid configuration of the 2-searcher. Also, the beams of light can mimic the continuous motion of the outer guards. By the end of the search, the chain of 3 guards lies along a contiguous portion of the border of the polygon not containing d, which corresponds to the 2-searcher guard and its beams all lying along some portion of the boundary not containing d, indicating a complete search. So the complete search by a chain of 3 guards defines a valid search by a 2-searcher. □

Claim 4.2.2 *There exist rooms that can be searched by a 2-searcher and cannot be searched by a chain of 3 guards.*

The rest of this section will prove the claim with an example of such a room, the one in figure 4.1. The door is labelled D, and the room has five arms, where each arm contains one of the vertices A, B, C, F, and G. Bolded boundary regions s, t, and u in the diagram indicate areas that are visible from the middle of arms A, C, and B respectively, and the dotted lines indicate the areas of visibility from within arms F and G. First, we describe how a 2-searcher can search the room. The searcher starts at the door and clears up to region s and arm A with the flashlight beams (top-left of figure 4.2). To clear arm A, the searcher must move inside arm A. It keeps one beam shining at s to ensure no intruders escape through the door from the uncleared region, and the other beam is used to search the arm (top-right of figure 4.2). Note that the 2-searcher can search all parts of arm A while shining one beam at region s. Afterwards, the 2-searcher clears arms C and B in similar ways.

At this point, the beams of the 2-searcher are between B & F, and between C & G. Next, the 2-searcher will search arm F. In doing so, the flashlight beam on the right side of the room reaches back near D, but still on the right side of it to prevent intruders from leaving through it. This leaves arm C *contaminated*, as any intruders hiding in arm G could move undetected into arm C while the beam is near the door. After clearing arm F, the 2-searcher moves to and clears arm G, while the other beam is still just near D (still on the right side of it). Finally, the 2-searcher must clear arm C again, since it was contaminated. It moves towards arm C, bringing the two beams to where it is connected to the rest of the room (bottom row of 4.2), and then it searches the arm, completing the search.

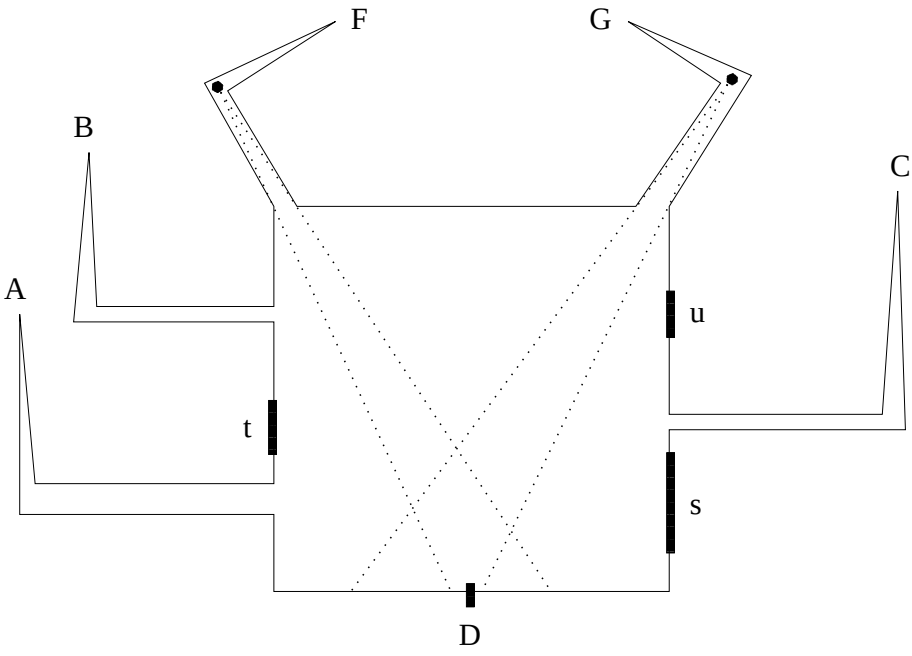

Figure 4.1: Searchable by a 2-searcher but not by three guards

We will now show that this room cannot be searched by a chain of 3 guards by showing that there is a limited amount of progress that can be made towards searching the entire room. First of all, notice that the search schedule given above for the 2-searcher cannot be converted directly into a search schedule with a chain of 3 guards, since one flashlight beam "jumps" across arm C during the search (just before arm F is searched), which an outer guard (who is restricted to moving along the boundary) cannot do. We will call the outer guard moving in the clockwise direction *guard 1*, the outer guard moving in the counter-clockwise direction *guard 3*, and the inner guard *guard 2*. We begin by showing that (except for useless repetition) the arms must be searched in the order A, C, B.

Claim 4.2.3 *Consider the events that a guard is at one of the vertices A, B, F, G, and C. In a minimum-distance search schedule for a chain of 3 guards, the order of these events must start with a guard at A, then C, and then B.*

Proof. First consider arm C. To search it first, guard 3 would need to reach vertex C, which can only see points in the outer half of arm C. Thus guard 2 needs to be in this area. Since guard 1 is on the other side of the door (if we're searching arm C first), it needs to be in region t to see guard 2. This implies arm A was already searched, contradicting the idea that we're searching arm C first. Either arm A or arm C must be searched first, since they are the first arms that can be reached when moving from the door, so arm A must be searched first. By a very similar argument, arm C must be the next one to be searched after arm A (rather than arm B). Therefore the order for the first two arms is A and then C.

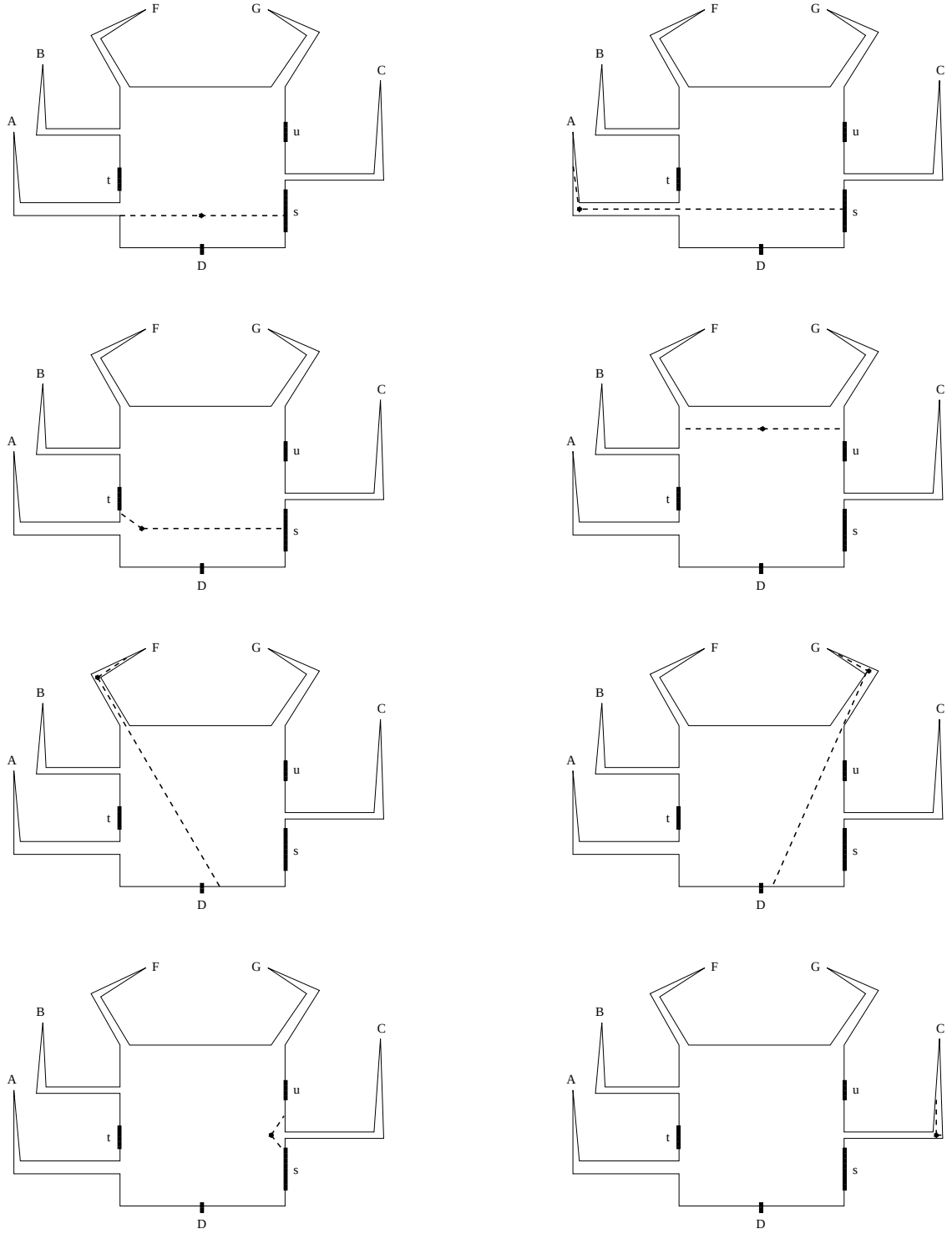

Figure 4.2: 2-searcher search schedule

At this point, either arm B or arm G must be searched next, since they are the next arms that the guards can encounter. Suppose we try to search arm G next. Guard 2 is required to go into the outer part of the arm when guard 3 hits vertex G. Since guard 1 cannot reach arm G, it is then required to go back near the door to maintain visibility with guard 2. This would require backtracking through A. But this in turn would require guard 3 to backtrack through C to get to region s, eliminating the progress made up to this point (thus making the search longer than necessary). Thus B must be the next arm to be searched. □

Claim 4.2.4 *The room in figure 4.1 cannot be searched by a chain of 3 guards.*

Proof. Note that there exists a partial search schedule that searches arms A, C, and then B. As guard 1 approaches vertex A, guard 2 enters the outer part of arm A, and guard 3 must be in region s. After finishing the search of arm A, the same basic idea is repeated for arms C and B. After searching these arms, the two outer guards are past arms B and C on the boundary of the room, leaving arms F and G to be searched.

If we try to search arm G next, we run into a problem of undoing the search of the left side of the polygon, as guard 1 must go back near the door to remain in sight of guard 2. Getting guard 1 back near the door would require guard 2 to move back into sections u and then s, hitting a previous configuration. This cannot happen in a minimum-distance search schedule. Similarly, if we try to search arm F next, we must backtrack through C to get guard 3 back near the door. This in turn requires guard 1 to go through region t, which means that arm B would no longer be searched and we would need to search it again before searching arm F. Again, we've hit a previous configuration. Therefore we cannot search arm F or arm G with a chain of three guards. □

Thus some rooms can be searched by a 2-searcher but not by a chain of 3 guards.

4.3 Link Diagram

As mentioned earlier, the link diagram used by Efrat et al. categorizes pairs of points on ∂P according to their link distance, which is one less than the number of interior guards that would be needed to create a chain of guards between them, with mutual visibility between each pair of consecutive guards [14]. For the case of 3 guards, a link distance of 1 or 2 between two points indicates that the chain of guards can meet the visibility constraints with the two outer guards at those 2 points. Any greater link distance indicates an invalid configuration, since the visibility constraints cannot be met with just 3 guards while the outer guards occupy those 2 points. We modify the link diagram by keeping only the information necessary to find a full search schedule starting at the door.

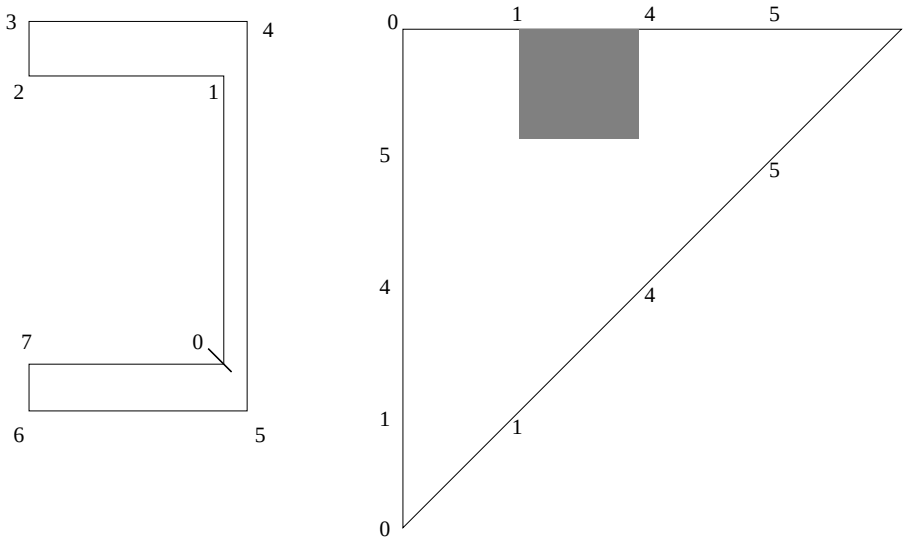

Figure 4.3: A reduced link diagram

For any room (P, d), the *reduced link diagram* encodes, for any pair of points on the boundary of P, whether or not the link distance of the two points is less than or equal to 2. It contains the minimum amount of relevant information for our room search problem. The diagram's boundary is a right triangle with the left and top sides equal in length to the perimeter of P. With the top left corner $(0, 0)$ representing the door d, the x (resp. y) coordinate represents the distance along ∂P from the door in the clockwise (resp. counterclockwise) direction. Then we can associate any point (x, y) in the diagram with positions of the two outer guards on ∂P.

To represent the feasibility of the link distance of pairs of points, any point (x, y) in the reduced link diagram is shaded iff the corresponding positions of the two guards have a link distance of 3 or more. Shaded regions make up obstacles. Note that the reduced link diagram only represents information for combinations of positions of the outer guards; valid positions for the inner guard are not represented. See Figure 4.3 for an example of a simple reduced link diagram.

4.4 Correspondence Between Paths in Reduced Link Diagram and Search Schedules

In the case of searching a room with two guards, the V-diagram is a representation of all possible configurations of the guards (i.e., it represents the configuration space of the problem). In the case of a chain of 3 guards, each of the outer guards is restricted to moving one-dimensionally along ∂P, but the inner guard may move two-dimensionally inside the polygon. Thus the configuration space for this problem is four-dimensional. As we stated earlier, we will simply search through the 2-dimensional link diagram, and we therefore must prove that a path from the door

to the goal in the modified link diagram leads to a corresponding search schedule for the chain of three guards. In particular, we will show that any continuous curve between points in the diagram corresponds to valid motion of the three guards where the two outer guards travel between pairs of corresponding points on the border of the polygon. It appears that while some of the results in [14] use this fact, they do not prove it. Also, Tan [41] proves a very similar result using a graph approach that is different from the link diagram. However, the distance information for a path through the reduced link diagram, which is vital to the algorithm here, is lost in the graph construction.

Theorem 4.4.1 *There is a path from the door to the goal in the link diagram iff there is a valid schedule for the chain of three guards to search the room. Furthermore, the correspondence is such that the L_1 length of the path in the link diagram is equal to the total distance travelled by the two outer guards in the search schedule for the chain of three guards.*

Proof. The backwards direction of the theorem is immediate because if we take a three guard schedule and ignore the middle guard, we get a path in the link diagram, since the motion of the guards is continuous through valid configurations for the outer guards. The total distance travelled by the outer guards is the L_1 length of the link diagram path.

For the other direction, we will divide the path in the link diagram into subpaths for which the motion of the middle guard can be specified.

Any continuous path can be divided into subpaths that are each monotone in both x and y. In the link diagram, this corresponds to each outer guard moving in one direction along ∂P. In any of these subpaths, a guard may be moving along multiple edges. Each of these subpaths can be further subdivided so that in each subpath, each outer guard is moving only along a single edge (in one direction). We now look at these smaller components of a path.

Lemma 4.4.2 *Given a path in the modified link diagram, consider a monotone subpath from (p, q) to (p', q') such that p and p' are on the same polygon edge and q and q' are on the same polygon edge. Then there is a valid schedule for a chain of three guards such that the outer guards travel from p to p' and from q to q' respectively, and such that the sum of the distances travelled by the outer guards is equal to the L_1 length of the subpath in the modified link diagram.*

Proof. (Our proof is constructive, leading to a definition of motion of all three guards given starting and ending points for the outer guards. This motion is used by the algorithm described later.)

Because (p, q) is a valid point in the modified link diagram there is a path π in the polygon from p to q that has link distance at most 2. Similarly, there is a path π' in the polygon from p' to q' of link distance at most 2.

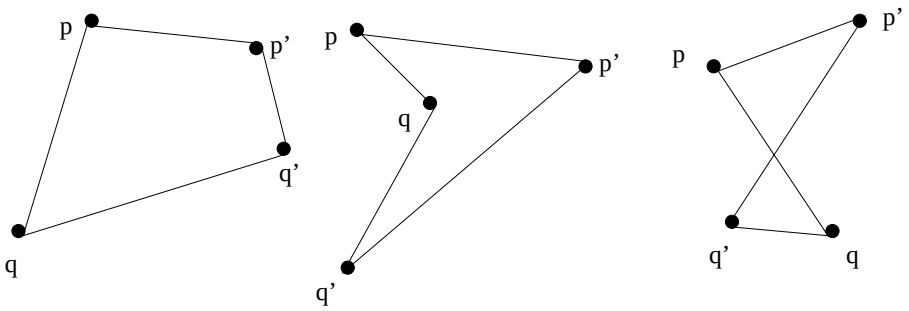

Figure 4.4: Line of sight exists between outer guards

Consider first the case where π and π' are both link distance 1 paths. We will show that there is a schedule for two guards, one travelling from p to p' and one travelling from q to q' so that they are always mutually visible. Later on we will use this as a stepping stone for the three'guard version. The line segments pq and $p'q'$ lie inside the polygon though they may cross each other. The cycle $pqq'p'$ forms a 4-gon Q that may or may not be simple, but is entirely contained in P. There are 3 cases for Q: a simple convex quadrilateral, a simple non-convex quadrilateral, and a non-simple "bow-tie shaped" 4-gon (Figure 4.4). We consider them in turn. In a convex quadrilateral, every two points can see each other; thus direct motion from p to p' and from q to q' yields a schedule where the guards can see each other at all times. In a simple, non-convex quadrilateral, exactly one of the vertices is a reflex vertex, from which all other points are visible. In this case, one of the guards must be at the reflex vertex for the entire period that the other guard travels in the region that can only see that reflex vertex (the shaded region in Figure 4.6 can see only q of line segment qq'). The remaining region is convex where both guards can always see each other, and the movement of the guards is direct to the endpoints. In the third case, where visibility lines cross, mutual visibility is always possible through that intersection point between any point on pp' and some point on qq'. Thus guard 1's movement is continuous along pp' as guard 3's motion is continuous along qq'.

Now we return to the general case where π and π' have link distance at most 2. Let m be the *bend point* on the path π that can see p and q (in case π has a single edge, let m be any interior point of it). Similarly, let m' be the bend point of π'. We will define a schedule for the three guards where guard 1 travels from p to p', guard 2 travels from m to m', and guard 3 travels from q to q'. See Figure 4.5 for a simple example. Note that pp' and qq' lie on polygon edges, and that the line segments pm, mq, $p'm'$, and $m'q'$ lie inside the polygon, though they may cross each other. The cycle $pmqq'm'p'$ forms a polygon Q that may or may not be simple but is entirely contained in P and has a well-defined inside and outside. Define the path μ for guard 2 to be the shortest path from m to m' that is entirely contained in Q.

Now consider cases where π or π' is a link distance 2 path. Take the shortest path μ from m to m' lying inside Q. If μ is a straight line then the paths of guards

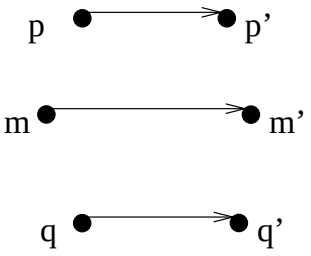

Figure 4.5: Three guards' motion

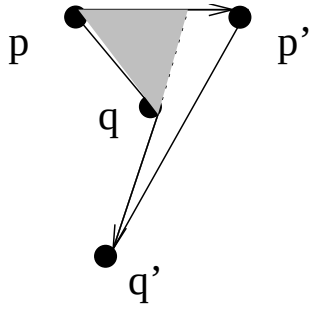

Figure 4.6: Reflex vertex in quadrilateral

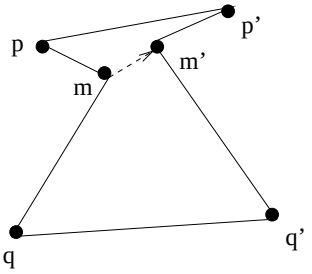

Figure 4.7: Direct path in Q from m to m'

1 and 2 form a 4-gon, and the paths of guards 2 and 3 form a 4-gon (Figure 4.7). The two guard case shows how to maintain mutual visibility between each pair of guards. For the schedules in each of these 4-gons, guard 2 travels along μ without backtracking, possibly at varying speeds. For any point along μ, the speed of guard 2 is defined separately for each of the search schedules of the 4-gons $pmm'p'$ and $qmm'q'$; at each point along μ, guard 2 will move forward at the slower of these two speeds. Each outer guard mimics its movement from its 4-gon search schedule based on its position and the position and speed of guard 2. Then, for example, at every point in time in the search of Q, the positions of guards 1 and 2 are a configuration of the guards at some point in the search of the 4-gon.

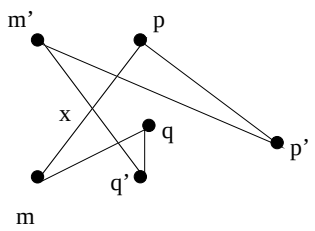

Figure 4.8: μ bends at intersection point

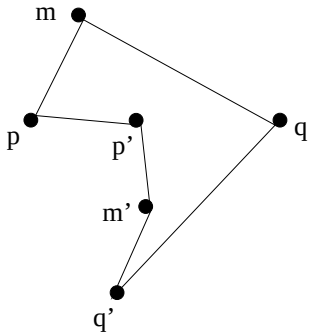

Figure 4.9: μ bends at reflex vertex

Otherwise, μ bends somewhere. First we consider the case where μ bends at an intersection point of two visibility lines, such as pm and $q'm'$ in Figure 4.8. We will label the intersection point x. Note that pp' and qq' may not intersect any edges of Q, so the edges that cross must be incident to m or m'. Also, two edges incident with the same vertex cannot cross, so each of the two edges is incident with exactly one of those vertices. Thus mx and $m'x$ lie on edges of Q, and x lies on μ. Then μ must be mxm'. We will assume, without loss of generality, that the intersecting edges are pm and $q'm'$. Then as guard 2 moves to x along mp, it maintains visibility with a guard 1 at p. Also, $mxq'q$ is a 4-gon covered by the two guard case with guard 3 moving from q to q' and guard 2 moving from m to x. Then $m'xpp'$ is also a 4-gon covered by the two guard case while guard 3 remains at q' in sight of guard 2.

The last case is when μ bends at a reflex vertex, one of p, p', q, and q'; without loss of generality, assume it's at p' (Figure 4.9). We will assume that π and π' are minimum link paths; this will restrict our choices of m and m' when constructing paths. Then μ is $mp'm'$. Consider extending the line segment $m'p'$ past p'. It starts off inside Q and exits it somewhere. It can't exit through edges $m'p'$, pp', or $q'm'$, since they share a vertex with $m'p'$. It can't exit through pm because μ bends at p' towards m', and the edge on the other side of the reflex vertex p' is p. If it exits through qq', then p' sees q', so there was a link 1 path from p' to q'. This contradicts π' being a minimum link path. This means that the extension of $m'p'$ intersects mq; call this intersection point x. Now we will define the path of guard 2 to be slightly different from μ. From m, the path goes to x, along the edge mq. Thus guard 3 can stay at q during this section. The motion of guards 1 and 2 is taken care of in the two guard case, as $p'pmx$ is a convex 4-gon. At the end of this, guard 1 is at p' and guard 2 is at x. The remainder of the search involves the case where the path for guard 2 is a straight line (from x to m') which is taken care of in an earlier case.

□

This completes the proof of Theorem 4.4.1.

□

4.5 Algorithm for Finding An Optimal Schedule

As mentioned in the introduction, due to the perceived hardness of searching for a path in the 4-dimensional configuration space, we are restricting our search for an optimal path to one which minimizes the total distance travelled by the outer guards. Based on the previous section, we can search for a path in the modified link diagram from the door to the goal line to get a search schedule for the outer guards that is part of a valid schedule for all three guards.

Efrat et al. [14] give a $O(n^3)$ time algorithm to find the link diagram. The same algorithm can be used to find the modified link diagram where the link distance is determined to be either greater than 2, or 2 or less. Then, much like in the previous chapter, the schedule that minimizes the distance travelled by the outer guards can be found by finding the shortest L_1 path from the door to the goal in the modified link diagram. Since the modified link diagram is essentially the same in terms of the information it captures, the same approach for finding the shortest path in the V-diagram will work for finding a shortest path in the modified link diagram.

One of the steps in finding the shortest path required finding the x and y extreme points of the diagram. It is not immediately obvious how to define a tight bound for the worst case number of extreme points in a link diagram. Efrat et al. [14] show that the worst case size of a link diagram is $\Theta(n^3)$, thus there are $O(n^3)$ extreme points in the worst case. This bound may be loose; further analysis may show that there are $o(n^3)$ extreme points in the worst case. After the extreme points are identified during the construction of the modified link diagram, a graph is constructed from these extreme points, along with the door point and goal line, just as in Section 3.7. The construction of the graph required only the knowledge of the extreme points of the obstacles, so the same technique is valid here. With N extreme points, it took $O(N^2)$ time, so graph construction here takes $O(n^6)$ time. Running Dijkstra's algorithm on the graph finds a shortest L_1 path from the door to the goal in the modified link diagram in $O(N^2)$, or $O(n^6)$ time.

At this point, a schedule for the outer guards minimizing the total distance they travel has been determined. In order to find a valid search schedule for all three guards, the path through the modified link diagram must be divided into subpaths, each of which corresponds to each outer guard travelling in one direction along a single edge of P. Consider only the endpoints of all of these subpaths. By Theorem 4.4.1, for each subpath, there is a schedule for all of the guards between the corresponding points in the polygon that satisfies the visibility constraints and has the outer guards travelling the distance of the L_1 distance between the endpoints. Given additional valid endpoints for the middle guard, this proof is constructive, describing a search schedule for the 3 guards between the endpoints. Thus we need to know, for a pair of positions for the outer guards, a valid position for the middle guard. This information is not captured by the link diagram, but the link diagram is constructed using an algorithm by Arkin et al. [1] that does find actual link paths between any pair of points on a polygon. Note in our proof, we assumed minimum link paths through the position of the middle guard; this algorithm finds minimum link paths, as required. After the initial $O(n^3)$ time to preprocess the polygon, each minimum link path is found in $O(\log n)$ time. Since there are at most $O(n^3)$ vertices in the link diagram, there are at most $O(n^3)$ edges in the shortest path, and thus there are at most $O(n^4)$ of these monotone, single-edge segments in the shortest path. Finding the points for the middle guard for each of these segments thus takes at most $O(n^4 \log n)$ time, and outputting the path from this point takes an additional $O(n^4 \log n)$ time at most.

Chapter 5

Conclusions and Future Work

5.1 Contributions

For the two-guard room search, no previous work has given an algorithm to find a schedule. More importantly, it seems no attempts had been made at finding an optimal schedule. We have given algorithms to find both shortest-distance and quickest search schedules for a two-guard room search, running in time $O(n^2)$ and $O(n^4)$ respectively. For the chain of guards problem, we give an $O(n^6)$ time algorithm to find a search schedule that minimizes the distance travelled by the outer guards. We also explicitly show how a 2-searcher can search more rooms than a chain of 3 guards.

More generally, the idea of finding L_1 and L_∞ shortest paths in a configuration space in order to find optimal schedules seems to be a new idea.

5.2 Future Work

There are many variants of polygon-search problems that have not been considered here. We believe optimal schedules can be found for some of these problems using the approach described here. In particular, we believe that the approach to solving shortest paths in visibility diagrams is applicable to the cases of chains of two or three guards searching a polygon without a door point and a street with an endpoint specified. This approach would require using a visibility diagram that is specialized for the type of problem to deal with the different starting or ending point situation.

Minimizing L_1 and L_∞ paths is applicable when using multiple guards, but in situations other than two guards moving one-dimensionally, the dimension of the configuration space seems to become unmanageable. Attempts to reduce the number of dimensions that need to be considered may yield reasonable running times for similar algorithms in other variants.

Additionally, the analysis of some algorithms presented here is probably not tight. Careful analysis of worst cases of the V-diagram and link-diagram may show that these algorithms are indeed faster than the worst case analysis here would suggest.

References

[1] Esther M. Arkin, Joseph S. B. Mitchell, and Subhash Suri. Optimal link path queries in a simple polygon. In *SODA '92: Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 269–279, Philadelphia, PA, USA, 1992. Society for Industrial and Applied Mathematics. 36

[2] Stephen Bahun, Binay Bhattacharya, Tsunehiko Kameda, Anna Lubiw, and John Z. Zhang. Two-guard room search: testing and scheduling. Manuscript, 2008.

[3] Stephen Bahun and Anna Lubiw. Optimal schedules for 2-guard room search. In *Proceedings of the 19th Canadian Conference on Computational Geometry (CCCG'07)*, pages 245–248, 2007.

[4] Binay Bhattacharya, John Z. Zhang, Qiaosheng Shi, and Tsunehiko Kameda. An optimal solution to room search problem. In *Proceedings of the 18th Canadian Conference on Computational Geometry (CCCG'06)*, pages 55–58, 2006. 2, 7, 11, 15, 16

[5] John F. Canny. *The Complexity of Robot Motion Planning*. MIT Press, Cambridge, MA, USA, 1988. 8

[6] Svante Carlsson and Håkan Jonsson. Computing a shortest watchman path in a simple polygon in polynomial-time. In *WADS '95: Proceedings of the 4th International Workshop on Algorithms and Data Structures*, pages 122–134, London, UK, 1995. Springer-Verlag.

[7] Danny Z. Chen, Kevin S. Klenk, and Hung-Yi T. Tu. Shortest path queries among weighted obstacles in the rectilinear plane. *SIAM Journal on Computing*, 29(4):1223–1246, 2000. 20

[8] Wei-Pand Chin and Simeon Ntafos. Shortest watchman routes in simple polygons. *Discrete Comput. Geom.*, 6(1):9–31, 1991. 1, 5

[9] V. Chvatal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory Series B*, 18:39–41, 1975. 1, 5

[10] Kenneth L. Clarkson, Sanjiv Kapoor, and Pravin M. Vaidya. Rectilinear shortest paths through polygonal obstacles in $O(n(\log n)^2)$ time. In *SCG '87: Proceedings of the Third Annual Symposium on Computational Geometry*, pages 251–257, New York, NY, USA, 1987. ACM Press. 20

[11] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, December 1959. 9

[12] David P. Dobkin and Diane L. Souvaine. Computational geometry in a curved world. *Algorithmica*, 5(3):421–457, 1990. 20

[13] Moshe Dror, Alon Efrat, Anna Lubiw, and Joseph S. B. Mitchell. Touring a sequence of polygons. In *STOC '03: Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, pages 473–482, New York, NY, USA, 2003. ACM. 1, 5

[14] Alon Efrat, Sariel Har-Peled, Leonidas J. Guibas, Joseph S. B. Mitchell, and T.M. Murali. New similarity measures between polylines with applications to morphing and polygon sweeping. *Discrete and Computational Geometry*, 28:535–569, 2002. 2, 8, 25, 26, 30, 32, 36

[15] S. Fisk. A short proof of Chvatal's watchman theorem. *Journal of Combinatorial Theory Series B*, 24:374, 1978. 5

[16] Gerald B. Folland. *Advanced Calculus*. Prentice-Hall, Inc., 2002. 10, 16

[17] Subir Ghosh. *Visibility Algorithms in the Plane*. Cambridge University Press, New York, NY, USA, 2007. 11

[18] Leonidas J. Guibas, Jean-Claude Latombe, Steven M. LaValle, David Lin, and Rajeev Motwani. A visibility-based pursuit-evasion problem. *International Journal of Computational Geometry and Applications*, 9(4-5):471–494, 1999.

[19] John Hershberger and Subhash Suri. An optimal algorithm for Euclidean shortest paths in the plane. *SIAM Journal on Computing*, 28(6):2215–2256, 1999. 11

[20] Christian Icking and Rolf Klein. The two guards problem. *International Journal of Computational Geometry and Applications*, 2(3):257–285, 1992. 7

[21] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/. 8

[22] Steven M. LaValle, Borislav H. Simov, and Giora Slutzki. An algorithm for searching a polygonal region with a flashlight. In *SCG '00: Proceedings of the Sixteenth Annual Symposium on Computational Geometry*, pages 260–269, 2000. 2, 6, 12, 13, 15

[23] D T Lee and A K Lin. Computational complexity of art gallery problems. *IEEE Trans. Inf. Theor.*, 32(2):276–282, 1986. 1, 5

[24] D. T. Lee and C. K. Wong. Voronoi diagrams in L_1 (L_∞) metrics with 2-dimensional storage applications. *SIAM Journal on Computing*, 9(1):200–211, 1980. 10

[25] Jae-Ha Lee, Sang-Min Park, and Kyung-Yong Chwa. Searching a polygonal room with one door by a 1-searcher. *International Journal of Computational Geometry and Applications*, 10(2):201–220, 2000. 1, 2, 7, 14

[26] Jae-Ha Lee, Sung Yong Shin, and Kyung-Yong Chwa. Visibility-based pursuit-evasion in a polygonal room with a door. In *SCG '99: Proceedings of the Fifteenth Annual Symposium on Computational Geometry*, pages 281–290, New York, NY, USA, 1999. ACM.

[27] Elefterios A. Melissaratos and Diane L. Souvaine. Shortest paths help solve geometric optimization problems on planar regions. *SIAM Journal on Computing*, 21(4):601–638, 1992. 20

[28] Joseph S. B. Mitchell. L_1 shortest paths among polygonal obstacles in the plane. *Algorithmica*, 8(1):55–88, 1992. 20

[29] Joseph S. B. Mitchell. Geometric shortest paths and network optimization. In *Handbook of Computational Geometry*, pages 633–701. Elsevier Science Publishers B.V. North-Holland, 2000. 10

[30] Joseph S. B. Mitchell, David M. Mount, and Christos H. Papadimitriou. The discrete geodesic problem. *SIAM Journal on Computing*, 16(4):647–668, 1987. 11

[31] Joseph S. B. Mitchell and Micha Sharir. New results on shortest paths in three dimensions. In *SCG '04: Proceedings of the Twentieth Annual Symposium on Computational Geometry*, pages 124–133, New York, NY, USA, 2004. ACM. 25

[32] Joseph O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, Inc., New York, NY, USA, 1987. 1

[33] Sang-Min Park, Kyung-Yong Chwa, and Jae-Ha Lee. A characterization of the class of polygons searchable by a 1-searcher. In *Proceedings of the 17th European Workshop on Computational Geometry*, pages 133–136, 2001. 6

[34] Sang-Min Park, Jae-Ha Lee, and Kyung-Yong Chwa. Visibility-based pursuit-evasion in a polygonal region by a searcher. *Lecture Notes in Computer Science*, 2076:456–468, 2001. 6

[35] Sang-Min Park, Jae-Ha Lee, and Kyung-Yong Chwa. Searching a room by two guards. *International Journal of Computational Geometry and Applications*, 12(4):339–352, 2002. 2, 7, 8, 13

[36] Michel Pocchiola and Gert Vegter. Computing the visibility graph via pseudo-triangulations. In *SCG '95: Proceedings of the Eleventh Annual Symposium on Computational Geometry*, pages 248–257, New York, NY, USA, 1995. ACM. 11

[37] Michel Pocchiola and Gert Vegter. Topologically sweeping visibility complexes via pseudotriangulations. *Discrete & Computational Geometry*, 16(4):419–453, 1996. 11

[38] Stéphane Rivière. Topologically sweeping the visibility complex of polygonal scenes. In *SCG '95: Proceedings of the Eleventh Annual Symposium on Computational Geometry*, pages 436–437, New York, NY, USA, 1995. ACM. 11

[39] Ichiro Suzuki and Masafumi Yamashita. Searching for a mobile intruder in a polygonal region. *SIAM Journal on Computing*, 21(5):863–888, 1992. 1, 6, 13

[40] Xuehou Tan. The two-guard problem revisited and its generalization. *Lecture Notes in Computer Science*, 3341:847–858, 2004. 8

[41] Xuehou Tan. Sweeping simple polygons with the minimum number of chain guards. *Information Processing Letters*, 102(2-3):66–71, 2007. 8, 26, 32

[42] Csaba D. Tóth. Art gallery problem with guards whose range of vision is 180°. *Comput. Geom. Theory Appl.*, 17(3-4):121–134, 2000. 5

[43] Csaba D. Tóth. Art galleries with guards of uniform range of vision. *Comput. Geom. Theory Appl.*, 21(3):185–192, 2002. 5

[44] Csaba D. Tóth. Illumination of polygons by 45°-floodlights. *Discrete Math.*, 265(1-3):251–260, 2003. 5

[45] L. H. Tseng, Paul J. Heffernan, and D. T. Lee. Two-guard walkability of simple polygons. *International Journal of Computational Geometry and Applications*, 8(1):85–116, 1998. 11

[46] Zhong Zhang. *Applications of Visibility Space in Polygon Search Problems*. PhD thesis, Simon Fraser University, August 2005. 12, 13, 14, 15, 26