The Visibility Freeze-Tag Problem

by

Yizhe Zeng

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, 2014

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 Freeze-Tag Problem, we are given a set of robots at points inside some metric space. Initially, all the robots are frozen except one. That robot can awaken (or "unfreeze") another robot by moving to its position, and once a robot is awakened, it can move and help to awaken other robots. The goal is to awaken all the robots in the shortest time. The Freeze-Tag Problem has been studied in different metric spaces: graphs and Euclidean spaces.

In this thesis, we look at the Freeze-Tag Problem in polygons, and we introduce the *Visibility Freeze-Tag Problem*, where one robot can awaken another robot by "seeing" it. Furthermore, we introduce a variant of the Visibility Freeze-Tag Problem, called the *Line/Point Freeze Tag Problem*, where each robot lies on an awakening line, and one robot can awaken another robot by touching its awakening line.

We survey the current results for the Freeze-Tag Problem in graphs, Euclidean spaces and polygons. Since the Visibility Freeze-Tag Problem bears some resemblance to the Watchman Route Problem, we also survey the background literature on the Watchman Route Problem. We show that the Freeze-Tag Problem in polygons and the Visibility Freeze-Tag Problem are NP-hard, and we present an $O(n)$-approximation algorithm for the Visibility Freeze-Tag Problem. For the Line/Point Freeze-Tag Problem, we give a polynomial time algorithm for the special case where all the awakening lines are parallel to each other. We prove that the general case is NP-hard, and we present an $O(1)$-approximation algorithm.

Acknowledgements

I would like to thank my supervisor, Anna Lubiw, for her constant encouragement, patience, and immense knowledge. Without her guidance and understanding, it is impossible for me to finish this thesis.

I also want to thank my parents for their love and support. They encouraged and supported me to go abroad seven years ago, and that is where everything started.

Table of Contents

List of Figures

Chapter 1

Introduction

Freeze Tag or "Stuck in the Mud"[1] is a children's game where players (or "robots') who are tagged as "frozen" may not move until they are unfrozen by being touched by another player. With this idea, Arkin et al. [3] proposed the Freeze-Tag Problem: There are n frozen players (or "sleeping robots") inside a domain $\mathcal{D}$, and initially only one player is unfrozen (or "awake"). An awake player may move to another player to unfreeze it. The goal is to unfreeze all the players as soon as possible. Once a robot is unfrozen, it can help to unfreeze other players.

The results for the Freeze-Tag problem (FTP) vary by the domain $\mathcal{D}$. Graphs and the Euclidean plane are two major spaces being studied. For the graph version, Arkin et al. [3] show that the FTP is NP-hard even for star graphs, and they also prove that it is NP-hard to approximate the Freeze-Tag Problem within a factor less than $\frac{5}{3}$, even for a weighted graph with maximum degree 4. For the FTP in the Euclidean plane, Arkin et al. give a Polynomial-Time Approximation Scheme (PTAS) such that for a parameter $\epsilon > 0$, their algorithm yields an approximation ratio of $(1+\epsilon)$ with running time $O(2^{poly(1/\epsilon)}+n \log n)$ for n frozen robots. It is also worth noting that the complexity of the FTP in the plane remains unknown. Arkin et al. conjectured that the FTP is NP-hard in the plane for Euclidean and Manhattan distance metrics. One property that makes this problem particularly intriguing is that any non-lazy algorithm (and in particular the greedy algorithm) gives an approximation factor of $O(n \log n)$. However, no approximation algorithm better than greedy has been found for graphs and the Euclidean space.

Arkin et al.[3] discuss an application of the FTP to data distributing where transmission requires physical proximity because wireless communication could either be bandwidth

[1] http://en.wikipedia.org/wiki/Freeze_tag/Freeze_tag

consuming or unsecured. The problem also has applications in broadcasting, where an urgent message is expected to be broadcasted to a group of people as soon as possible by only allowing communication between two individuals. The Freeze-Tag Problem also has some applications in the areas of routing, scheduling, and network design.

The Freeze-Tag Problem is a natural variant of the Traveling Salesman Problem, where after a salesman reaches a recipient, the recipient becomes a new salesman who can help with further deliveries. Thus, the Freeze-Tag problem is also called the "parallel" version of the Traveling Salesman Problem and is studied as a variant of the Cooperative Traveling Salesman Problem [6].

A natural extension of the FTP in the Euclidean plane is to ask what if all the robots are inside a polygon. This setting is more practical since it simulates the situation where the robots are inside a room. Note that the FTP in the Euclidean plane is a special case of this problem, where all the robots are inside a large bounding box.

For a further extension to the FTP in polygons, we allow one player to unfreeze another player if it is able to "see" the other player inside the polygon, i.e., the line segment between them is inside the polygon. We named this problem the Visibility Freeze-Tag Problem (VFTP). This problem has also been called the Laser-Tag Problem by Joseph S.B. Mitchell[2].

Finally, we study a version of the Freeze-Tag Problem where each player lies on a line, and is unfrozen when another player touches the line. We named this version the Line/Point Freeze-Tag Problem.

This thesis is organized as follows.

In Chapter 2, we will take a closer look at some variants of the problems discussed above. In addition, we will give the preliminary background for understanding the discussion in the following thesis.

In Chapter 3, we will show that the FTP in a polygon and the VFTP are NP-hard by reduction from the Freeze-Tag problem in star graphs. We will also give an $O(n)$-approximation algorithm for the VFTP.

In Chapter 4, we will look at the Line/Point version of the FTP. We provide a polynomial time algorithm for the case where all the awakening lines are parallel to each other. In addition, we show this problem is NP-hard, and give an algorithm that achieves constant approximation ratio for the general case.

[2]private communication, June 25, 2014

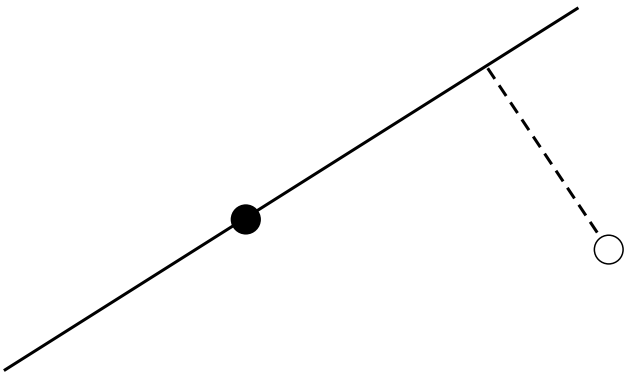

Figure 1.1: An example of the Line/Point FTP. In order to wake up the robot positioned at the solid circle, the shortest path for the robot at the empty circle is to move along the dashed line.

Chapter 2

Background Information

In this chapter, we provide necessary preliminaries for understanding the rest of the thesis. In Section 2.1, we will first provide more details of the background for the Freeze-Tag Problem, and as this problem is defined on different domains, we will survey the results for each domain. In Section 2.2, we will look at the background for the Visibility Freeze-Tag Problem and its variants including the Watchman Route Problem and the Line/Point Freeze-Tag Problem.

2.1 The Freeze-Tag Problem

In an instance of the Freeze-Tag Problem, we are given a set of robots $R = \{s_0, s_1, s_2, ..., s_n\} \subset \mathcal{D}$ in a domain $\mathcal{D}$. All the robots in R are frozen initially except s_0. An active robot u can move to a frozen robot v's position to awaken it. Once a frozen robot becomes active, it can help to unfreeze other robots. Let $d(u, v)$ be the distance u needs to move to awaken v, the measurement of $d(u, v)$ depends on the domain $\mathcal{D}$.

- The domain $\mathcal{D}$ is a weighted graph $G = (V, E)$, with non-negative edge weights. Then $d(u, v)$ is the sum of edge weights along the path robot u travelled to awaken v.
- The domain $\mathcal{D}$ is a d-dimensional Euclidean space. Then $d(u, v)$ is the geodesic distance robot u travelled to awaken v.

A feasible solution to the Freeze-Tag Problem can be described as a weighted tree T, which spanning R and rooted at the initial active robot where every internal node has at

most 2 children. If robot v is awakened robot by robot u, then the two children of v are the robots awakened next by v and u. In addition, the weight of an edge (u, v) in T is the distance u travelled to awaken v. Arkin et al. [3] call such a tree an *awakening tree*. See Figure 2.1 for an example of the awakening tree. The goal of the Freeze-Tag Problem is to

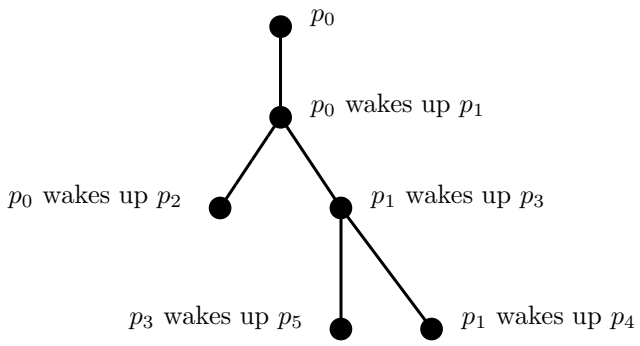

Figure 2.1: An example of the awakening procedure describing as a tree.

For any robot u in T, the *awakening path* of u is the unique path that connects the initially active robot s_0 to u in T. An awakening path $P = \{s_0, s_1, \ldots, s_k\}$ represents an awakening schedule such that s_i awakens s_{i+1} for $i < k$. The objective of the Freeze-Tag Problem is to find an awakening tree T^*, such that the depth of T^* is minimized, i.e., the longest root to leaf path is minimized.

As mentioned in Chapter 1, most of the current results are for the Freeze-Tag Problem in graphs and Euclidean spaces. We will fill in more details for these problems in this section.

One of the most intriguing properties of the FTP is that any algorithm that satisfies the following two properties yields an approximation ratio of $O(\log n)$ in any domain $\mathcal{D}$ that satisfied the symmetry property, i.e., for any two robots u and v, we have $d(u, v) = d(v, u)$. (Proposition 1.1 of [3]) An algorithm that satisfies these properties is called "non-lazy".

- Once a robot becomes active, it immediately claims and travels to an unclaimed frozen robot's position.
- If there are no unclaimed robots left, then a robot that becomes active will stay where it is.

In the greedy algorithm, when a robot becomes active it claims and travels to the closest unclaimed robot (if there is one). Since the greedy algorithm satisfies the above two properties, it also yields an approximation ratio of $O(\log n)$ for any domain $\mathcal{D}$ that satisfied the symmetry property.

2.1.1 FTP on Graphs

The graph version of the FTP is defined in a weighted graph $G = (V, E)$, where each vertex contains zero or more robots, and each edge is associated with a non-negative weight. The robots on a vertex can be awakened by another robot s traveling to that vertex. The time taken by robot s is the sum of the edge weights along the path it follows. The FTP on general graphs has been proved to be NP-hard by Sztainberg et al. [29]. Furthermore, this problem has been proved to be NP-hard even for some special graph classes such as trees and star graphs with at most one robot at each vertex [3].

No effective polynomial time algorithm is known so far for the Freeze-Tag Problem in general graphs. The only polynomial time exact algorithm that has been found is for unweighed star graphs. Arkin et al. [3] showed that the greedy algorithm gives an optimal solution for the Freeze-Tag Problem in unweighted star graphs.

Since the Freeze-Tag Problem in general graphs is NP-hard, people are interested in looking for approximation algorithms. However, the only algorithm known to have a better approximation ratio than $O(\log n)$ on general graphs is the greedy algorithm. Konemann et al. [24] showed that the greedy algorithm achieves an approximation ratio of $O(\sqrt{\log n})$.

Approaches other than the greedy algorithm have been attempted. Arkin et al. [4] gave a "Density-Based" approximation algorithm which yields an approximation ratio of $O(\log n^{\log(\frac{5}{3})})$ for unweighed graphs. The algorithmic dilemma for the Freeze-Tag Problem is that an awakened robot needs to decide whether it should wake up a nearby robot cluster with a smaller number of robots as helpers or a large cluster of robots for more helpers that is further away [3]. The density-based algorithm tackles this dilemma by dividing the graph into smaller components, and first waking up the "densest" component, i.e., the component that contains the maximum number of robots, then sending those active robots to awaken the robots in the rest of the components. Note that they only consider the case where there is exactly one robot at each vertex.

Arkin et al. [3] also investigated lower bounds on the approximation ratio for the FTP. They showed that it is impossible to have an approximation within a factor of $\frac{5}{3}$ in general weighted graphs unless $P = NP$. However, they showed that it is possible to have a better

approximation ratio for certain graph classes. In the same paper, they gave a PTAS for the Freeze-Tag Problem in weighted star graphs where initially the active robot is at the center and there are equal numbers of robots at each leaf vertex.

It is also natural to look at the the online version of the Freeze-Tag Problem, where each robot can only see its neighbourhood. Arkin et al. [3] gave an $O(\log \Delta)$-competitive online algorithm for locally bounded edge-weighted graphs with maximum degree Δ.

2.1.2 FTP in Euclidean Space

The complexity of the FTP is open for Euclidean spaces, even for the Euclidean plane. Arkin et al. [3] conjectured that the problem is NP-hard. In addition, they gave a PTAS for the Freeze-Tag Problem in any fixed dimension Euclidean space. Their algorithm finds a solution to the Freeze-Tag Problem within a factor of $1 + \epsilon$ of the optimal solution, and takes time $O(2^{(m^2 \log m)} + n \log n)$, where $m = 1/\epsilon$.

The most natural algorithm for the Freeze-Tag Problem in a Euclidean space is the greedy algorithm where each awakened robot claims and moves towards the nearest unclaimed robot. Sztainberg et al. [29] proved that the greedy algorithm is a 4-approximation in 1-D and $O(\sqrt{\log n})$-approximation in 2-D. Generally, for d-dimensional Euclidean space, they showed that the greedy algorithm achieves an approximation ratio of $O(\log n^{1-\frac{1}{d}})$ for $d \geq 2$.

Arkin et al. gave an $O(1)$-approximation algorithm as the first step for the PTAS in [3]. Later, Arkin et al. [4] gave a density-based $(3 + o(1))$-approximation algorithm for the FTP in Euclidean spaces as an improvement. For the FTP in the Euclidean plane, the algorithm first bounds all the robots with a $d \times d$ square, and divides the $d \times d$ square into $\sqrt{n}$ equal size squares of size $d/n^{1/4} \times d/n^{1/4}$. The initial active robot is sent to wake up all the robots in the square with the largest number of robots. Finally, all the robots from that square are distributed into the rest of the squares to wake up the rest of the robots.

2.1.3 FTP in Polygons

The Freeze-Tag Problem in polygons was first raised by Arkin et al. as an open problem in [3]. For this problem, note that the robots inside a polygon still satisfies the symmetry property, thus any non-lazy algorithm gives an approximation ratio of $O(\log n)$. However, the PTAS in [3] for the FTP in the Euclidean plane no longer applies to this problem.

There has been increased research interest in controlling robot swarms in a cooperative manner, and since a polygon is a good model of an indoor environment, a lot of research has been done to study robot swarms inside a polygon. An interesting and practical problem is to place robots in a polygon to form a communication network [12]. Some other variations of movement problems have also been studied by Demaine et al. [13].

2.2 The Visibility Freeze-Tag Problem

A *simple polygon* P is a polygon that exactly two edges meet at each vertex. Let $R = \{s_0, s_1, s_2, ..., s_n\}$ be a set of points (or "robots") positioned inside P. All the robots in R are frozen initially except s_0. An unfrozen (or "active") robot u can awaken another frozen robot v if the line segment uv remains inside the polygon P. Once a frozen robot becomes active, it can help to unfreeze other robots. The goal is to unfreeze all the robots in R in the minimum time. The VFTP resembles the idea of the Watchman Route Problem, which can be considered as a variant of the VFTP where only the initial robot has the ability to wake up other robots. We will look at the Watchman Tour Problem in Section 2.2.1 and an interesting variant of the VFTP in Section 2.2.2.

2.2.1 The Watchman Tour Problem

Given a polygon P and a point s inside P that represents the "watchman", the *Watchman Route Problem* (WRP) asks for the minimum length path for s to travel inside P so that every point inside P is visible from some position of s along the path. Clearly the WRP provides an upper bound for the VFTP, as it would be sufficient to wake up all the robots if the initial robot travels along a path that "sees" every point inside the polygon. Based on whether an "anchor" point has been provided which the watchman's route must go through, there are two versions of the WRP: If the the watchman's route must go through a specified anchor point, this version is called the "anchored" WRP. If no such point is required, then the problem is called the "floating" WRP.

The WRP was first introduced as a variant of the Art Gallery Problem by Chin and Ntafos [10]. The difference between the WRP and the Art Gallery Problem is that instead of having static guards watching over the area, now we have a single guard which is granted mobility. It can be observed that the WRP shares some characteristics of the TSP, which also asks for a shortest tour. Both the TSP and the Art Gallery Problem are NP-hard. However, both versions of the WRP can be solved in polynomial time in simple polygons.

Chin and Ntafos also gave a NP-hardness proof for the the WRP in polygons with holes when they introduced the problem. Later, Dumitrescu and Tóth [16] pointed out some flaws in Chin and Ntafos's proof, and gave another proof to show that the problem is indeed NP-hard in polygons with holes.

Chin and Ntafos gave the first algorithm for solving the WRP in simple polygons. Their algorithm takes time $O(n^4 \log \log n)$ for the anchored WRP. The current best algorithm is found by Dror et al. [15]. They approached this problem as the touring polygons problem, where the the order of visiting is determined by the polygon, and they improved the running time to $O(n^3 \log n)$ for the anchored WRP and $O(n^4 \log n)$ for the floating WRP.

An interesting variant of the Visibility Freeze-Tag Problem is when only the initial robot is granted the ability to awaken the other robots. It is not hard to see that the problem can be solved with the touring polygons technique. Take the example from Fig 2.2, where initially only v_0 is awake. We can easily convert it to an anchored Watchman Route Problem starting at position v_0.

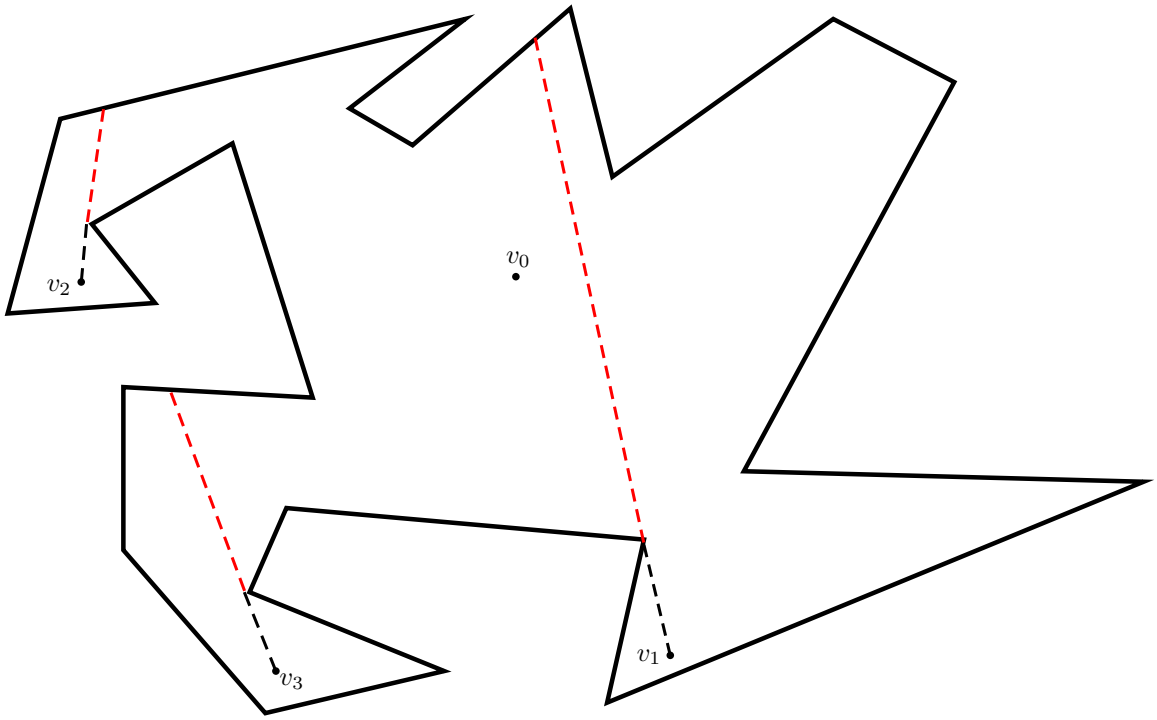

Figure 2.2: The Visibility Freeze-Tag Problem where only the initial robot can awaken others can be modelled as a Touring Polygons Problem. In the example shown here, v_0 has to visit the dashed red lines to see v_1, v_2 and v_3.

Since the current best solution for the WRP still takes time $O(n^3 \log n)$, some work has been done on finding approximation algorithms for this problem. The goal is to find a linear time algorithm that provides a constant approximation ratio. Carlsson [9] first provided a linear time algorithm which yields an approximation ratio of 98.9823 for the anchored WRP. Later, Tan greatly improved the ratio to $\sqrt{2}$ for the anchored version and 2 for the floating version in simple polygons [30]. Very little has been known about the approximability for the WRP in a polygon with holes until recently Mitchell [27] gave an $O(\log^2 n)$-approximation algorithm.

Another variant of the WRP is the Multi-Watchman Route Problem (MWRP). Instead of having a single watchman, there are multiple watchmen who can each travel on their own path. The goal of the MWRP could be minimizing the total travel length of all the watchman, or minimizing the maximum travel length made by any watchmen. Unfortunately, both versions have been shown to be NP-hard by Packer [28]. Packer also showed

that neither problem has an approximation algorithm which achieves a polynomial approximation ratio unless $P = NP$. The MWRP could be treated as another variant of the Freeze-Tag Problem in polygons where a few active robots are given and only those robots are able to awaken others.

Besides increasing the number of watchman, another variant of the WRP asks to optimize a different objective function. This variant was first introduced by Alsuwaiyel and Lee [2]. It asks for a watchman tour that minimizes the number of line segments of the tour, i.e., that minimizes the link length of the tour. Thus, the problem is named the "Minimum Link Watchman Tour Problem". In the same paper, the problem is proven to be NP-hard in simple polygons. Later, Arkin et al. [5] showed that this problem is NP-hard in a rectangle with convex holes. In addition, they provided an $O(\log n)$-approximation algorithm.

A natural variant of the WRP is where the watchman does not have any information about the polygon it must explore. This is the online version of the WRP, which is also known as the polygon exploration problem. This variant has many real life applications in robot navigation. It is inevitable that the touring path in the online setting will be longer than the shortest watchman path in the offline setting. Deng et al. [14] first proved the existence of an online strategy which achieves a constant competitive ratio . Hoffman et al. [20] provided an online strategy which is 133-competitive for simple polygons. Later, they improved their algorithm by reducing the competitive ratio to 26.5 [21]. This is the best online strategy known so far. For the lower bound on the competitive ratio of any online algorithm, Hagius et al. [19] proved a lower bound of 1.2825 for simple polygons. Note that there is still a large gap between the lower bound of 1.2825 and the upper bound of 26.5 for the polygon exploration problem in simple polygons. In addition, no effective algorithms has yet been found for exploring polygons with holes.

2.2.2 The Line/Point Freeze-Tag Problem

Recall that in the Visibility Freeze-Tag Problem, a robot u can unfreeze another robot v if and only if they can see each other. If robot v is not visible to the awake robot u then there is a line that goes through v and a reflex vertex and u must reach that line in order to see v. This line is called the "window" of v. See the red lines in Figure 2.2 for examples.

Motivated by this, we propose a variant of the Visibility Freeze-Tag Problem where every frozen robot v has an associated "awakening line" l_v and v is awakened when another robot touches its awakening line. We call this the "Line/Point Freeze-Tag Problem". Note that the points and lines are in the plane with no surrounding polygon. We hope that the

Line/Point Freeze-Tag Problem gives us better understanding for the Visibility Freeze-Tag Problem.

There is a close relationship between the Line/Point Freeze-Tag Problem and the Travelling Salesman Problem for lines in the plane. In particular, consider the special case of the Line/Point Freeze Tag Problem where all the frozen robots are infinitely far away along their awakening lines. In this case the initial awake robot must visit all the awakening lines by itself. This is the TSP for lines in the plane, which can be solved in time $O(n^5)$ [17].

Chapter 3

The Visibility Freeze-Tag Problem

3.1 Introduction

This chapter is about the Visibility Freeze-Tag Problem (VFTP) in a polygon which was defined in Section 2.2. In Section 3.2, we look at the relationship between the Visibility Freeze-Tag Problem in a polygon and the original Freeze-Tag Problem in a polygon. Then we show that both problems are NP-hard in Section 3.3. In Section 3.4, we provide an algorithm in the VFTP setting to find the shortest path for waking up a sleeping robot with a given initial active robot. Finally, in Section 3.5, we give an $O(n)$-approximation algorithm for the Visibility Freeze-Tag Problem.

3.2 VFTP vs. FTP

One of the major differences between the FTP and the VFTP in a polygon is that distances are no longer symmetric, i.e., given two robots A and B, the length of the shortest path for A to wake up B does not have to equal the length of the shortest path for B to wake up A. See Figure 3.1 for an example.

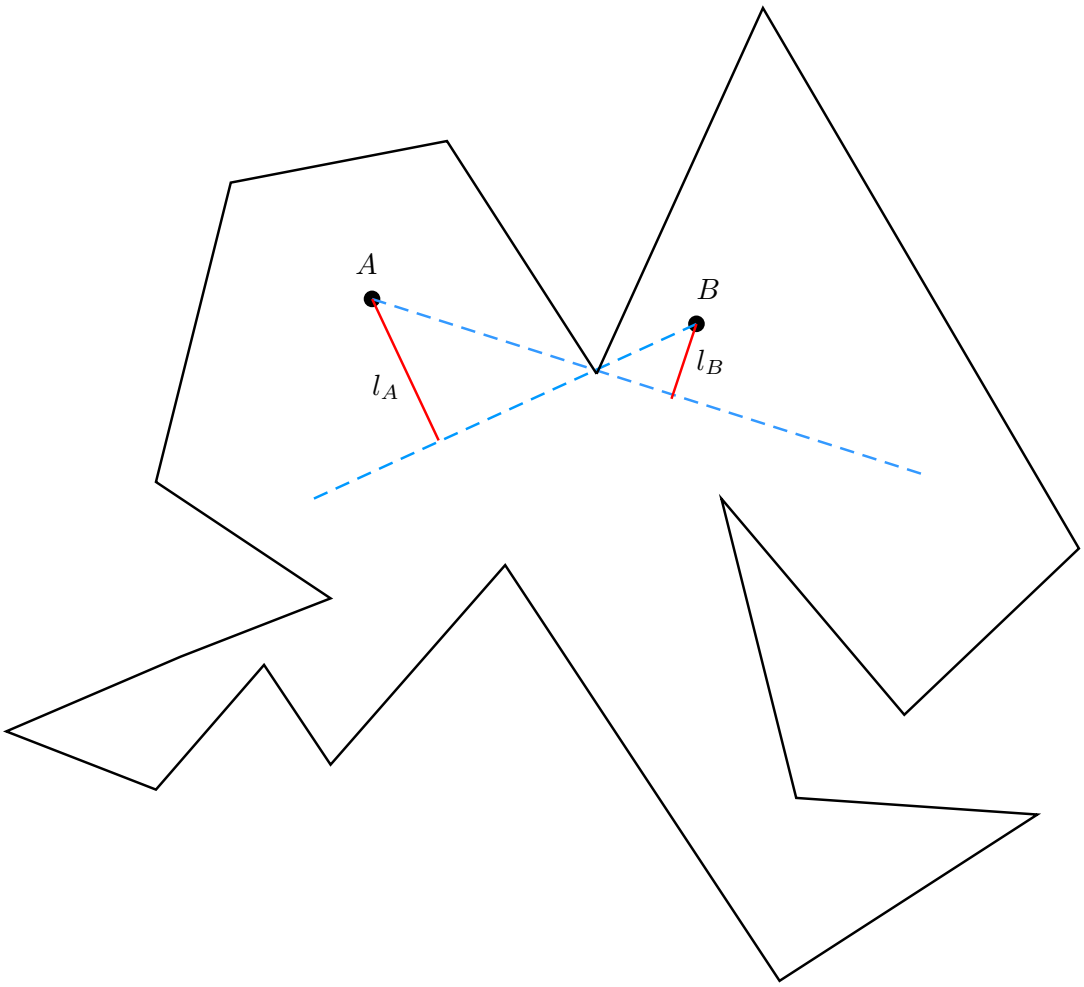

Figure 3.1: This is an example to show the asymmetric property of the VFTP. The distance that A needs to travel to wake up B, l_A, is greater than the distance that B needs to travel to wake up A, l_B.

3.3 NP-Hardness of the VFTP and the FTP in Polygons

In this section, we will show that the Freeze-Tag Problem in polygons is NP-hard by reduction from the Freeze-Tag Problem for star graphs. The same reduction will show

that the Visibility Freeze-Tag Problem in polygons is NP-hard.

Arkin et al. [3] proved, using a reduction from 3-PARTITION, that the FTP is NP-hard even for a star (a tree where all the nodes except one are leaves):

Instance: A star graph with $n-1$ leaves, an integer L, and non-negative integer edge weights $w_1, w_2, w_3, ..., w_{n-1}$ for each edge of the star. Initially there is a frozen robot at each leaf node of the star, and an active robot at the center node.
Question: Is there an awakening schedule such that all the robots in the graph can be awakened before time L?

Theorem 3.3.1. *The Freeze-Tag Problem in polygons is NP-hard.*

Proof. We will reduce the FTP in a star to the FTP in a polygon. Suppose we have an instance of the FTP in a star (as described above). We first start with a *base* rectangle with width $n-1$ and arbitrary height $\epsilon > 0$, and place the initial active robot v_0 at the upper left corner of the base rectangle. For each edge weight w_i, we add a *spike* rectangle on top of the base rectangle with height $n^2 w_i$ and width $1/2$. In addition, the length of the gap between two spikes is also $1/2$ (Figure 3.3). We place one robot at the top-right corner of each spike. It is easy to see that the construction can be done in polynomial time.

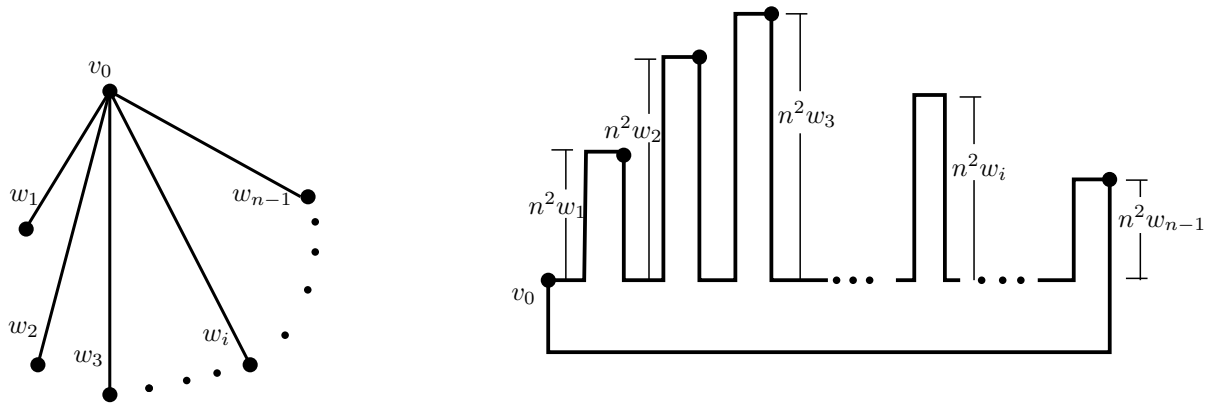

Figure 3.2: Converting a star graph to a polygon

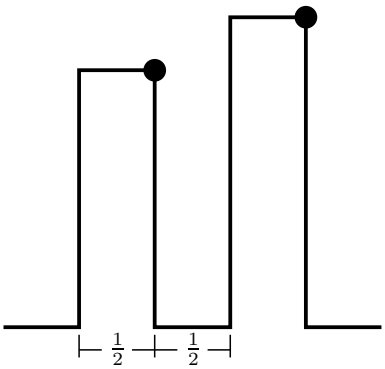

Figure 3.3: Horizontal distance between two holes is $\frac{1}{2}$

We claim that there is an awakening schedule in time $n^2L+(n-1)^2$ for the FTP in the constructed polygon if and only if there is an awakening schedule in time L for the FTP in the star.

For the proof of the "if" part, suppose that there is an awakening schedule in time L for the star graph. We want to show that by following the same awakening schedule, all the robots can be awakened by time $n^2L+(n-1)^2$ in the FTP in the polygon. Let v be a robot other than v_0 in the star version of the Freeze-Tag problem. Then, in a feasible solution to the Freeze-Tag Problem, there exists an awakening path for v: $v_0, v_1, v_2, ..., v_k = v$. Since there are n robots in total, thus $k \leq n-1$. In addition, note that a robot needs to travel at most $n-1$ horizontally to wake up another robot. This implies that the total horizontal movement is at most $(n-1)^2$. Thus, by following the same awakening schedule, all the robots could be awakened by the time limit even if all the robots can only move horizontally and vertically. This completes one direction of our proof.

Conversely, suppose that there is an awakening schedule in time $n^2L+(n-1)^2$ for the FTP in the constructed polygon. We want to show that by following the same awakening schedule, all the robots can be awakened by time L in the FTP on the star. If we ignore the horizontal component of all the robot moves, this does not increase the total movements. Therefore the sum of the vertical motions along any path of the awakening tree is at most $n^2L+(n-1)^2$. Note that the vertical distance a robot travels inside the polygon is equal to the distance the corresponding robot travels in the star times n^2. Since all the edge weights in the star are integers, the sum of distance of the robots travel vertically in the polygon must be a multiple of n^2. The largest multiple of n^2 less than $n^2L+(n-1)^2$ is n^2L. This implies that by following the same awakening schedule, all the robots in the star graph can be awakened no later than time L. □

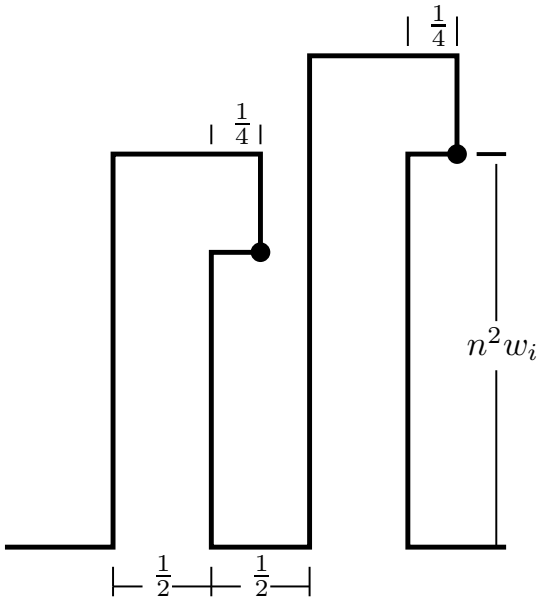

Figure 3.4: Modified spikes for the VFTP NP-hardness Proof

We can make a slight modification to the proof to show that the Visibility Freeze-Tag Problem is NP-hard. As shown in Figure 3.4, we add a small horizontal spike at the top of each original spike, and we place one robot in the bottom-right corner of each new spike. It is obvious that the construction can be done in polynomial time. In this case, the horizontal distance v_0 needs to move to wake up the furthest robot remains $n-1$, and the longest distance for a robot other than v_0 to wake up another robot is at most $n-2+1/4 = n-7/4$, since the robot needs to travel distance $1/4$ to move out of the spike, and at most distance $n-2$ to reach the other robot. Thus, the horizontal movement is still bounded by $(n-1)^2$. By the same argument, we have the following corollary:

Corollary 3.3.1. *The Visibility Freeze-Tag Problem is NP-hard.*

3.4 Shortest Awakening Paths

For the standard Freeze-Tag Problem on points in the plane we know exactly how to make an awake robot s awaken another robot t—robot s should just travel in a straight line to t. There is never an advantage in using an intermediate robot if the goal is just to awaken

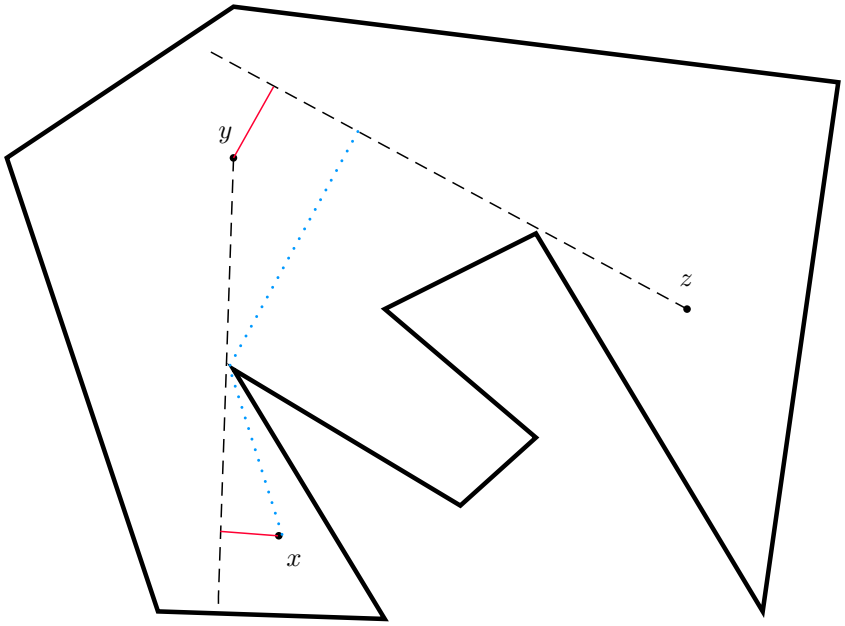

Figure 3.5: The shortest path for robot x to see z without intermediate robots (shown in dotted blue) is much longer than the shortest path (shown in solid red) with the help of the intermediate robot y.

t as quickly as possible. This part is easy and the difficult part of the Freeze-Tag Problem is deciding the the order in which robots awaken other robots.

However, in the Visibility Freeze Tag Problem, the basic question of how to make an awake robot s awaken another robot t is not so trivial. It may help to use intermediate robots. See Figure 3.5 for an example.

In this subsection we address this problem of finding a shortest awakening path from one robot s to another. In the first step s wakes up some robot s_1. Then s_1 wakes up s_2 and so on. Each step of the path consists of one robot u starting from its initial position and moving to wake up another robot v (without using any other robots). We can find such a path using a shortest path algorithm on points R where for edge (u, v) we define its weight $w(u, v)$ to be the length of a shortest path for u to awaken v without using any intermediate robots. In other words, $w(u, v)$ is the length of a shortest path inside polygon P from u to a point z that is visible from v. In particular, if u and v are visible in P then $w(u, v) = 0$.

A single source shortest path algorithm (such as Dijkstra's algorithm) using the weights

$w(u, v)$ will find a shortest path tree that gives the shortest awakening path from the initial active robot s to every other robot t. We call this the *shortest awakening path tree*. The shortest awakening path tree does not provide a solution to the VFTP because, e.g., the root s may have several children in the tree, and s cannot wake them all up at once. However, the shortest awakening path tree does give us a lower bound on the amount of time it takes for each robot t to be awakened.

We will give an algorithm to find the shortest awakening path tree in time $O(n^2 \log n)$ and space $O(n)$ where n is the total input size, i.e., $n = n_P + n_R$ where n_P is the number of vertices of P and n_R is the number of points in R. We will first show how to achieve the time bound, which is straight-forward based on known results, and then show how to modify this to achieve the space bound.

The straight-forward method for finding the shortest awakening path tree is to find the weights $w(u, v)$ and apply Dijkstra's single source shortest path algorithm. To find the weights we will use an algorithm of Knaur et al. [23]. Given a fixed source point u in a polygon, their algorithm builds a data structure to handle queries of the form: Given any point v in the polygon, find the shortest path from u to a point visible from v. They call this the "shortest inspection-path". The length of the shortest inspection path is the weight $w(u, v)$ that we want to compute. Their algorithm takes $O(n)$ time and space for preprocessing, and $O(\log n)$ time for each query.

Because we want to compute $w(u, v)$ for each pair u, v, we must build their data structure for each point u and then query for each point v. Thus the total running time is $O(n^2 \log n)$.

Once we have computed the weights $w(u, v)$, we can apply Dijkstra's shortest path algorithm to compute the shortest awakening path tree. Dijkstra's algorithm takes time $O(m + n \log n)$ with a priority queue implementation for a graph with m edges and n vertices. Since our graph is a complete digraph, thus our total time bound is dominated by the step of calculating edge weights, which takes $O(n^2 \log n)$ time. This method uses space $O(n^2)$ since we store all the edge weights.

Before we describe how to reduce the space, we will describe the algorithm of Knauer et al. [23] and mention which steps do not need to be repeated for each source vertex u. Briefly, their algorithm uses the following steps (many of which involve previously solved problems):

1. Compute a data structure that supports $O(\log n)$ time shortest path queries between any two points $a, b \in P$ with $O(n)$ preprocessing time. Note that this step only needs to be done once.

2. Compute a data structure that supports $O(\log n)$ time ray-shooting queries from any point a with any direction d. This step also requires $O(n)$ preprocessing time, and only needs to be done once.

3. Compute the shortest path tree rooted at u and preprocess it to support $O(1)$ time Least Common Ancestor queries.

Using these data structures, they show how to handle a query for point v in $O(\log n)$ time. Even more briefly, this involves finding the shortest path from u to v in the polygon, then shooting a ray from v back along the last segment of the shortest path to find the "best" line from which to see v, and then finding how to reach this line via a shortest path from u. For details, see their paper.

Note that this method uses $O(n^2)$ space because we explicitly store the weights $w(u, v)$ for all pairs u, v. Next we show how to save space by only computing the weights as needed during Dijkstra's algorithm.

Recall that Dijkstra's algorithm first pushes all the nodes into a priority queue with infinitely large weights except the root node with weight 0. In each round of Dijkstra's algorithm we dequeue a node u with the minimum weight, and update the weights of the rest of the nodes in the queue. The information required in each round is the weights of edges incident with u. Thus, for each dequeued node u, we apply Knauer et al.'s algorithm to calculate the weights on edges (u, v) for all v. Note that we do not need to store all the n^2 weights. Thus, our algorithm only requires $O(n)$ space.

3.5 $O(\Delta)$-Approximation Algorithm

Recall that from the last section, we are able to build the shortest awakening path tree T in $O(n^2 \log n)$ time. Clearly the height of T is a lower bound for solutions to the VFTP. In this section, we will give an $O(\Delta)$-approximation algorithm for the Line/Point Freeze-Tag Problem, where Δ is the maximum degree of the shortest awakening path tree T. In the worst case Δ is $(n_R - 1)$. The basic idea is that each awakened robot u will awaken all its children in T in a certain order.

More specifically, we first construct a shortest awakening path tree T using the algorithm in Section 3.4. Then for every awakened internal node u of T, order the children $c_1, \ldots, c_k$ of u in order of the edge weight $w(u, c_i)$ from smallest to largest. Our algorithm sends robot u to awaken its children in this order, returning to its original position after each child.

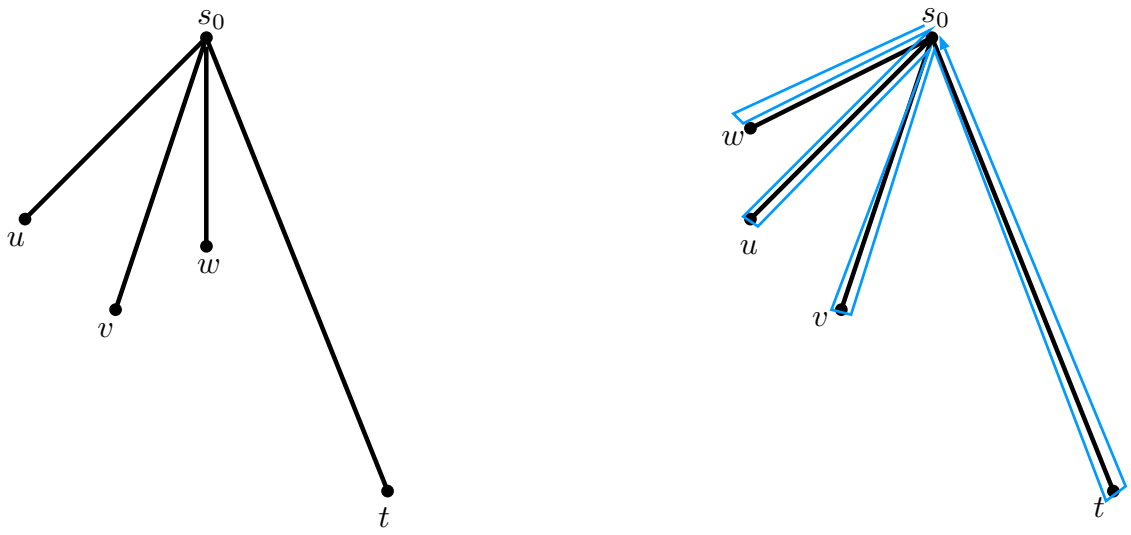

Figure 3.6: An example explaining the awakening strategy: We first sort all the children of s_0—(u, v, w, t) based on the edge weights in ascending order (w, u, v, t). Once s_0 is active, s_0 will first unfreeze w, then move back to its original position. After that, s_0 will move to unfreeze u, and then move back. The awakening procedure for v and t is similar.

To show that the above algorithm yields an approximation ratio of $O(\Delta)$, we will first prove the following lemma about an upper bound on the length of the path each internal robot travels.

Lemma 3.5.1. *Let $w(u, c_j)$ be the weight of the edge between u and its child c_j and $w_{ALG}(u, c_j)$ be the distance u actually travels when u awakens c_j by our algorithm. Then $w_{ALG}(u, c_j) \leq (2j - 1)w(u, c_j)$ for $j = 1$ to k, where k is the the number of children of u in T.*

Proof. When u awakens c_j, by our algorithm u has already awakened its children $c_1, \ldots, c_{j-1}$. In addition, since u returns to its original position after awakening each child, u has travelled through each edge $(u, c_1), \ldots, (u, c_{j-1})$ twice. Thus, the total time it takes for u to reach c_j is $w_{ALG}(u, c_j) = 2\sum_{i=1}^{j-1} w(u, c_i) + w(u, c_j)$. In addition, robot u visits its children in increasing order of the edge weighs. Thus, we have $w(u, c_i) \leq w(u, c_{i+1})$ for all $i = 1$ to $j - 1$.

Combining the above we have

$$w_{ALG}(u, c_j) = 2\sum_{i=1}^{j-1} w(u, c_i) + w(u, c_j) \leq 2\sum_{i=1}^{j-1} w(u, c_j) + w(u, c_j) = (2j - 1)w(u, c_j)$$

This completes the proof. □

Next we bound the approximation ratio.

Lemma 3.5.2. *The above algorithm is an $O(\Delta)$ approximation.*

Proof. Consider a root to leaf path $P = \{s_0, s_1, ..., s_i\}$ in T. By the definition of the shortest awakening path tree, the sum of the edge weights along P, i.e., $T(s_i) = \sum_{j<i} w(s_j, s_{j+1})$ is the minimum time required for s_0 to reach s_k with the help of intermediate robots. Let $T_{OPT}(s_i)$ be the time to awaken robot s_i in the optimum schedule. Clearly $T_{OPT}(s_i) \geq T(s_i)$. Let $T_{ALG}(s_i)$ be the time to awaken robot s_i using our algorithm. We will show that $T_{ALG}(s_i) \leq \Delta T_{OPT}(s_i)$.

By Lemma 3.5.1, our algorithm will increase the distance for each robot to reach its child c_j by a factor of at most $(2j-1)$, i.e., $w_{ALG}(s_j, s_{j+1}) \leq (2j-1)w(s_j, s_{j+1})$. Since Δ is the maximum degree the T, we will always have $j \leq \Delta$.

Combining these we have

$$\begin{aligned} T_{ALG}(s_i) &= \sum_{j<i} w_{ALG}(s_j, s_{j+1}) \\ &\leq \sum_{j<i} (2j-1) w(s_j, s_{j+1}) \\ &\leq \sum_{j<i} (2\Delta - 1) w(s_j, s_{j+1}) \\ &\leq (2\Delta - 1) \sum_{j<i} w(s_j, s_{j+1}) \\ &\leq (2\Delta - 1) T(s_i) \leq (2\Delta - 1) T_{OPT}(s_i) \end{aligned}$$

Thus our algorithm is an $O(\Delta)$ approximation. □

Note that the running time of the above algorithm is dominated by the step of constructing the shortest path awakening tree. Thus, our algorithm runs in time $O(n^2 \log n)$. In addition, in the worst case Δ could be $n_R - 1$. Thus, our algorithm is $O(n_R)$-approximation in the worst case.

Chapter 4

The Line/Point Freeze-Tag Problem

4.1 Introduction

In this chapter, we will look more closely at the Line/Point Freeze-Tag Problem. Recall from Chapter 1 that in the Line/Point Freeze-Tag Problem, each point (robot) p has an *awakening line* through it, and robot p is awakened when another awake robot touches the line. There is one initial awake robot and the goal is awaken all the robots in the minimum time. In Section 4.2, we will give a polynomial time algorithm for a special case of this problem, where all the awakening lines are parallel to each other. In Section 4.3, we will show that the Line/Point Freeze-Tag Problem is NP-hard, even for the special case of horizontal and vertical awakening lines.. In Section 4.4, we will give an $O(1)$-approximation algorithm for the Line/Point Freeze-Tag Problem.

4.2 Parallel Awakening Lines

For the special case of the Line/Point Freeze-Tag Problem where all the awakening lines are parallel, we are able to give a polynomial time algorithm. The idea is to transform the problem to a 1-Dimensional (1-D) Freeze-Tag Problem, i.e., the FTP for points on a line. Then we will give an algorithm for the 1-Dimensional Freeze-Tag Problem.

Lemma 4.2.1. *Suppose we have an instance of the Line/Point Freeze-Tag Problem with the initial active robot s, and a set of robots R associated with awakening lines $\{l_0, l_1, ..., l_{n-1}\}$, such that all the awakening lines are parallel to each other. We can construct an instance*

of the 1-D Freeze-Tag Problem, such that an optimum solution for the instance of the Line/Point Freeze-Tag Problem can be obtained by following the same awakening schedule of the optimum solution for the instance of the 1-D Freeze-Tag Problem we constructed.

Proof. We will first show the construction. Place a line l_p perpendicular to all the awakening lines. Clearly, l_p intersects all the awakening lines. Next we place n robots r_i on the intersection of l_p and l_i for $i = 0$ to $n - 1$. At last, we place the new initial active robot s' by projecting s orthogonally onto l_p. This completes our construction. See Figure 4.1 for an example.

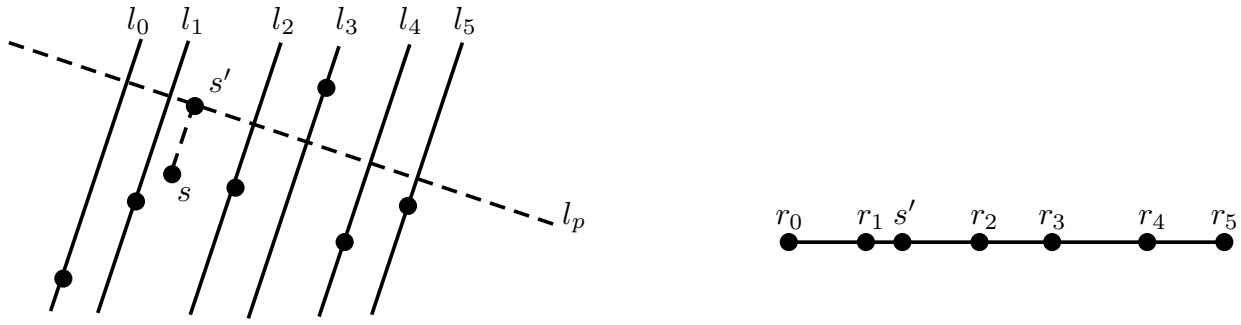

Figure 4.1: The left figure is an example of the Line/Point FTP with an active robot at points s, where l_p (shown in dashed) is the base for the corresponding 1-D FTP instance. The right figure is the corresponding 1-D FTP instance with the active robot s'.

By our choice of l_p, the distance between any two robots r_i and r_j on the line we constructed is equal to the shortest distance for the robot on l_i to reach l_j or the robot on l_j to reach l_i in the Line/Point Freeze-Tag Problem. In addition, since s' is the orthogonal projection of s on l_p, the distance from s' to r_i is equal to the distance for robot s to reach l_i for all $i = 0$ to $n - 1$.

We claim that the Line/Point FTP instance has a solution within time T if and only if the constructed 1-Dimensional FTP instance has a solution within time T.

For one direction, suppose that there exists a solution to the the 1-D FTP instance within time T. We can obtain a solution to the Line/Point FTP instance within time T by following the same awakening schedule and asking each corresponding robot in the Line/Point FTP instance to always travel by the shortest path. Since the shortest path

for any robot r_i to awaken another robot r_j equals the shortest distance for the robot on l_i to reach l_j, we have a solution to the Line/Point FTP instance within time T.

For the other direction, suppose that there exists a solution to the Line/Point FTP instance within time T. We can obtain a solution to the 1-D FTP instance within time T by following the same awakening schedule in the solution to the Line/Point FTP instance. If all the robots in the solution to the Line/Point FTP always travel by the shortest paths to wake up other robots, then our solution to the 1-D FTP will stay within time T by construction. Otherwise, if a robot doesn't following the shortest path to awaken some other robots, then we can shorten the paths, and this won't affect the later robots since all the awakening lines are parallel to each other. □

The above Lemma shows that for the special case of the Line/Point Freeze-Tag Problem where the awakening lines are parallel, we can convert it to an instance of the 1-D Freeze-Tag Problem. Next we give an algorithm for solving the 1-D Freeze-Tag Problem. Although this algorithm is very simple, we have not found it in the literature. The 1-Dimensional Freeze-Tag problem is discussed by Armon et al. [6]. They show that the greedy approach does not give optimum solution. In addition, they prove that the greedy heuristic for the 1-Dimensional Freeze-Tag Problem is a 4-approximation.

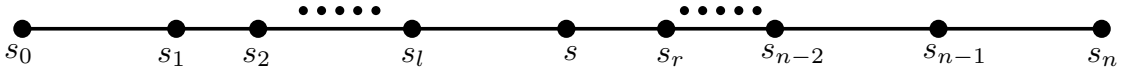

Figure 4.2: An instance of the FTP on a line

Theorem 4.2.1. *Suppose we have an instance of the FTP on a line with the initial active robot s, and frozen robots $s_0, s_1, s_2, ..., s_n$ in that order along the line. Let s_l and s_r be the robots to the left and right of s, respectively (See Figure 4.2), and let $d(s,t)$ denote the distance between two points s and t on the line. The optimum awakening schedule will take time:*

$$min \left\{ \begin{array}{c} max\{d(s,s_0), d(s,s_l) + d(s_l,s_n)\} \\ max\{d(s,s_n), d(s,s_r) + d(s_r,s_0)\} \end{array} \right\}$$

Proof. We claim that in the optimum awakening schedule, either s_l or s_r will be the first robot awakened by s, and after either s_l or s_r is awakened, it will awaken all the robots on

one side of the line, and s will awaken the robots on the other side. See Figure 4.3 for an example.

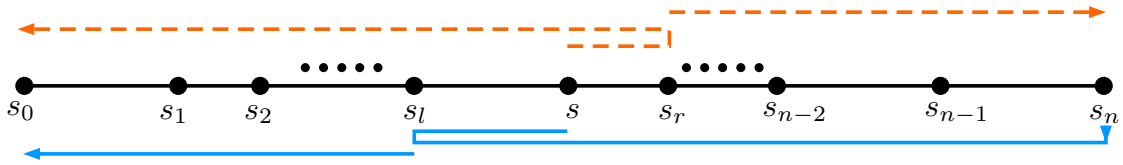

Figure 4.3: Two possible optimal awakening schedules: either s awakens s_r and then awakens every robot to the left while s_r awakens those to the right (shown in dashed red); or s awakens s_l and then awakens every robot to the right while s_l awakens those to the left (shown in solid blue).

To prove our claim, note that either s_l or s_r will be the first robot awakened by s no matter which direction s chooses to start with. Without loss of generality, assume s chooses to go left and awaken s_l first. Since both active robots are at s_l, it will take at least time $d(s_0, s_l)$ to reach s_0, and at least time $d(s_l, s_n)$ to reach s_n. Thus, the minimum time needed to wake up all the robots if s chooses to go left is $max\{d(s_0, s), d(s_l, s) + d(s_l, s_n)\}$. Furthermore, this minimum is achievable. The analysis for s to go right at first is similar. Thus, the minimum amount of time to awaken all the robots is the smaller of the two. □

4.3 NP-hardness of the Line/Point Freeze-Tag Problem[1]

In this section, we will show that the Line/Point Freeze-Tag Problem is NP-hard by reduction from a well-known NP-complete problem, called 3-Satisfiability (or 3SAT). The 3SAT problem is one of the first problems shown by Cook [11] to be NP-complete. The 3SAT problem consists a set of n variables $v_1, v_2, \ldots, v_n$ and m clauses $C_1, C_2, \ldots, C_m$, where each clause is a disjunction of three literals. Note that both v_i and $\neg v_i$ count as literals. The 3SAT problem asks whether there exists an assignment to each variable with either *true* or *false*, such that $\bigwedge_{j \leq m} C_j$ evaluates to *true*.

Briefly, our construction is as follows. For each variable v_i we make a *primary robot* (which we also call v_i) with a horizontal awakening line, and create two sets of *literal*

[1]This section represents joint work with Kathie Cameron and Juraj Stacho as well as Anna Lubiw.

robots with vertical awakening lines — the "true" lines T_i and the "false" lines F_i. By manipulating the position of the awakening lines, we force v_i to choose either to awaken the lines in T_i or the lines in F_i, but not both. For each clause C_i, we create a *clause robot* c_i with a horizontal awakening line that can only be awakened by one of three points, with those points corresponding to the literals in the clause. This is the main idea, and the one complication is that we need to add some *relay robots* to help awaken all the robots, not just the ones associated with the true literal robots.

Formally, the 3SAT problem is as follows.

Instance: A set of Boolean variables $V = \{v_1, v_2, \ldots, v_n\}$ and a set of clauses $C = \{C_1, C_2, \ldots, C_m\}$, where each clause is a disjunction of at most three variables from V.

Question: Is there a truth assignment to V such that the formula $\bigwedge_{j<n} C_j$ evaluates to *true*?

Theorem 4.3.1. *The Line/Point Freeze-Tag Problem is NP-hard, even for the case where all the awakening lines are horizontal or vertical.*

Proof. We will reduce 3SAT to the Line/Point FTP with only horizontal and vertical awakening lines. Let ϵ be a sufficiently small number (e.g., $\epsilon = 1$), and let d be sufficiently large (e.g., $d = 4 > 3\epsilon$). Consider the Euclidean plane with s_0 the initial active robot at the origin $(0, 0)$. Given an instance of the 3SAT problem, we construct an instance of the Line/Point FTP as follows. See Figure 4.4 for an example:

- **Primary Robots:** n robots $R_v = \{v_1, v_2, \ldots, v_n\}$ associated with horizontal awakening lines. Robot v_i is positioned at point $(6di, \frac{\epsilon}{n}(n - i + 1))$.
- **Clause Robots:** m robots $R_c = \{c_1, c_2, \ldots, c_m\}$ associated with horizontal awakening lines. Robot c_i is positioned at point $(0, -6di)$.
- **Literal Robots:** Let v_l be a variable in the 3SAT instance, and let $\{C_{t_1}, C_{t_2}, \ldots, C_{t_j}\}$ and $\{C_{f_1}, C_{f_2}, \ldots, C_{f_k}\}$ be the set of clauses which contain v_l and $\neg v_l$, respectively. For each v_l, we place j robots $T_l = \{t_{l,1}, \ldots, t_{l,j}\}$ such that $t_{l,i}$ is at point $(6dl - d - \frac{\epsilon}{l}i, -6dt_i + d)$, and place k robots $F_l = \{f_{l,1}, \ldots, f_{l,k}\}$ such that $f_{l,i}$ is at point $(6dl + d - \frac{\epsilon}{l}i, -6df_i + d)$. Define $T' = \bigcup_{i \le n} T_i$ and $F' = \bigcup_{i \le n} F_i$. At last, we associate all the robots in T' and F' with vertical awakening lines.
- **Relay Robots:**
  - n robots $R_w = \{v'_1, v'_2, \ldots, v'_n\}$ associated with horizontal awakening lines. Robot v'_i is positioned at point $(6di, -(6md + 6di))$.

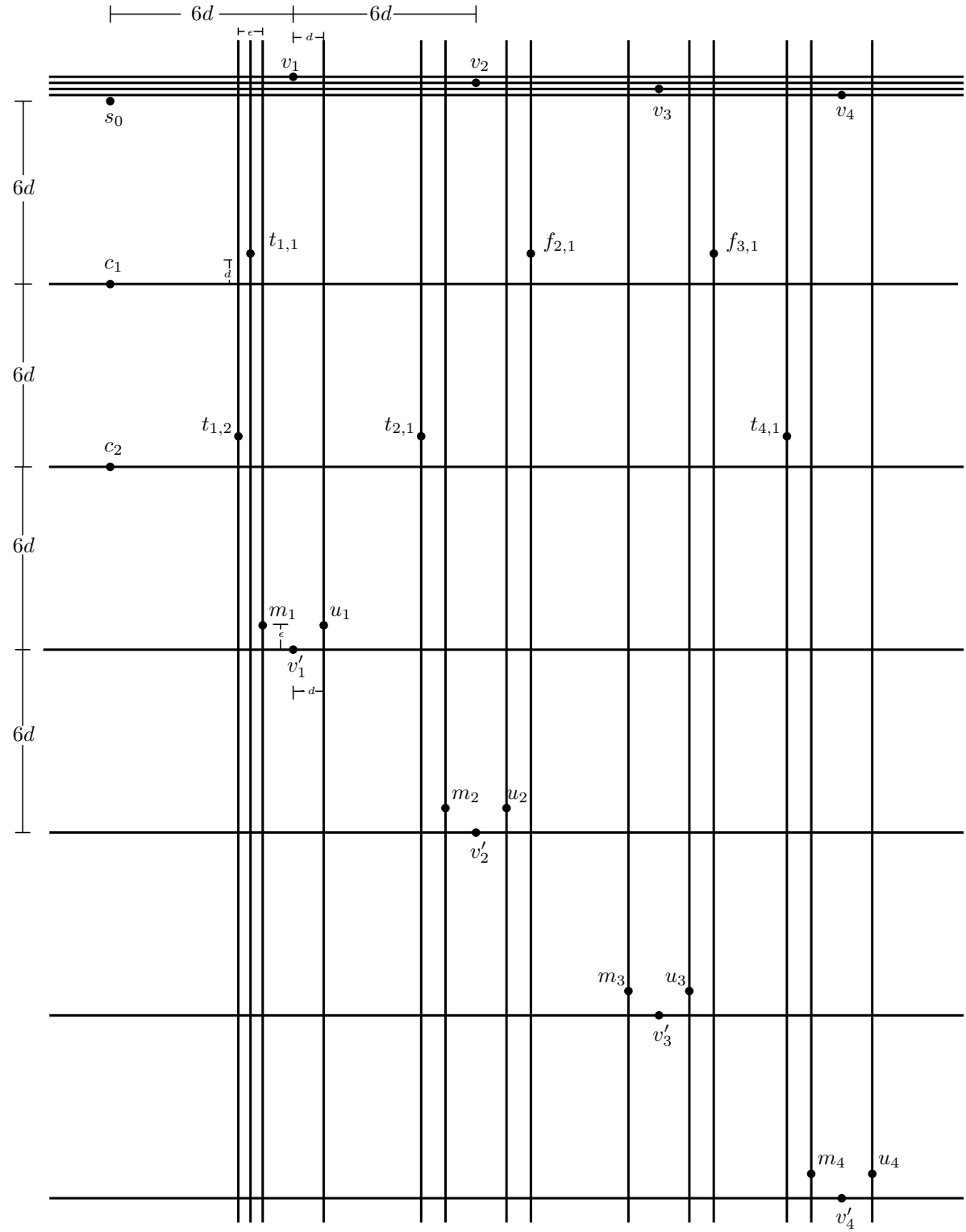

Figure 4.4: This is an example of a Line/Point FTP instance constructed from a 3SAT instance with variables $V = \{v_1, v_2, v_3, v_4\}$ and clauses $C = \{C_1 = \{v_1, \neg v_2, \neg v_3\}, C_2 = \{v_1, v_2, v_4\}\}$.

– $2n$ robots $R_m = \{m_1, m_2, \ldots, m_n\}$ and $R_u = \{u_1, u_2, \ldots, u_n\}$. Each robot is associated with a vertical awakening line: m_i is positioned at point $(6di - d, -(6md + 6di) + \epsilon)$ and u_i is positioned at point $(6di + d, -(6md + 6di) + \epsilon)$.

Clearly the construction can be done in polynomial time. We claim that there is a schedule to awaken all the robots within time $2d+3\epsilon$ in the Line/Point Freeze-Tag Problem instance we constructed, if and only if there is an assignment to satisfy all the clauses in the original 3SAT instance.

To see the "if" part holds, suppose that there is an assignment ϕ to satisfy all the clauses in C in the 3SAT instance. We want to show that we will have a schedule to awaken all the robots within time $2d + 3\epsilon$ in the Line/Point FTP instance we constructed. We start with moving s_0 straight up until it awakens all the primary robots in R_v. This can be done within time ϵ. For each active robot v_i, we move v_i leftward to awaken all the literal robots in T_i if $\phi(v_i) = true$. Otherwise we move v_i rightward to awaken the literal robots in F_i. Since the furthest literal robot in T_i and F_i is distance $(d + \epsilon)$ away from v_i, the robots in T' and F' will be awakened within time $2\epsilon + d$. Next, we move each awakened literal robot v in T' and F' straight down until it reaches the awakening line of a clause robot $c_k \in R_c$. Since ϕ satisfies all the clauses in C, for each c_i there must be at least one literal robot at distance d above the awakening line of c_i. Thus, all the robots in R_c can be awakened by time $2\epsilon + d + d = 2\epsilon + 2d$. Note that for each primary robot v_i, it only awakens either T_i or F_i. We still need to wake up all the robots in the other set (F_i or T_i) that was not awakened by v_i for all the primary robots. First note that m_i or u_i will be awakened by v_i within time $\epsilon + d$ when r_i is on its way to awaken T_i or F_i. Without loss of generality, suppose that v_i moves to awaken T_i. Since m_i is exactly distance d away from v_i, then m_i will be awakened within time $\epsilon + d$. We move m_i straight down to awaken w_i (which takes time ϵ), and move w_i rightward to awaken all the robots in $F_i \cup u_i$ (which takes at most time $d + \epsilon$). The analysis for the case where v_i awakens F_i is symmetric. Thus, the total time required to awaken all the robots in the Line/Point FTP instance we constructed is $2d + 3\epsilon$.

Conversely, suppose that there is a schedule to awaken all the robots within time $2d+3\epsilon$ in the Line/Point FTP instance we constructed. We want to show that we will have an assignment ϕ to satisfy all the clauses in the 3SAT instance. We will first prove the following lemmas.

Lemma 4.3.1. *For every primary robot v_i, the first active robot in $T_i \cup \{m_i\} \cup F_i \cup \{n_i\}$ must be awakened by v_i. In addition, robot v_i can only awaken the robots in either $T_i \cup \{m_i\}$ or $F_i \cup \{n_i\}$, but not in both sets.*

Proof. Observe that only robots v_i and v_i' are within distance $2d + 3\epsilon$ to any robot in $T_i \cup \{m_i\} \cup F_i \cup \{n_i\}$. However, v_i' must be awakened by m_i or n_i, since the nearest robot to v_i' besides m_i and n_i is at distance $6d - \epsilon$ away. Thus, the first active robot in $T_i \cup \{m_i\} \cup F_i \cup \{n_i\}$ must be awakened by v_i. In addition, the minimum time required for v_i to awaken at least one robot from both $T_l \cup \{m_i\}$ and $F_l \cup \{n_i\}$ is at least $3d > 2d + 3\epsilon$. The minimum occurs when v_i only awakens m_i and n_i. Thus, v_i can only awaken the robots in either $T_i \cup \{m_i\}$ or $F_i \cup \{n_i\}$. □

Lemma 4.3.2. *For every robot $c_i \in R_c$, c_i is awakened by a robot u either in T' or in F'. In addition, robot u must be constructed from one of the literals in clause C_i in the original 3SAT problem.*

Proof. Observe that the nearest robot to c_i not in T' and F' is at least distance $6d$ away. Thus, c_i can only be awakened by a robot in T' or F'. By our construction, only the robots which are create from the literals in C_i are within distance $2d + 3\epsilon$ of the awakening line of c_i. Thus, u must be constructed from one of the literals in C_i. □

Lemma 4.3.3. *If robot v_i chooses to awaken the robots in $F_i \cup \{n_i\}$, then the first active robot in $T_i \cup \{m_i\}$ can only be awakened by v_i'. Similarly, if robot v_i chooses to awaken the robots in $T_i \cup \{m_i\}$, then the first active robot in $F_i \cup \{n_i\}$ can only be awakened by v_i'.*

Proof. We prove the first statement (the other follows by symmetry). First note that the robots within distance $2d+3\epsilon$ to any robot in $F_i \cup \{n_i\}$ are the robots in $T_l \cup \{m_i\} \cup \{v_i\} \cup \{v_i'\}$. However, by Lemma 4.3.1, v_i can only awaken the robots in either $T_i \cup \{m_i\}$ or $F_i \cup \{n_i\}$. By a similar argument, robots in $T_i \cup \{m_i\}$ also can not awaken the robots in $F_i \cup \{n_i\}$ without exceeding the time constraint. This leave us with only v_i'. □

Lemma 4.3.4. *For any robot $c_k \in R_c$, if c_k is awakened by a robot in T_i then v_i must awaken some robots in $T_i \cup \{m_i\}$. Otherwise, if c_k is awakened by a robot in F_i then v_i must awaken some robots in $F_i \cup \{n_i\}$.*

Proof. Without loss of generality, suppose that c_k is awakened by a robot in T_i. Assume to the contrary that v_i awakens some robots in $F_i \cup \{n_i\}$. By the previous lemma, the first active robot in $T_i \cup \{m_i\}$ can only be awakened by v_i'. Since it takes at least time d for v_i to awaken any robot in $F_i \cup \{n_i\}$ and time d for v_i' to awaken any robot in $T_i \cup \{m_i\}$, the lower bound for awakening any robot in $T_i \cup \{m_i\}$ is at least $d + d = 2d$ (if we ignore all the ϵ). In order to awaken any $l_k \in R_c$, a robot in $F_i \cup \{n_i\}$ must travel at least extra time d. Then the lower bound for awakening l_k is $3d$, which exceeds the time constraint. (The proof for the other direction is omitted by symmetry). This completes the proof. □

Having the lemmas above, next we will prove that we can obtain a solution to the original 3SAT problem.

By Lemma 4.3.2, every clause robot c_k is awakened by a robot u which is constructed from one of the literals of C_k. If $u \in T'$, then by our construction u is associated with a positive literal v_i. By assigning the Boolean value *true* to v_i, C_k can be satisfied. Otherwise, if $u \in F'$, then by our construction u is associated with negative literal $\neg v_i$. By assigning the Boolean value *false* to v_i, C_k can be satisfied. For those variables without assignment, we assign *true* or *false* arbitrarily. In addition, we claim that a robot will never be assigned both *true* and *false*. Suppose the contrary. This implies that two clause robots c_k and c_j are awakened by some T_i and F_i, respectively. By Lemma 4.3.4, this implies that v_i visits robots from both $T_i \cup \{m_i\}$ and $F_i \cup \{n_i\}$, which is impossible. Thus, we obtain an assignment which is valid and satisfies all the clauses in C_k. □

4.4 O(1)-Approximation Algorithm

The Line/Point Freeze-Tag Problem is very similar to the Visibility Freeze-Tag Problem. In both problems, an active robot may take advantage of intermediate robots to awaken other robots. See Figure 4.5 for an example.

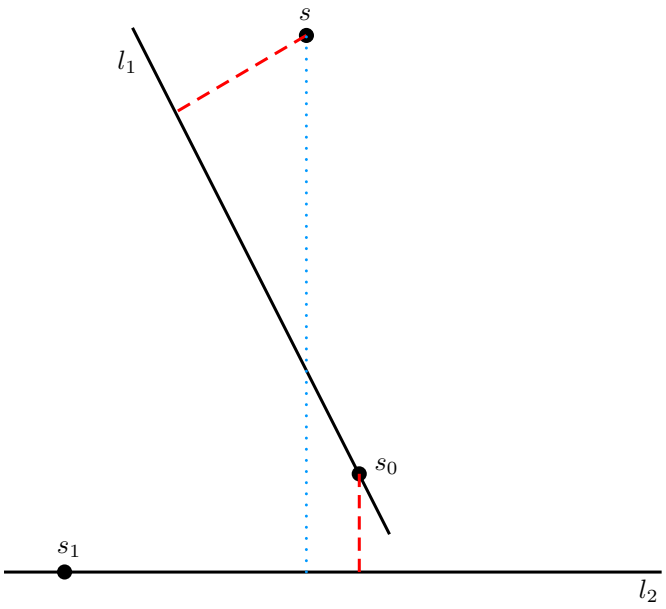

Figure 4.5: The path for s to wake up s_1 directly (shown in dotted blue) is longer than the path for s to awaken s_0 first, and let s_0 awaken s_1 (shown in dashed red).

As for the Visibility Freeze-Tag Problem, we can construct a shortest awakening path tree with Dijkstra's algorithm on the points R, where, for edge (u, v) we define its weight $w(u, v)$ to be the distance from point u to the awakening line associated with v. After constructing the shortest awakening path tree, we can obtain an $O(\Delta)$-approximation using the same method as in Section 3.5.

In this section, we will give an improved $O(1)$-approximation algorithm for the Line/Point Freeze-Tag Problem. Our algorithm asks every robot to follow a logarithmic spiral when it is awakened. This is a somewhat "oblivious" strategy since the robots do not even make use of the information about the locations of other points and lines. Nevertheless, we will show that this algorithm achieves a constant approximation factor.

The idea of following a spiral comes from solutions to the problem of searching for a line in the plane. We will begin by discussing the background for the searching problem, since we will use some of the results. The general form of the problem is that we must design a trajectory starting from the origin that will eventually reach an object at unknown position/direction in a minimum amount of time. The problem was first introduced by Bellman [8] in the 1950's. For the case of searching for a line that lies at a known distance from the origin, a solution was given by Isbell [22]. The case where the distance to the line

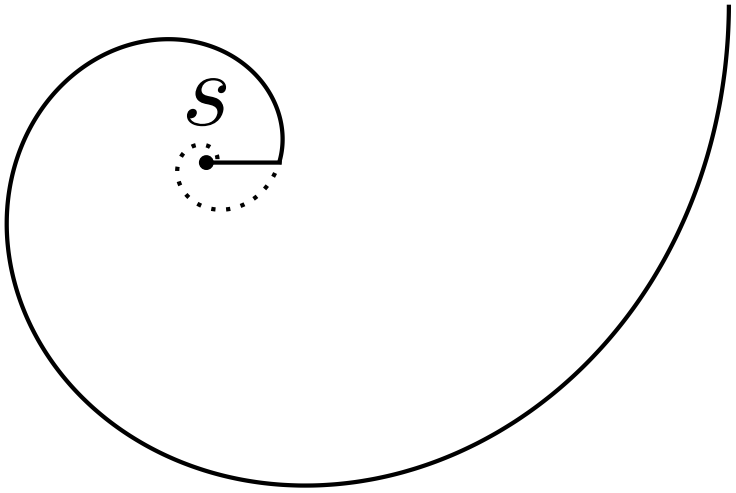

Figure 4.6: To solve the issue with starting from the origin, we ask s to travel to a straight distance first and then follow the spiral.

is not known was considered by Baeza-Yates, Culberson, and Rawlins [7]. They suggest following a logarithmic spiral and claim that a line at distance d from the origin will be encountered within distance $13.81d + O(\log d)$ along the spiral trajectory. They leave out some details. One issue is how to connect the origin to the spiral (the spiral never actually reaches the origin). The solution from Gal [1, p. 158, Figure 9.5] is to assume that d is at least ϵ and to begin the trajectory with a straight line segment of length ϵ from the origin, and then join the logarithmic spiral. See Figure 4.6 for an example of this. Clearly this is correct, and the length is reduced. The other issue is the details of the analysis of the bound $13.81d + O(\log d)$. Further details of the analysis for the case $d = 1$ were given in an unpublished paper by Finch [18]. However, details of the constant in front of the $\log n$ term do not seem to be available anywhere. As we shall see below, we need this constant in order to make our analysis precise.

To complete our background discussion we mention that it is an open question whether the logarithmic spiral is the best solution for the problem of searching for a line in the plane. Langetepe proved a similar result for the case of searching for a point in the plane [25] and also for the case of searching for an orthogonal line in the plane [26].

We now turn to our $O(1)$ approximation for the Line/Point Freeze-Tag Problem. Our algorithm is as follows. Compute the minimum distance, $d_{\min}$, from any initial robot point

to a non-incident line. When a robot is awakened, it moves distance $d_{\min}$ horizontally to the right and then joins the logarithmic spiral centered at its original location.

In order to prove that this spiraling algorithm is an $O(1)$ approximation, we need an upper bound on the absolute competitive ratio of the spiral trajectory. Let L be a line in the plane at distance d from the origin, and let t be the length of the logarithmic spiral from the origin to the first intersection with line L. We want a constant K such that $t \leq Kd$. In theory, such a bound is available from the result of Baeza-Yates et al. [7], that $t \leq 13.81d + O(\log d)$. However, the details of this asymptotic bound are not available, and in order to turn it into an absolute bound, we need the constant in front of the $\log d$ which is not given explicitly. We therefore prove a rougher absolute bound.

Claim 4.4.1. *Let L be a line in the plane at distance d from the origin, and let t be the length of the logarithmic spiral from the origin to the first intersection with line L. Then $t \leq 17.289d$.*

Proof. We need two basic facts about the logarithmic spiral $r = e^{\alpha\theta}$ [31]:

(1) if z is a point on the spiral at distance $d(z)$ from the origin then the distance along the spiral from the origin to z converges to $\frac{\sqrt{1+\alpha^2}}{\alpha} d(z)$.

(2) if the spiral crosses the x-axis at coordinate b, then the next crossing is at x-coordinate $e^{\alpha 2\pi} b$.

Consider a line L in the plane and the logarithmic spiral from the origin. Suppose without loss of generality that L is a vertical line. Then L intersects the x-axis at the point $(d, 0)$. The spiral crosses the x-axis infinitely often. Let $(f, 0)$ and $(f', 0)$ be the two points where the spiral crosses the x-axis before and after $(d, 0)$. By the time the spiral reaches $(f', 0)$, it has crossed line L. We have $f \leq d < f'$. By (2), $f' = e^{\alpha 2\pi} f \leq e^{\alpha 2\pi} d$. Thus, the distance along the spiral to $(f', 0)$ is at most $\frac{\sqrt{1+\alpha^2}}{\alpha} f' \leq \frac{\sqrt{1+\alpha^2}}{\alpha} e^{2\alpha\pi} d$. Solving this (using Wolfram Alpha) as a function of α, we have $\frac{\sqrt{1+\alpha^2}}{\alpha} e^{2\alpha\pi}$ achieves a local minimum of $17.289\ldots$ at $\alpha = 0.155402\ldots$. Thus the distance along the spiral until we cross L is at most $17.289d$. □

Lemma 4.4.1. *The spiraling algorithm described above is an $O(1)$ approximation for the Line/Point Freeze-Tag Problem.*

Proof. Consider an optimum awakening schedule for an instance of the Line/Point FTP. Let $T_{\mathrm{OPT}}(s_i)$ be the time to awaken robot s_i in this optimum schedule. Let $T_S(s_i)$ be the time to awaken robot s_i using the spiraling strategy. We will show that $T_S(s_i) \leq kT_{\mathrm{OPT}}(s_i)$.

Suppose that in the optimum awakening schedule, s_i is awakened via the sequence $s = s_0, s_1, \ldots, s_i$. In other words, for $j = 0, \ldots, i-1$, robot s_j, once it is awakened, follows some path in the plane, eventually touching the awakening line of s_{j+1}. The time for s_j to awaken s_{j+1} is at least $w(s_j s_{j+1})$, the distance from the original position of s_j to the awakening line of s_{j+1}. Thus $T_{\mathrm{OPT}}(s_i) \geq \sum_{j=0}^{i-1} w(s_j, s_{j+1})$.

Let $S(s_j, s_{j+1})$ be the time for the spiral path from s_j to reach the awakening line of s_{j+1}. By the Claim above $S(s_j, s_{j+1}) \leq 17.289 w(s_j, s_{j+1})$.

Combining these we have

$$T_S(s_i) \leq \sum_{0 \leq j < i} S(s_j, s_{j+1}) \leq \sum_{0 \leq j < i} 17.289 w(s_j, s_{j+1}) \leq 17.289 T_{\mathrm{OPT}}(s_i)$$

.

□

Chapter 5

Conclusion and Open Problems

For the Freeze-Tag Problem in polygons and the Visibility Freeze-Tag Problem, no previous work has been done to show the complexity of either problem. We prove both problems are NP-hard, and we give an algorithm to approximate the Visibility Freeze-Tag Problem within a factor of $O(\Delta)$, where Δ is the maximum degree of the shortest awakening path tree. Our algorithm takes time $O(n^2 \log n)$ and space $O(n)$. For the Line/Point Freeze-Tag problem, we show this problem is NP-hard, and provide a polynomial-time $O(1)$-approximation algorithm. We also give an algorithm to find the optimum awakening schedule for a special case of the Line/Point Freeze-Tag Problem where all the awakening lines are parallel, and this algorithm also works for the 1-D Freeze-Tag Problem.

There are many open problems left for the FTP and its variants.

For the Freeze-Tag Problem in polygons, the current best algorithm (which is the greedy algorithm) gives an approximation ratio of $O(\log n)$. However, the Freeze-Tag Problem for points in the plane has a PTAS. One interesting open problem is whether the Freeze-Tag Problem in polygons has a PTAS.

For the Visibility Freeze-Tag Problem, our algorithm could still give an approximation factor as bad as $O(n)$ in the worst case. Finding an algorithm that achieves an $O(\log n)$ or ideally an $O(1)$ approximation ratio is worth pursing. Our algorithm made use of the shortest awakening path tree. Each node s awakened all its children, in order of edge weights, returning to its original position after each one. This is not the best path for the node to awaken its children. For a given node s and its children, we know how to calculate the shortest path for s to awaken all its children using the algorithm for the Watchman Route Problem. However, this makes the analysis much more difficult. We believe that a careful analysis may give a better approximation ratio than $O(\Delta)$.

For the Line/Point Freeze-Tag problem, we gave a polynomial time O(1)-approximation algorithm. The movement of the robots in our algorithm was independent of the input lines, which gives hope for improvement. We conjecture that there is a PTAS for this problem.

References

[1] S. Alpern and S. Gal. *The Theory of Search Games and Rendezvous*. International Series in Operations Research & Management Science. Springer, 2003.

[2] Muhammed H Alsuwaiyel and D.T. Lee. Minimal link visibility paths inside a simple polygon. *Computational Geometry*, 3(1):1–25, 1993.

[3] Esther M Arkin, Michael A Bender, Sándor P Fekete, Joseph SB Mitchell, and Martin Skutella. The freeze-tag problem: how to wake up a swarm of robots. *Algorithmica*, 46(2):193–221, 2006.

[4] Esther M Arkin, Michael A Bender, and Dongdong Ge. Improved approximation algorithms for the freeze-tag problem. In *Proceedings of the Fifteenth Annual Symposium on Parallelism in Algorithms and Architectures*, pages 295–303. ACM, 2003.

[5] Esther M Arkin, Joseph SB Mitchell, and Christine D Piatko. Minimum-link watchman tours. *Information Processing Letters*, 86(4):203–207, 2003.

[6] Amitai Armon, Adi Avidor, and Oded Schwartz. Cooperative TSP. *Theoretical Computer Science*, 411(3133):2847 – 2863, 2010.

[7] Ricardo A Baeza-Yates, Joseph C Culberson, and Gregory JE Rawlins. Searching in the plane. *Information and Computation*, 106(2):234–252, 1993.

[8] R. Bellman. A minimization problem. *Bull. Amer. Math. Soc. 62 (1956) 270.*

[9] Svante Carlsson, Håkan Jonsson, and Bengt J. Nilsson. Approximating the shortest watchman route in a simple polygon. In *In Proc. 4th Annual International Symposium on Algorithm and Computation, volume 762 of Lecture Notes Computer Science*, pages 58–67. Springer-Verlag, 1993.

[10] Wei-pang Chin and Simeon Ntafos. Optimum watchman routes. *Information Processing Letters*, 28(1):39–44, 1988.

[11] Stephen A. Cook. The complexity of theorem-proving procedures, 1971.

[12] Bastian Degener, Sándor P. Fekete, Barbara Kempkes, and Friedhelm Meyer auf der Heide. A survey on relay placement with runtime and approximation guarantees. *Computer Science Review*, 5(1):57 – 68, 2011.

[13] Erik D. Demaine, Mohammadtaghi Hajiaghayi, Hamid Mahini, Amin S. Sayedi-Roshkhar, Shayan Oveisgharan, and Morteza Zadimoghaddam. Minimizing movement. *ACM Trans. Algorithms*, 5(3):30:1–30:30, July 2009.

[14] Xiaotie Deng, Tiko Kameda, and Christos Papadimitriou. How to learn an unknown environment. I: The rectilinear case. *J. ACM*, 45(2):215–245, March 1998.

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

[16] Adrian Dumitrescu and Csaba D Tóth. Watchman tours for polygons with holes. *Computational Geometry*, 45(7):326–333, 2012.

[17] Adrian Dumitrescu and Csaba D. Tóth. The traveling salesman problem for lines, balls and planes. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '13, pages 828–843. SIAM, 2013.

[18] Steven R. Finch and Li-Yan Zhu. Searching for a shoreline. *Available online at arXiv:math/0501123*, 2005.

[19] Roland Hagius, Christian Icking, and Elmar Langetepe. Lower bounds for the polygon exploration problem. In *Abstracts 20th European Workshop Comput. Geom*, pages 135–138, 2003.

[20] Frank Hoffmann, Christian Icking, Rolf Klein, and Klaus Kriegel. A competitive strategy for learning a polygon. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '97, pages 166–174, Philadelphia, PA, USA, 1997. SIAM.

[21] Frank Hoffmann, Christian Icking, Rolf Klein, and Klaus Kriegel. The polygon exploration problem. *SIAM Journal on Computing*, 31(2):577–600, 2001.

[22] J. R. Isbell. An optimal search pattern. *Naval Res. Logist. Quart. 4 (1957)*, pages 357–359.

[23] Christian Knauer, Günter Rote, and Lena Schlipf. Shortest inspection-path queries in simple polygons. In *Proceedings of the 24th European Workshop on Computational Geometry (EuroCG)*, pages 153–156, March 2008.

[24] Jochen Könemann, Asaf Levin, and Amitabh Sinha. Approximating the degree-bounded minimum diameter spanning tree problem. *Algorithmica*, 41(2):117–129, 2005.

[25] Elmar Langetepe. On the optimality of spiral search. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1–12. SIAM, 2010.

[26] Elmar Langetepe. Searching for an axis-parallel shoreline. *Theoretical Computer Science*, 447:85–99, 2012.

[27] Joseph S. B. Mitchell. Approximating watchman routes. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, chapter 60, pages 844–855. 2013.

[28] Eli Packer. Computing multiple watchman routes. In CatherineC. McGeoch, editor, *Experimental Algorithms*, volume 5038 of *Lecture Notes in Computer Science*, pages 114–128. Springer Berlin Heidelberg, 2008.

[29] Marcelo O Sztainberg, Esther M Arkin, Michael A Bender, and Joseph SB Mitchell. Theoretical and experimental analysis of heuristics for the "freeze-tag" robot awakening problem. *Robotics, IEEE Transactions on*, 20(4):691–701, 2004.

[30] Xuehou Tan. Approximation algorithms for the watchman route and zookeeper's problems. *Discrete Applied Mathematics*, 136(2):363–376, 2004.

[31] Eric W. Weisstein. "Logarithmic Spiral." From MathWorld—A Wolfram Web Resource. http://mathworld.wolfram.com/LogarithmicSpiral.html. Last visited on 30/7/2014.