Searching for a Shoreline

by

Sumi Acharjee

Master of Science, University of Dhaka, 2009

Bachelor of Science, University of Dhaka, 2008

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Science

in the program of

Applied Mathematics

Toronto, Ontario, Canada, 2020

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A THESIS

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 authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

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

Abstract

Searching for a Shoreline

Master of Science 2020

Sumi Acharjee

Applied Mathematics

Ryerson University

Search theory has a long history that dates back to the 50's. In this work, we focus on the two-dimensional search problem where n unit speed robots starting from the origin move along their own trajectories to find a line. The search algorithm terminates when any of the robots discovers the line for the first time. Our main objective is to minimize the worst case relative time until the first searcher hits the line. In this thesis, we do the competitive analysis of the two-dimensional search problem for $n \geq 2$ and restudy the existing upper bounds for $n \geq 2$. We improve the best lower bound known [8] for $n = 2$ robots from 1.5993 to 3. Also, we prove the first lower bound for $n = 3$ which is $\sqrt{3}$. For $n \geq 4$ we prove the lower bound of $\frac{1}{\cos(\pi/n)}$ which matches the best upper bound known.

Acknowledgements

I would like to express the deepest appreciation to my supervisor Dr. Konstantinos Georgiou for his valuable guidance, understanding, and support throughout my master's study and research. I have been extremely lucky to be his student. His incredible mentorship and teaching has inspired me to be a better researcher every single day. Finally, I would like to thank my family for their consistent encouragement and providing unconditional support to me throughout this journey.

Table of Contents

List of Figures

List of Appendices

Chapter 1

Introduction

Search Theory has a long history. Since the first time it was introduced 50 years ago, the field has expanded and developed in versatile ways. In this section, readers will be familiarized with the search problem we study in this work.

Consider the simple *Hide and Seek Game*, which we used to play in our childhood. In this game the hider chooses his own strategy to hide so that the searcher does not find the hider very soon. At the same time, the searcher chooses his strategy so that he can find the hider as soon as possible. This game can be related to a simple search optimization problem, where the searcher minimizes the total search time without having any knowledge of the location of the hider. The entire hide and seek space can be referred as the search space. We can expand the idea of the problem in many directions. In terms of the search space, it can be bounded or unbounded. In the perspective of analytical geometry, the search space can be one, two or three

dimensional. Even the nature of hider can be mobile or immobile. The number of searchers can be one or more. The searcher might be totally unaware of the location of the hider or it might have some prior or partial information of the hiding location. Similarly at hider's end, the hider may or may not track the location of the searcher. If the hider can track the location or strategy of the searcher, then it can choose its strategy based on that to maximize the search time, if it is mobile in nature. Besides these, there are many other extensions of the problem in different contexts.

The underlying mathematical concepts of the similar idea we just stated above give rise to a group of problems known as *Search Problems* which are popular topics in Theoretical Computer Science. Due to its numerous and versatile nature, it has been an interesting topic to researchers for decades. Over the years, Search Theory has been applied to Computer Science, Economics and Biology, and many other sectors to solve real life problems. Some well known applications are in the military, search and rescue, scheduling, evacuation, planning and many more. For example, in military and anti-terror activity, where the target is a hider, who wants to evade the searcher as long as possible, in such situations, the concept of search theory can be applied to minimize the total search time. Another simple example which is closely related to our problem is searching for a shoreline in a sea. If a ship is lost in a sea in the dense fog and wants to reach the shore at minimum time, then the concept can be applied to find an optimal measure or an algorithm to minimize the search time.

In this work, we focus on the two dimensional search problem where a number of

searchers starting at the same point search for a line in the two dimensional plane. We consider the searchers as robots and their number is denoted by n. The goal is to minimize the worst case relative time until the first searcher hits the line, i.e. the time until the line is found divided by the distance of the line to the origin, which is known as competitive ratio. A detailed explanation on this is discussed in the next chapter. Before that, below we discuss previous research works on different search problems.

1.1 Related Work

The concept of Search Theory was initiated 50 years ago. In [11, 12] almost similar Search Problems have been studied. In [11] a man searches for another man who is located at some point of a certain road. He starts at a given point considering the probability that the man might be sought in either direction from that point. How does he search to minimize the expected distance travelled and when can this minimum expectation actually be achieved was studied in [11]. With the similar idea consider the simple problem where a unit speed robot is moving on an infinite line and looking for an object which is somewhere on the line (bounded away from the origin). The goal is to minimize the bounded relative time to find the object. The problem we just described is known as Linear-Search or Cow-Path Problem, which first appeared in [12] and later restudied by the computer science community in the late 80's in [9]. The Linear Search Problem became interesting over time and drew attention of the researchers as a challenging algorithmic problem in different contexts.

As a result, over the decades, the results of numerous variations of the problem were summarized in many surveys [13, 20, 24]. Also, the underlying mathematical theory gave rise to a number of books such as [1, 3, 4].

The *Linear Search Problem* [12] can be referred as one-dimensional problem. Among different variations of the problem, searching for a line in a plane can be referred as two-dimensional problem. Our current work focuses on the two-dimensional problem where n unit speed searchers referred as *robots*, search in parallel for a line referred as shoreline on the plane. We call our problem $Shoreline_n$. Problem $Shoreline_n$, and its variations have been studied as early as in the late 50's. Our main concern is to do the *competitive analysis* [15] of the problem. We study our problem in perspective of analytical geometry and consider the plane as a Cartesian plane. Our problem can be referred as an *online problem*, where n unit speed robots starting from the origin search for a line that is placed in an unknown location. Without loss of generality we consider the distance of the line is $d \geq 1$ bounded away from the origin. To do the competitive analysis our goal is to minimize the relative worst case search time, i.e. the time until the line is found by any of the robots divided by the distance d from the origin with respect to all possible placements of the line and all possible values of d.

The two dimensional problem was studied in different contexts based on the number of searchers, placement of the line with and without prior information. Searching for a line with one robot was first proposed in [9] with no prior information of the

line. A similar problem was first proposed by Bellman(1956) named as *Swimming in a fog* where a person lost at a point in a fog and wishes to minimize the maximum time required to reach the shore, given its shape and some information about its location. The best algorithm known for $Shoreline_1$ is a logarithmic spiral search that has competitive ratio approximately 13.81 [9] when the shoreline is placed at an arbitrary distance. Technical report [23] included the mathematical derivation for this upper bound. There are many variations of the problem with some prior information. Searching for a line of arbitrary slope and a known, say unit, distance from the origin in the plane was solved by Isbell [28] and achieved the competitive ratio 6.39. When both distance and slope are known, the best possible competitive ratio is 3. When the line is axis parallel and the distance is known then the best competitive ratio is $3\sqrt{2}$ [9].When the line is known to be axis parallel, then [9] gives an upper bound of 13.02, which was improved to 12.5406 [29] and then to 12.5385 [31]. By assuming a cyclic-type trajectory, the competitive ratio is at least 12.5385 [31]. A randomized online algorithm for $Shoreline_1$ problem was discussed in [25, 26]. The only lower bound for $Shoreline_1$ problem with no prior information is reported in [8] as 6.3972.

For $n \geq 2$ robots numerous variations as well as improvements were proposed in [7, 8, 10, 29]. For $n = 2$ the best algorithm known is a double logarithmic spiral search [8] which induces an upper bound 5.2644. In this paper, we present a detailed calculation of the bound. The only lower bound for $Shoreline_2$ was reported in [8] is 1.5993 with the prior information that the distance is known. For $n \geq 3$ the ray algorithm of [8] where robots split the plane evenly and move along rays induces

competitive ratio at most $1/\cos(\frac{\pi}{n})$. To the best of our knowledge, no lower bound has been reported for $n \geq 3$.

For $n \geq 2$ robots, there are two other types of search problems, widely known as *Evacuation Problems* and *Rendezvous Problems*. In an Evacuation Problem, the algorithm terminates when the last searcher finds the target. In other words, it is similar to say that the searchers of the evacuation problem are looking for an exit to evacuate. Here, the goal is to minimize the time to reach the exit for all the robots in a worst-case scenario. Initially, evacuation problems were considered as planning of flow-problems [27] on dynamic networks with a starting point (source) and an end point (sink). In the last decade, evacuation problems have been studied with other variations such as multiple exit points [17], exit in a known or unknown domain [5]. Also the communication method has a great impact on evacuation problems. There are two basic communication methods; one is *face to face* (in which robots exchange information only when simultaneously located at the same point) and another one is *wireless communication* (in which robots can communicate with one another anywhere at any time). Based on communication methods, different problems were discussed in [18, 21] and surveys on different algorithms were discussed in [19].

The rendezvous problem refers to a problem in which robots are expected to meet at the same time and point in space. The meeting can be a task on its own or it may form a part of a more complex communication or coordination process in which the robots are involved. The rendezvous problem as we now know it was first informally

introduced by Steve Alpern in mid 1970s, known as Astronaut Problem and Telephone Problem [1], which initiated the discussion of coordination problems. Later, different versions of rendezvous problems were studied in [2, 6].

Two-dimensional search problems have been considered other than line searching. Searching for a circle was considered in [26]. Searching for a point on the plane was considered in [16, 35]. Later in 2010, Langetepe [30] proved that spiral search is optimal for two dimensional search by one robot where a searcher searches for a point in the plane. Parallel Search with bounded memory robots on the grid was considered in [22, 32, 33, 34]. Recently, in probabilistic aspect [14] studied the problem where a unit speed robot searches the half-line (or 1-ray) for a hidden item.

1.2 Thesis Organization

In Chapter 1 we discussed the motivation behind our problem with some real life examples. We also discussed the related work of Search Theory in different contexts. We briefly discussed the history of Search Theory. In Chapter 2 we focus on few basic terminologies along with some basic definitions. In this chapter we discuss the classic *Cowpath Problem* as a motivating example. Later on, we present our problem definition and specifically mention our contributions. At the end of Chapter 2, we also make a list of frequently used notations. In Chapter 3 we restudy the two dimensional cases for two or more robots. We present a detailed calculation of searching for a shoreline with two robots by the double logarithmic spiral algorithm and with three

and more robots, using the ray search algorithm. In Chapter 4 we focus on lower bounds for two and more robots which is our main contribution. Finally, in Chapter 5 we give a brief conclusion and also discuss future work.

Chapter 2

Preliminaries

In this chapter we introduce a few terminologies, definitions and notations. We also discuss some motivational problems and finally present our problem definition.

In Search Problems, a number of searchers search for an object in a search space. Starting from the same point, searchers follow their own arbitrary strategies to reach to the object which are referred as trajectories. The overall search strategy is known as *search algorithm*. Based on the nature of the problem, the termination of the algorithm differs. In this work, we only focus on search problems in which the algorithm terminates when the object is found by any of the searchers. Based on the context, there are many variations of the search problem. Some major contexts are based on the search space, prior information regarding the object, number of searchers, type of algorithm, search mechanism, objective of searches and speed of searchers. To illustrate the search mechanism and associated terminologies, we start with the following

motivating example of the classic *Linear-Search Problem*.

2.1 Motivating Example

Linear Search An object is placed on the real line at a point H. A searcher, starting from the origin O at a unit speed wishes to discover the object in minimal relative time. It is assumed that the searcher can switch direction of travel instantaneously. Consider the searcher as a robot. That means the robot starts at the origin of the real line and travels one unit of distance per one unit of time along the real line in either direction. In order to determine that there is an object at location H the robot has to be physically be present at H. In that case, what should be the strategy for the robot to explore the real line in order to find the object minimizing the total search time in worst case? This problem is known as the *linear search problem* or *the cow path problem*, which appeared initially in [12]. Later it was restudied in [9].

Suppose that the object is placed at distance d from the origin. If the robot knew that the object was placed to the right of the origin or to the left of the origin, the robot could start moving in the correct direction, finding the object in time d. This is an optimal *offline solution*. Since the robot does not know in which direction it should be moving to find the object, it needs to explore both directions. The best strategy for this is a zig-zag strategy which is an *online solution*. Initially the robot moves to the positive direction and walks for 1 unit of distance in that direction. If no object is found, the robot returns to the origin, flips the direction and doubles the

distance. These steps are repeated until the object is found.

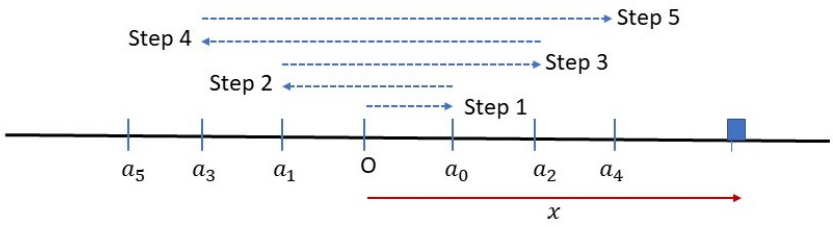


Figure 2.1: The zigzag trajectory followed by the robot. Overall relative search time minimizes when $a_i = 2^i$.

Mathematically, it has been proved that the relative search time is minimized when the robot follows the zigzag strategy in such a way that it has returned to the origin for the i'th even time that is, it moves 2^i to the right and back to the origin or 2^{i+1} to the left and back to the origin, until the object is found.

In Search Problems the underlying optimization objective is to minimize the total search time. So, the robot wants to reach the goal in the least possible time. In algorithm analysis, it is assumed that the inputs of the *online algorithm* (that performs without knowing the inputs) is viewed by an *adversary*, who wants to maximize the search time. To do that, the adversary deliberately chooses the worst placement of the object so as to maximize the search time. In this situation, the efficiency of the algorithm depends on how fast it can reach to the object by minimizing the total

relative search time in the worst case scenario. This analysis is known as *competitive analysis*. Competitive analysis is a method invented for analysing the performance of online algorithms in a worst case scenario. Here the performance of an online algorithm is compared to the performance of an optimal offline algorithm (that performs knowing the input). In other words, one compares the performance of an online algorithm relative to what can be achieved optimally with respect to all possible inputs. Now to maximize the search time, the adversary may have the choice to place the object infinitely far away (if the search space is unbounded). If it does so, then the robot will need unbounded time to reach to the object. To overcome the situation, we consider the relative cost measure. The best possible relative time is known as *competitive ratio*.

Consider ω represents the zigzag algorithm for the linear search problem. Without loss of generality, we consider the placement of the object along positive x-axis. Consider the distance of the object from the origin is x. Then the competitive ratio of the algorithm ω (denoted by $CR(\omega)$) is given by the ratio of the distance traversed by the robot to reach the object (which is known as *online cost*) to the distance x of the object from the origin (which is known as *offline cost*) for the worst placement. The mathematical representation of $CR(\omega)$ is given by,

$$CR(\omega) = \sup_{|x|>1} \frac{\text{Online cost}(\omega, x)}{\text{Offline cost}(x)}.$$

To do the competitive analysis of the Linear Search Problem, we have to consider

the worst case. This appears when the robot closely missed the object in any direction and returns to the origin while doubling the distance and travels in the wrong direction, returns to the origin, and eventually discovers the object by travelling in the right direction. Such worst case appears when the adversary places the object between the locations 2^i and 2^{i+1} for some i and arbitrarily close to 2^i. In other words, if an object is placed at a distance $x = 2^i + \epsilon > 2^i$ for $\epsilon > 0$ in direction $(-1)^i$. Now if the robot knowns the location of the object, it can directly travel distance x and discover the object. So, the offline cost of the algorithm is given by x. Following the zigzag trajectory, we have the online cost is given by

$$\begin{aligned} \textit{online cost} &= 2^0 + 2 \cdot 2^1 + 2 \cdot 2^2 + \cdots + 2 \cdot 2^i + 2 \cdot 2^{i+1} + x \\ &= 2 \sum_{j=0}^{i+1} 2^j + x. \end{aligned}$$

Thus, we have the competitive ratio $CR(\omega)$ of the algorithm condition on the object is in interval $(2^i, 2^{i+1}]$ is

$$\begin{aligned} CR(\omega) &= \frac{2 \sum_{j=0}^{i+1} 2^j}{x} + 1 \\ &= \frac{2 \sum_{j=0}^{i+1} 2^j}{2^i} + 1 \text{ [for } \epsilon \to 0] \\ &= \frac{2(2^{(i+2)} - 1)}{2^i} + 1 \\ &= 9 - 2^{(1-i)} \end{aligned}$$

For $i \to \infty$ the competitive ratio tends to 9. Thus, this doubling strategy gives a competitive ratio of at most 9.

There are two other terminologies which describe the bounds of competitive ratio more specifically. In competitive analysis, bounds have a significant role to play in determining, how much improvement is possible based on different algorithms for a certain problem. In other words, this concept helps to estimate the optimal algorithm for a problem which is discussed below.

2.2 Upper Bounds and Lower Bounds

When we propose an algorithm for solving a minimization problem and do the competitive analysis i.e, worst case analysis then we give an *upper bound*. By giving an upper bound we show how well the algorithm performs in solving the minimization problem in the worst case scenario. An upper bound of a problem guarantees that the best value possible will be no more than that.

On the other hand, *lower bound* guarantees that no algorithm can do better than that. That means the best value will be at least the lower bound.
An optimal algorithm is achieved when the condition *upper bound* = *lower bound* is satisfied by the algorithm. For this purpose, the ultimate goal is to propose an optimal algorithm that achieves the lower bound. See Figure 2.2 for how the optimal solution is approached.

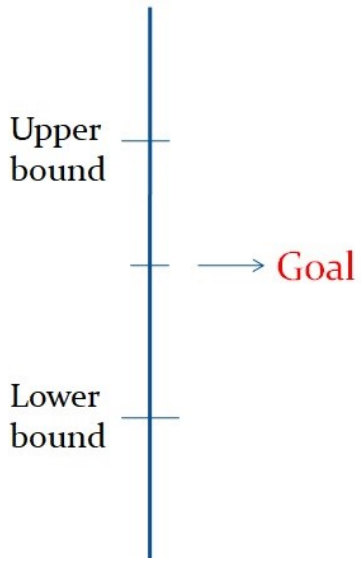


Figure 2.2: Here 'Goal' represents the optimal algorithm that lies between upper bound and lower bound

Now we will introduce the problem that we study in this thesis. Our problem is a two dimensional search problem, where a number of unit speed searchers starting from a point search for a line in the plane. Here, we refer to searchers as robots and the line is referred as the *shoreline*. We will analyse our problem from the perspective of analytic geometry, which means the robots will start from the origin of the Cartesian plane and the trajectories followed by the robots will be analytic curves in the plane. We call our problem $Shoreline_n$. Below we present the problem definition.

Definition 2.2.1. *In the $Shoreline_n$ problem we provide an algorithm for n unit speed robots starting from the origin search for a given line l in the plane by moving along their own fixed trajectories τ_n. The algorithm terminates when any trajectory hits the line l for the first time. The objective of the problem is to find trajectories τ_n to minimize the competitive ratio: if $T_{\tau_n}(l)$ is the time by which l is discovered for the*

first time by any of the robots and $\delta(l)$ is the distance of the line to the origin, then the search competitive ratio is defined as

$$CR(\tau_n) := \sup_l \frac{T_{\tau_n}(l)}{\delta(l)}$$

The best possible search completion ratio will be denoted by S_n given by $\inf_{\tau_n} CR(\tau_n)$.

2.3 Our Contributions

Our main contribution in this paper is the improvements of lower bounds of two dimensional search problem with two or more robots. We improve the best lower bound known [8] for $n = 2$ robots from 1.5993 to 3 . Also, we prove the first lower bound for $n = 3$ which is $\sqrt{3}$. For $n \geq 4$ we prove the lower bound $\frac{1}{\cos(\pi/n)}$ which matches the best upper bound known. Besides this, we show the complete calculations of the upper bounds for $Shoreline_n$ for $n \geq 2$. To the best of our best knowledge, these calculations have not appeared anywhere before.

2.4 A Few Notations

Throughout the paper, we will use the following notation frequently-

- The number of robots is denoted as n.
- Our problem is defined as $Shoreline_n$, where n represents the number of robots associated to the problem.

- The trajectories are denoted as τ_n .
- Competitive ratio is denoted as CR of the search problem.
- The best possible competitive ratio is denoted by S_n.
- We refer to the shoreline as l.
- The robots are denoted as R_i where $i = 1, \ldots, n$.

Chapter 3

Some Positive Results

In this chapter we review some known results of searching for a shoreline with two and more robots which appeared in [8]. Even though the results were known, only the values were reported and to the best of our knowledge, calculations have not been shown before.

3.1 Searching with Two Robots

The best algorithm known for $Shoreline_2$ problem is the *double logarithmic spiral*, where two unit speed robots starting from the origin follow two logarithmic spiral trajectories.

Theorem 3.1.1. $S_2 \leq 5.27$.

Now we will discuss the algorithm for the problem $Shoreline_2$ and prove necessary claims and lemmata and finally prove Theorem 3.1.1 .

The best algorithm known for searching for a shoreline with two robots is the double-spiral trajectory traversed by the two robots. We are considering $n = 2$ robots starting from the origin moving along two logarithmic spirals with an angle difference π with parameter b where algorithm can choose the best b minimizing the worst case competitive ratio. For $t \in (-\infty, +\infty)$, R_1 and R_2 follow the trajectories respectively as follows.

$$R_1(t) = \exp(bt)(\cos(t), \sin(t)).$$

and

$$R_2(t) = \exp(bt)(-\cos(t), -\sin(t)).$$

In what follows we use abbreviation $t_k := 2\pi k$ where $k \in \mathbb{N}$, which represents the angle of the polar coordinate (not the time) where k represents the number of cycles.

Now, we will focus on the shorelines that are discovered first by R_2 for a fixed $t = t_k$.

The shoreline can be placed anywhere in the plane. Now for a fixed t, for which $R_2(t)$ hits the line first, the worst case happens when the shoreline gets closer and closer to the origin as long as $R_1(t)$ does not find it first. This situation appears when the shoreline is just missed by the $R_1(t)$ while $R_2(t)$ found it.

Fix t' and consider all shorelines passing through $R_2(t')$ that intersect $R_1(t)$ for $t > t'$ for the first time. Among all these lines the one inducing the competitive ratio

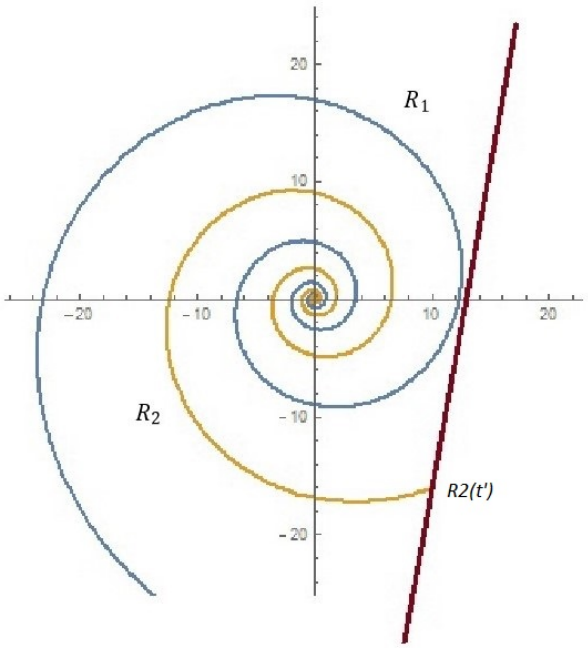


Figure 3.1: Double Spiral trajectories, where R_1 closely missed the shoreline and R_2 found it later.

is when the shoreline becomes (nearly) tangent to spiral R_1 at an angle say, τ. That means R_1 miss the shoreline at $R_1(\tau)$ and R_2 finds it for the first time at t' where, $t' = \tau + \phi$ for the smallest ϕ, $0 < \phi < 2\pi$.

Now by symmetry and rotating the spiral, we can make $R_1(\tau)$ be on the positive $x = axis$. Considering this, we assume that $R_1(t)$ missed it at the point $t_k := (\exp(2\pi kb), 0)$ and R_2 found it for the first time at the point $(t_k + \phi) = (\exp(b(2\pi k + \phi))((-\cos(2\pi k + \phi), -\sin(2\pi k + \phi))$ for the smallest $\phi > 0$.

Therefore the competitive ratio of the algorithm is given by the ratio between the distance traversed by R_2 to find the shoreline and the optimal distance which is the distance to the origin.

To find the Competitive ratio for this algorithm we need the following two claims.

Claim 3.1.2. *The distance of the tangent $R_1(t)$ at $t = t_k$ to the origin is* $\dfrac{\exp(2\pi kb)}{\sqrt{(1+b^2)}}$.

Proof. The trajectory of a logarithmic spiral is given by

$$x = \exp(bt)\cos(t).$$

$$y = \exp(bt)\sin(t).$$

So, the slope of the spiral at (x, y) is given by

$$\begin{aligned}
\delta &= \frac{dy}{dx} \\
&= \frac{\frac{dy}{dt}}{\frac{dx}{dt}} \\
&= \frac{b\exp(bt)\sin(t) + \exp(bt)\cos(t)}{b\exp(bt)\cos(t) - \exp(bt)\sin(t)} \\
&= \frac{by + x}{bx - y}.
\end{aligned}$$

So the slope to $R_1(t)$ at $t = t_k$ is given by

$$\delta = \frac{\exp(2\pi b)}{b\exp(2\pi b)} = \frac{1}{b}.$$

Consider the tangent equation

$$y = \frac{1}{b}x + c.$$

This line passes through the point $(\exp(2\pi kb), 0)$ which gives $c = -\dfrac{\exp(2\pi kb)}{b}$. Thus the equation of the tangent line to the first spiral is

$$x - by - \exp(2\pi kb) = 0. \tag{3.1}$$

Using the known equation of the distance of lines to points, we have the distance d from the origin to the tangent line (3.1) is given by

$$d = \frac{\exp(2\pi kb)}{\sqrt{(1+b^2)}}.$$

□

Now we find the total time taken by R_2 to discover the line, in other words, the total distance traversed by R_2 will be the arc length of the second spiral upto $t = t_k$.

From basic calculus, we know the formula stated in the following lemma.

Lemma 3.1.3. *The arc-length of the spiral $r = \exp(b\theta)$ upto t is $\dfrac{\sqrt{(1+b^2)}}{b}\exp(bt)$.*

Claim 3.1.4. *The total time that R_2 needs to hit the tangent line to $R_1(t)$ at $t = t_k = 2\pi k$ for the first time is $\dfrac{\sqrt{(1+b^2)}}{b}\exp(b(2\pi k + \phi))$.*

Proof. If R_2 found the shoreline at point $(\exp(b(2\pi k+\phi))((-\cos(2\pi k+\phi), -\sin(2\pi k+\phi))$ for the smallest $\phi > 0$ then, it will satisfy equation (3.1).

So we have,

$$-\exp(b\phi)\cos(\phi) + b\exp(b\phi)\sin(\phi) - 1 = 0 \tag{3.2}$$

$$\Rightarrow b\sin(\phi) = \cos(\phi) = \exp(-b\phi). \tag{3.3}$$

Now by Lemma 3.1.3, we have the length of the trajectory of R_2 that is the total time that R_2 needs to discover the shoreline for the first time is given by

$$\frac{\sqrt{(1+b^2)}}{b}\exp(b(2\pi k + \phi)).$$

□

Notice, Claim 3.1.2 and 3.1.4 represent respectively the offline and online cost. Also by solving equation (3.3) we have the value of ϕ as a function of b. Hence,the competitive ratio of the algorithm is given by,

$$\frac{1+b^2}{b}\exp(b\phi).$$

Using Mathematica we have found that the optimal competitive ratio for $n = 2$ is $S_2 \leq 5.27$ attained for $b = (0.61, 0.7)$.

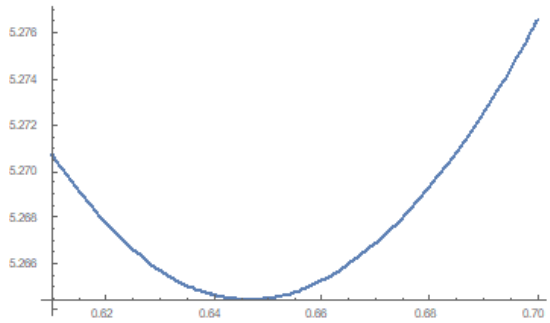


Figure 3.2: Mathematica output figure, showing the competitive ratio for $n = 2$.

3.2 Searching with Three or more Robots

For $n \geq 3$ the best algorithm known is the ray algorithm which makes robots move along rays at a unit speed splitting the plane evenly while searching for the shoreline. Therefore, the angle between two rays is $\frac{2\pi}{n}$, where n represents the total number of robots.

We consider, R_1 moves along the horizontal axis following the trajectory $R_1(t) = t \cdot (\cos \frac{(i-1)2\pi}{n}, \sin \frac{(i-1)2\pi}{n})$, $\forall\, i = 1, \ldots, n$ where t represents time. Our goal is now to do the competitive analysis for this algorithm. Without loss of generality, we focus on the lines x away from origin (See Figure 3.3) first found by R_1 and then by R_2. Now we will analyse what could be the possible placements of the shoreline to make sure that no other robot found it before R_1.

In Figure 3.3, we are considering R_1 moves along OA and discovers the line l in time d_1 at the point A and R_2 moves along OB and discovers the line l in time d_2 at the point B. Also γ is defined as the angle that the normal of l forms with the horizontal axis. Now R_1 discovers the line before R_2 is the same as to characterize $d_1 < d_2$ which is shown in the following claim.

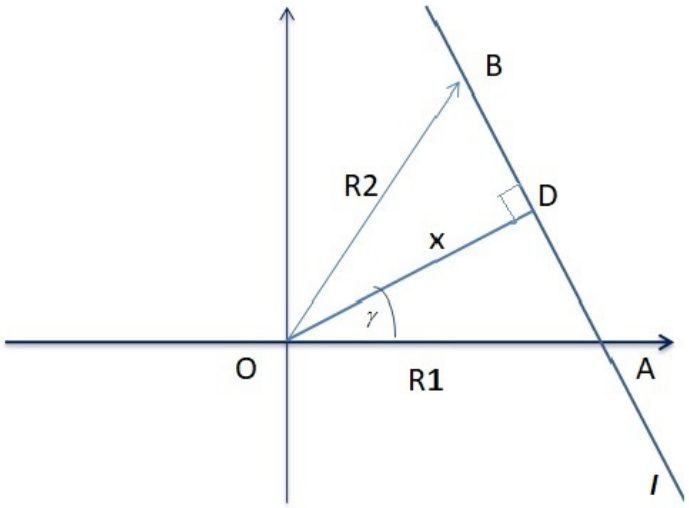


Figure 3.3: Ray trajectories

Claim 3.2.1. *Among lines l found first by R_1 and R_2, R_1 finds l no later than R_2 exactly when $\gamma \leq \frac{\pi}{n}$.*

Proof. If $\gamma = 0$ then R_1 finds it first. Now, from $\triangle OAD$ in Figure 3.3, we have $\cos(\gamma) = \frac{x}{d_1}$ and $\cos(\frac{2\pi}{n} - \gamma) = \frac{x}{d_2}$. So,we have, $\frac{d_1}{d_2} = \frac{\cos(\frac{2\pi}{n} - \gamma)}{\cos(\gamma)} < 1$ when $\gamma < \frac{\pi}{n}$ which implies $d_1 < d_2 \Leftrightarrow \gamma < \frac{\pi}{n}$. □

Now we are ready to find the corresponding competitive ratio of the ray algorithm for $n \geq 3$ by proving the following theorem.

Theorem 3.2.2. *The Competitive ratio for Shoreline$_n$ where $n \geq 3$ is at most* $\frac{1}{\cos(\frac{\pi}{n})}$.

Proof. Consider the algorithm runs for d time and by symmetry we assume that R_1 finds the shoreline at time d which represents the online cost. Consider γ is the angle formed by the normal of l and the horizontal axis. If the angle γ is given, then from Figure 3.3 we have , $x = d\cos(\gamma)$ which represents the offline cost. Say, τ_n represents the ray algorithm for $n \geq 3$. Hence, competitive ratio of the ray algorithm for $n \geq 3$ is given by, $CR(\tau_n) = \sup\limits_{0 \leq \gamma \leq \frac{\pi}{n}} \dfrac{d}{d\cos(\gamma)} = \sup\limits_{0 \leq \gamma \leq \frac{\pi}{n}} \dfrac{1}{\cos(\gamma)}$.

Now the worst case scenario appears when the highest value of γ is substituted which is $\frac{\pi}{n}$. So we have, $CR(\tau_n) \leq \dfrac{1}{\cos(\frac{\pi}{n})}$. □

Chapter 4

Lower Bounds for Shoreline_n , $n \geq 2$

In this chapter, we analyse the lower bounds for the problem $Shoreline_n$ with $n \geq 2$ where for every n we identify threshold values that cannot be beaten by any other algorithm. At first we will show the lower bound for $Shoreline_n$ when $n \geq 4$ and later we will show the lower bounds for S_n with $n = 3$ and $n = 2$ respectively.

4.1 Lower Bounds for $n \geq 4$ Robots

In this section our main objective is to prove the following theorems which is split in a number of lemmata.

Theorem 4.1.1. *For $n \geq 4$ we have $S_n \geq \dfrac{1}{\cos(\frac{\pi}{n})}$.*

To prove this theorem we need to prove the following lemmata.

Lemma 4.1.2. *Consider the right triangle BMO with $\angle BOM \leq \frac{\pi}{4}$ and $\angle BMO = \frac{\pi}{2}$. Then for any point L on the line segment OB and any point K on the line segment BM, we have $OK + KL \geq OB$.*

Proof. To prove this at first we take a reflection of the triangle OBM around the line BM which is triangle $O'BM$. Here O' is the reflection of the point O around BM.

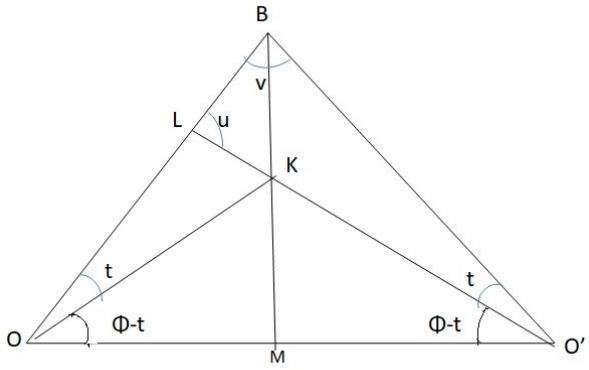


Figure 4.1: Triangle OBM and its reflection.

Consider any point L on the line segment OB and choose the point $K = K(L)$ on the

line segment BM that minimizes $OK + KL$ which applies when $\angle OKM = \angle LKB$. We notice that for the shortest trajectory O', K, L are co-linear. Now it is enough to show that $O'K + KL > O'B$ to prove $OK + KL > OB$. In Figure 4.1 we refer to $\angle BOM = \angle BO'M = \phi$, $\angle BOK = \angle BO'K = t$. Also consider $\angle O'BL = v$ and $\angle O'LB = u$.

From the triangle OBO' we have $\angle BOO' + \angle BO'O = 2\phi$. That means $\angle OBO' = v = \pi - 2\phi$. Also, from the triangle BLO' , we get, $\angle BLO' = u = \pi - t - v = 2\phi - t$.

Now, $v - u = \pi + t - 4\phi$. But as $\phi \leq \frac{\pi}{4}$, we observe that, $v - u \geq t \geq 0$. So, we have $v > u$. Also from the triangle $BO'L$ we can say see that $O'L > O'B$ as $v > u$.

That means, $O'K + KL > O'B$ which is by the law of reflection same as to say $OK + KL \geq OB$ which completes the proof. □

In the next lemma, we show that for a specific time bound and under certain conditions, there are lines which have not been discovered by the robots irrespective of the trajectories.

Lemma 4.1.3. *Consider a cone of angle 2ϕ centred at origin, where $\phi \leq \frac{\pi}{4}$. Also, consider A and B are two points on the extreme rays of the cone, such that $OA = OB = d + \epsilon$, for some $\epsilon > 0$. Consider a trajectory τ_n such that the robots start from the origin O at an unit speed and run for a fixed time $d > 0$. If at time d there is no robot inside the cone, then the line l passing through the points A, B could not have been discovered by any of the robots .*

Proof. We prove it by contradiction. Assume that a robot being outside of the cone at time d has found the line l passing through the points A and B at time d.

From Figure 4.2 we notice that, as the robot discovers the line l passing through

A, B at time d, the possible trajectories τ_n could have touched line l for the first time and intersected either of the line OA or OB to be outside of the cone.

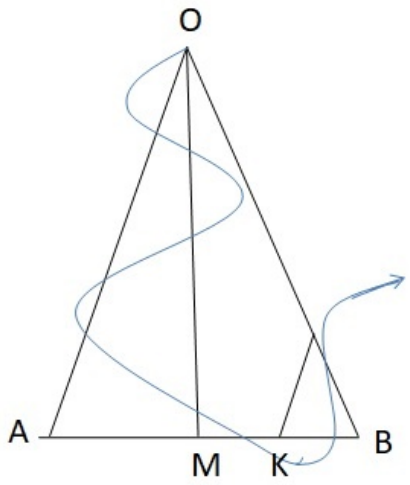


Figure 4.2: Cone OAB of Lemma 4.1.3. Robot's arbitrary trajectory is shown by the curved line.

Now without loss of generality, the robot could have touched the line l at some interior point K of AB and intersected line OB at the point L which is close to the point K to get outside of the cone. For such a trajectory the minimum time required would be $OK + KL$ to touch the line l and being outside of the cone, (see Figure 4.2).

We notice that the triangle OAB is an isosceles as $OA = OB$. Here M is the projection of the point O onto AB. So, $\angle BOM = \phi$ as $\angle AOB = 2\phi$. But then by Lemma 4.1.2 we have that $OK + KL > OB$. So the time required by the unit speed robot to find the line is $OK + KL > OB = d + \epsilon > d$, which contradicts the claim as the time d has already passed. □

Next, we focus on the lower bound to the competitive ratio for the $Shoreline_n$ problem where $n \geq 4$.

Lemma 4.1.4. *Consider trajectories τ_n ,where robots starting at the origin at an unit speed run for time $d > 0$. Considering a cone of angle 2ϕ (where $\phi \leq \frac{\pi}{4}$) centred at origin within which if there is no robot at time $d > 0$ then $CR(\tau_n) \geq \frac{1}{\cos(\phi)}$.*

Proof. Consider there is no robot at time $d > 0$ inside the cone centred at the origin O with an angle 2ϕ. Also consider there are the two points A and B on the two extreme rays of the cone which are $d + \epsilon$ distant away from the origin. According to Lemma 4.1.3 no robot has discovered the line l passing through the points by the time $d > 0$ under this condition. Since time d has passed we see that the search completion time required by the robot is greater than time d.

Also, as $\triangle OAB$ is an isosceles triangle, we have the distance of the line l passing through A, B is given by, $OM = (d + \epsilon) \cdot \cos(\phi)$ which represents the optimal offline time. So, we have that,

$$CR(\tau_n) \geq \sup_{\epsilon > 0} \frac{d}{(d + \epsilon)\cos(\phi)} = \frac{1}{\cos(\phi)}$$

. □

Now, even if we consider the case, when there is a robot lying in the origin inside the cone, till then the robot requires at least time $(d + \epsilon)\cos(\phi)$ to find the line l. So the above mentioned bound is true for this case as well.

Now we are ready to prove Theorem 4.1.1 based on these three lemmata.

Proof. For a fixed $n \geq 4$ consider the trajectories τ_n where the unit speed robots start from the origin and move for an arbitrary time $d > 0$. Now if all the robots are in the origin, then the competitive ratio will be unbounded. So, we assume, there is a robot which is not in the origin . Also, consider an arbitrary small cone of an angle $\delta << \phi$ ($\delta = o(\frac{1}{n})$), $\phi = \frac{\pi}{n}$. We then rotate the small cone of angle δ until a robot falls within the cone. Then we consider covering the rest of the space by n cones centred at the origin of angle $\frac{2\pi-\delta}{n}$ and hence one of which there is no robot. Now by Lemma 4.1.4 we have for any $\delta > 0$, $2\phi = \frac{2\pi-\delta}{n}$. So,we have $CR(\tau_n) \geq \frac{1}{\cos(\frac{\pi}{n}-\delta)}$, hence, $S_n \geq \frac{1}{\cos(\frac{\pi}{n})}$. □

4.2 Lower Bound for $n = 3$ Robots

Now we will analyse the lower bound for $Shoreline_3$. For this we will prove the following theorem.

Theorem 4.2.1. $S_3 \geq \sqrt{3}$.

We begin with the investigation of whether the lower bound argument for four or more robots works for $n < 4$ robots or not. Notice, in Lemma 4.1.2 where we showed $OK + KL > OB$, this argument fails when $\phi = \frac{\pi}{3}$. This means that if we consider a robot starting from the origin of a cone of angle 2ϕ, $\phi = \frac{\pi}{3}$, then the robot may touch the line and come out of the the cone within time 1. Hence, the previous lower bound argument does not work for $n < 4$. However, though this argument fails we

can still calculate the time required by the robot to touch the line within the cone and come out of the cone. Based on that, later on we may rescale the cone so that the robot cannot manage to touch the line and come out of the cone within the given time. In the following Lemma, we will find the time that robot requires to discover the line when $n = 3$.

Lemma 4.2.2. *Consider a cone of angle $2\pi/3$ centered at the origin O, along with two points A, B on its extreme rays at distance 1 from O. Then, a unit speed robot starting from the origin O requires at least time $\sqrt{3}/2$ to visit the line passing through A, B and leave the cone.*

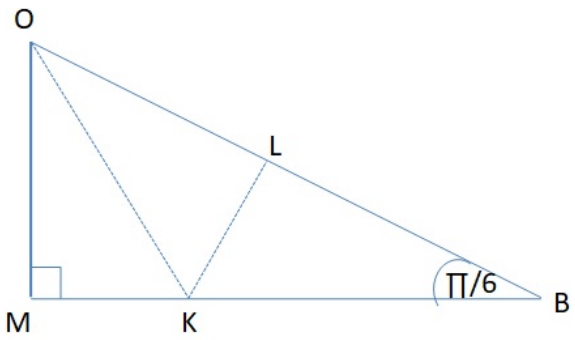


Figure 4.3: Cone OMB of Lemma 4.2.2. Robot's shortest trajectory is shown by the dotted line.

Proof. Let M be the projection of origin O on the line l passing through the points A, B. The distance of the points A and B from the origin is 1. The shortest trajectory of the robot is calculated as starting from the origin, hitting the line segment AB at K and leaving the cone through the line segment OB at the point L, (See Figure 4.3). We consider a coordinate system centered at M for convenience. In Figure 4.3 in the

right triangle OMB, $\angle BOM = \pi/3$, $\angle OBM = \pi/6$ and $OB = 1$. Considering M as the origin we have $O = (0, 1/2)$ and $B = (\sqrt{3}/2, 0)$. Let K be any arbitrary point on the line segment MB which is in other words a convex combination of the points M and B. Hence, the coordinate of K is $\lambda(\sqrt{3}/2, 0)$, for some $\lambda \in [0, 1]$. Now given that K is chosen, the shortest trajectory for the robot to leave the cone, is through the point L on the line segment OB which is given by $y + \sqrt{3}/3x - 1/2 = 0$. So, the shortest trajectory for the unit speed robot starting from the origin, to touch the line l and leave the cone is $\min\limits_{\lambda\in[0,1]} OK + KL$. We calculate

$$\begin{aligned} |OK| &= \sqrt{\frac{3}{4}\lambda^2 + \frac{1}{4}} \\ &= \frac{1}{2}\sqrt{3\lambda^2 + 1}. \end{aligned}$$

Also the distance from point K to line OB is given by

$$\begin{aligned} |KL| &= \frac{|\frac{\sqrt{3}}{3}\lambda\frac{\sqrt{3}}{2} - \frac{1}{2}|}{1 + \frac{3}{9}} \\ &= \frac{\sqrt{3}}{4}(1 - \lambda). \end{aligned}$$

Thus, we have,

$$\min_{\lambda\in[0,1]}[OK + KL] = \min_{\lambda\in[0,1]}[\frac{1}{2}\sqrt{3\lambda^2 + 1} + \frac{\sqrt{3}}{4}(1 - \lambda)].$$

Consider the latter function as $f(\lambda)$. To find the minimum of this function of λ, at first we find the derivative of the function as $f'(\lambda) = \frac{3\lambda}{2\sqrt{3\lambda^2+1}} - \frac{\sqrt{3}}{4}$, which has a unique

root at $\lambda_0 = 1/3$. Then we have, $f''(1/3) = \frac{9\sqrt{3}}{16}$. So, the minimum of the function $f(\lambda)$ is $f(1/3) = \frac{\sqrt{3}}{2}$. Thus the minimum time required by the unit speed robot to touch the line l and leave the cone is $\frac{\sqrt{3}}{2}$. □

Now we are ready to prove Theorem 4.2.1. The main idea behind the proof is to rescale the triangle $\triangle OMB$ in such a way so that a unit speed robot cannot touch the line l and leave the cone in time 1.

Proof. We are considering trajectories τ_3 for the problem $Shoreline_3$ where the unit speed robots start from the origin and move for an arbitrary time $d > 0$. Consider 3 cones, each of angle $2\pi/3$ centered at the origin and covering the entire plane. Now without loss of generality we may assume that one robot lies at one of the extreme rays of one cone. Therefore, there exists one cone, call it C with no robot in its interior. Now, consider two points A and B on the extreme rays of the cone C which are d away from the origin.

From Lemma 4.2.2 we found that if the distance $OB = 1$ then the robot requires time at least $\sqrt{3}/2$. So, if the robot has time d then by rescaling we have the distance $OB = \frac{2}{\sqrt{3}}d$. Now if we set the points A, B on extreme rays at a distance $(\frac{2}{\sqrt{3}} + 2\epsilon)d$ away from the origin O of the cone and let the line l_{AB} pass through the points A, B, then the robot will not have enough time to touch the line and leave the cone by time $d > 0$. Note that l_{AB} is exactly $(\frac{1}{\sqrt{3}} + \epsilon)d$ away from the origin.

Now, by Lemma 4.2.2 and since no robot lies within the cone C, we can say that no robot could have discovered line l_{AB} as the search completion time is at least d. Hence line l_{AB} induces competitive ratio for τ_3 is at least $\frac{1}{(\frac{1}{\sqrt{3}}+\epsilon)}$, for every $\epsilon > 0$. Thus we conclude that $S_3 \geq \sqrt{3}$. □

4.3 Lower Bound for $n = 2$ Robots

For $n = 2$ robots the lower bound S_2 is given by the following theorem.

Theorem 4.3.1. $S_2 \geq 3$.

The following function represents the boundary of an ellipse which will be useful to prove Theorem 4.3.1.

$$q(x, y, \delta, \theta) := 4(\cos(\theta)x + \sin(\theta)y - h_\delta)^2 + (-\sin(\theta)x + \cos(\theta)y)^2/b_\delta^2 - 1$$

where $h_\delta := \delta/2$, represents the abscissa of the center of the ellipse and $b_\delta := \sqrt{(1-\delta^2)}/2$ represents the half of the major axis of the ellipse. The details of the above equation of ellipse is discussed in the following lemma.

Lemma 4.3.2. *Consider an arbitrary algorithm τ_2 runs for time 1. Then there exist ϵ, $\delta \in [0, 1]$ and $\theta \in [0, \pi]$ so that no point outside the ellipses $q(x, y, \epsilon, \theta) \leq 0$ and $q(x, y, \delta, \theta) \leq 0$ has been explored by any robot.*

Proof. Let the two unit speed robots starting from the origin, run an arbitrary algorithm τ_n for time 1. Without loss of generality, suppose that both robots lie in the first two quadrants i.e, in the non-negative y-axis half-plane. Without loss of generality, suppose the location of R_1 is at the point $(\epsilon, 0)$ for some, $\epsilon \in [0, 1]$ at time exactly 1. Therefore, all points P that R_1 has visited by time 1 satisfy $OP + PR_1 \leq 1$. This means, the boundary of all points that could have been explored by R_1 is an ellipse with foci O, R_1. Thus, all boundary points $(x, y) \in \mathbb{R}^2$ that could have discovered by R_1 satisfy $4(x-h_\epsilon)^2+y^2/(b_\epsilon)^2 = 1$, where $(h_\epsilon, 0)$ is the center of the ellipse.

Similarly, R_2 is at a distance δ at the same time 1 where $\delta \in [0, 1]$. By the same arguments as before, boundary of the explored domain by R_2 is again an ellipse. Now if the line passing through the two foci of the second ellipse form an angle θ with the x-axis where $\theta \in [0, \pi]$, then all points $R_2 = (\delta\cos(\theta), \delta\sin(\theta))$ explored by R_2 is given by $4(\cos(\theta)x + \sin(\theta)y - h_\delta)^2 + (-\sin(\theta)x + \cos(\theta)y)^2/b_\delta^2 = 1$. Note that, as R_2 lies in the first two quadrants then we must have $\theta \in [0, \pi]$. □

Now, we are ready to prove Theorem 4.3.1.

Proof. Consider the two robots are following an arbitrary search algorithm τ_2 for time 1. According to Lemma 4.3.2 all the points that have been explored by these two robots are defined by the two ellipses given there. This means that R_1 and R_2 have not explored any point outside the two ellipses placed in the first two quadrants. That

is, they could not have been in any point past the line $y = -1/2$.

Our main claim is that if a shoreline $y = -1/2 - \zeta$, where $\zeta > 0$ has been placed in the negative y-axis half-plane, then none of the robots could have visited the line. To prove this, we need to show that none of the equations defining any of the two ellipses has any common point with $y = -1/2 - \zeta$, (see Figure 4.4).

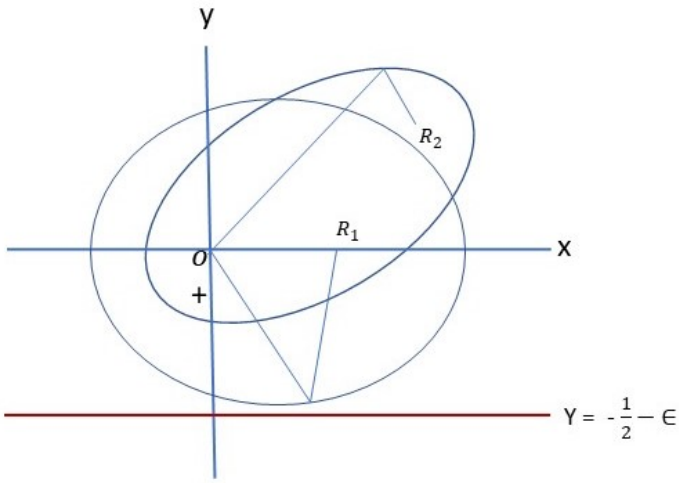


Figure 4.4: Two ellipses and possible placement of the shoreline.

To that end, we show that the equation $q(x, -1/2 - \zeta, \delta, \theta) = 0$ has no real root when $\delta \in [0, 1]$, $\theta \in [0, \pi]$ and $\zeta > 0$ is sufficiently small. This would also imply same for the first ellipse. We compute the discriminant of $q(x, y, \delta, \theta)$ which is

$$\frac{-16(\delta^2 + 2\delta(2\zeta + 1)sin(\theta) + 4\zeta(\zeta + 1)}{1 - \delta^2} \leq \frac{-16(\delta^2 + 4\zeta(\zeta + 1)}{1 - \delta^2}.$$

For arbitrary small $\delta > 0$ this expression is maximized for $\delta = 0$ and becomes $-64(\zeta^2 + \zeta)$ which is also negative.

Notice, the distance of the shoreline from the origin is $1/2+\zeta$ which is the optimal offline cost. As time 1 has passed, the time required for R_1 is at least $3/2 + \zeta$ which implies a bound to the online cost. Hence, the competitive ratio of the arbitrary search algorithm τ_2 is at least

$$\sup_{\zeta>0} \frac{3/2 + \zeta}{1/2 + \zeta} = 3.$$

□

Chapter 5

Conclusion

In this work we studied a two-dimensional search problem where n unit speed robots search for a line (Shoreline) in parallel on the plane. The problem was referred to as $Shoreline_n$. We studied the existing upper bounds for $n \geq 2$. The main contributions were new lower bound results for $n \geq 2$. More specifically, we improved the lower bound for $n = 2$ from 1.5993 to 3. We proved the first lower bounds for $n \geq 3$. More specifically we proved a lower bound of $\sqrt{3}$ for $n = 3$ and $\frac{1}{\cos(\pi/n)}$ for $n \geq 4$. For $n \geq 4$, we found the lower bound $\frac{1}{\cos(\pi/n)}$ which matches the best upper bound known, making the bound tight.

5.1 Future Work

In this work we have addressed upper bound and lower bounds of $Shoreline_n$ problem for $n \geq 2$. But the $n = 1$ case is still an open problem. For $n = 2, 3$ though we presented the lower bounds, the algorithms that can achieve those bounds are yet to

be explored. In fact, the existing algorithm known best for $n = 1, 2$ has yet to be proved to be an optimal algorithm. For $n = 3$ the upper bound is achieved by the same algorithm as for $n \geq 4$ but the lower bound conditions differ. Besides these, there are other variations of $Shoreline_n$ problem in terms of different perspectives, such as analysis on symmetric and asymmetric algorithms, average case and worst case analysis for different n.

The other two variations other than the search problem that are commonly considered are the evacuation and rendezvous problem. For a number of robots, if they want to evacuate from the line, there would be different analysis based on communication methods that is face to face and wireless model. Also, there would be different analysis for cases where they want to evacuate at a specific point on the line or from anywhere in the line. In terms of the starting point, there could be possible variations if the robots start from different points and want to meet at a specific point on the line.

Finally, we can conclude that, based on the nature of the problems there would be different analysis in terms of the communication methods, number of robots, search spaces, different efficiency measures etc.

Appendix A

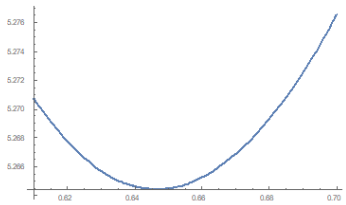
Mathematica Code

Below is the Mathematica code for computing the competitive ratio of the double logarithmic spiral algorithm.

```
In[•]:= test[b_] := phi /. FindRoot[
          Exp[-b * phi] == b * Sin[phi] - Cos[phi], {phi, 1}
        ][[1]]

In[•]:= cr[b_] := (b^2 + 1) / b * Exp[test[b] * b]

In[•]:= Plot[ cr[b], {b, .61, 0.7}]
```



References

[1] S. Alpern and S. Gal. *The theory of search games and rendezvous*. Springer, 2003.

[2] Steve Alpern. The rendezvous search problem. *SIAM Journal on Control and Optimization*, 33(3):673–683, 1995.

[3] Steve Alpern, Robbert Fokkink, L Gasieniec, Roy Lindelauf, and VS Subrahmanian. *Search theory*. Springer, 2013.

[4] Steve Alpern and Shmuel Gal. *The theory of search games and rendezvous*, volume 55. Springer Science & Business Media, 2006.

[5] Steven Alpern. Hide and seek games. In *Seminar, Institut für höhere Studien, Wien*, volume 26, 1976.

[6] Edward J Anderson and RR Weber. The rendezvous problem on discrete locations. *Journal of Applied Probability*, 27(4):839–851, 1990.

[7] Ricardo Baeza-Yates. Searching: an algorithmic tour. *Encyclopedia of Computer Science and Technology*, 37:331–359, 1997.

[8] Ricardo Baeza-Yates and René Schott. Parallel searching in the plane. *Computational Geometry*, 5(3):143–154, 1995.

[9] Ricardo A Baeza-Yates, Joseph C Culberson, and Gregory JE Rawlins. Searching with uncertainty. In *Scandinavian Workshop on Algorithm Theory*, pages 176–189. Springer, 1988.

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

[11] Anatole Beck. On the linear search problem. *Israel Journal of Mathematics*, 2(4):221–228, 1964.

[12] Richard Bellman. An optimal search. *Siam Review*, 5(3):274, 1963.

[13] Stanley J Benkoski, Michael G Monticino, and James R Weisinger. A survey of the search theory literature. *Naval Research Logistics (NRL)*, 38(4):469–494, 1991.

[14] Anthony Bonato, Konstantinos Georgiou, Calum MacRury, and Pawel Pralat. Probabilistically faulty searching on a half-line. *arXiv preprint arXiv:2002.07797*, 2020.

[15] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 2005.

[16] Sébastien Bouchard, Yoann Dieudonné, Andrzej Pelc, and Franck Petit. Deterministic treasure hunt in the plane with angular hints. In *29th International*

Symposium on Algorithms and Computation, ISAAC 2018, volume 123, pages 48–1. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018.

[17] Jurek Czyzowicz, Stefan Dobrev, Konstantinos Georgiou, Evangelos Kranakis, and Fraser MacQuarrie. Evacuating two robots from multiple unknown exits in a circle. *Theoretical Computer Science*, 709:20–30, 2018.

[18] Jurek Czyzowicz, Konstantinos Georgiou, Evangelos Kranakis, Fraser MacQuarrie, and Dominik Pajak. Fence patrolling with two-speed robots. In *ICORES*, pages 229–241, 2016.

[19] Jurek Czyzowicz, Kostantinos Georgiou, and Evangelos Kranakis. Group search and evacuation. In *Distributed Computing by Mobile Entities*, pages 335–370. Springer, 2019.

[20] James M Dobbie. A survey of search theory. *Operations Research*, 16(3):525–537, 1968.

[21] Yehuda Elmaliach, Noa Agmon, and Gal A Kaminka. Multi-robot area patrol under frequency constraints. *Annals of Mathematics and Artificial Intelligence*, 57(3-4):293–320, 2009.

[22] Yuval Emek, Tobias Langner, David Stolz, Jara Uitto, and Roger Wattenhofer. How many ants does it take to find the food? *Theoretical Computer Science*, 608:255–267, 2015.

[23] Steven R Finch and Li-Yan Zhu. Searching for a shoreline. *arXiv preprint math/0501123*, 2005.

[24] Shmuel Gal. Search games. *Wiley Encyclopedia of Operations Research and Management Science*, 2010.

[25] Brian Gluss. An alternative solution to the "lost at sea" problem. *Naval Research Logistics Quarterly*, 8(1):117–122, 1961.

[26] Brian Gluss. The minimax path in a search for a circle in a plane. *Naval Research Logistics Quarterly*, 8(4):357–360, 1961.

[27] Bruce Hoppe and Éva Tardos. Polynomial time algorithms for some evacuation problems. In *SODA*, volume 94, pages 433–441, 1994.

[28] JR Isbell. An optimal search pattern. *Naval Research Logistics Quarterly*, 4(4):357–359, 1957.

[29] Artur Jeż and Jakub Łopuszański. On the two-dimensional cow search problem. *Information Processing Letters*, 109(11):543–547, 2009.

[30] Elmar Langetepe. On the optimality of spiral search. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1–12. SIAM, 2010.

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

[32] Tobias Langner, Barbara Keller, Jara Uitto, and Roger Wattenhofer. Overcoming obstacles with ants. In Emmanuelle Anceaume, Christian Cachin, and

Maria Gradinariu Potop-Butucaru, editors, *International Conference on Principles of Distributed Systems (OPODIS)*, volume 46 of *LIPIcs*, pages 9:1–9:17. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

[33] C. Lenzen, N. Lynch, C. Newport, and T. Radeva. Trade-offs between selection complexity and performance when searching the plane without communication. In *Proceedings of the Symposium on Principles of Distributed Computing (PODC)*, pages 252–261, 2014.

[34] A. López-Ortiz and G. Sweet. Parallel searching on a lattice. In *Proceedings of the Canadian Conference on Computational Geometry (CCCG)*, pages 125–128, 2001.

[35] Andrzej Pelc. Reaching a target in the plane with no information. *Information Processing Letters*, 140:13–17, 2018.