SCATTER SEARCH ON A DISK

by

Nisha Chopra

BSc, Ryerson University, 2017

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, 2019

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

SCATTER SEARCH ON A DISK

MSc, 2019

Nisha Chopra

Applied Mathematics

Ryerson University

Consider a unit disk with two objects at unidentified locations. We examine the problem of two or more robots in search of both objects in the wireless communication model. We begin with two robots and both are needed to carry an object. Subsequently, we design several algorithms that describe robots trajectories in search of the objects. We were able to achieve a minimum worst-case search time of 6.7518 and a lower bound of $3 + \frac{\pi}{2}$.

Additionally, we define two general cases and bound the worst-case search time for both. The first of the cases is for $n \geq 3$ robots and an object can be moved by one robot. The second case is where we have $n \geq 3$ robots and two robots are needed to carry an object. We achieve an upper bound of $1 + \frac{2\pi}{n} + \sin\left(\lfloor \frac{n}{2} \rfloor \frac{\pi}{n}\right)$ for the first case and an upper bound of $3 + \frac{2\pi}{n} + \sin\frac{\pi}{n}$ for the second case, with lower bounds of $2 + \frac{\pi}{n}$ and $3 + \frac{\pi}{n}$ respectively.

ACKNOWLEDGEMENTS

To all the lonely robots out there, Konstantinos will guide you.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

A large component of applied mathematics aims to obtain solutions to practical problems by using optimization models. One such area of mathematics is Search Theory. In search problems active members, often called mobile agents or robots, explore a domain seeking out members with unknown locations. Ideally they want to find the optimal time it takes to locate immobile members, also called targets. There are a multitude of variations in search problems from types of search, to domain types, to mobility of players, to number of players and even in the quantities being optimized. The combinations are extensive. The domain of search problems can be linear, circular or a graph and there exists popular search problems on all domains. For example, in the famous Cow Path Problem [5] a cow is standing on an infinite straight line and would like to find a grazing field that she can only see when standing directly on it. The cow does not know which direction the field lies nor how far she is from the field. The algorithm explored in the paper aims to minimize the time spent in looking for the field no matter which direction the field lies. In this problem the mobile agent is the cow and the immobile agent is the field.

1.1 Overview of Research

We will commence by giving a brief background and history of the field of Search Theory. We follow with a description of our problem and lay out our main contributions. We then explore real life motivations and recent related work in the field.

In Chapter 2, we begin by introducing elementary bounds for the particular problem of two robots and both robots are required to move any given object. We tighten these bounds in Section 2.3 and 3.2, while spending some brief time in Section 3.1 exploring the difficulty involved with tightening the bounds.

In Chapter 4, we go beyond the particular problem and explore bounds for two general problems. The first of the problems is when we have n robots and each item can be carried by one robot. The second general problem is when there are n robots and exactly two are required to move any given object. For each of the problems we discover and prove both upper and lower bounds.

1.2 Search Theory

In the famous Cow Path problem, the mobile agent is an independent searcher. However search can also be performed as a group activity in which there are two or more searchers. These types of problems are called Group Search and typically they terminate when any cooperating member locates a target. One variation on Group Search is Evacuation problems. Evacuation problems terminate when all active members locate a target called an exit [16]. Another variation named Gathering problems, requires mobile searchers to collect a set of targets and terminates when all items are at the same location. While Gathering problems can fall under group search, this is not a requirement. It is possible that there is only one mobile agent searching for one or more targets to collect.

However which way search problems differ, the main objective is locating a target in a well-defined domain. Linear search is defined on an infinite line where targets are placed at a distance from the point of origin of the searcher [3]. Circular search is on a disk of specified size and objects are usually placed on the disk's perimeter [4]. Search can also occur in graphs where movement can be one-directional and location is defined by nodal points [4]. Decisions regarding direction for mobile agents is not solely restricted to graph search problems. All search is concerned with algorithmic design outlining trajectories of mobile agents. Trajectories can be defined deterministically or randomly. Randomness involves probabilistic methods for determining the performance of the algorithm. Algorithmic performance is defined as either the worst case or the average case cost of finding the hidden targets [10].

Algorithms determine the trajectories of the mobile agents and in problems where mobile agents are specifically robots, there is also the question of faulty robots [14]. Faultiness in Search Theory can be caused by a robot's inability to properly identify or locate a target or perhaps a robot is not honest and does not truthfully communicate their findings [11]. In Group Search, communication is key and can be defined in a couple of ways. Agents can either communicate wirelessly, at any distance and advise one another of findings in real time or can communicate non-wirelessly, which is also called the face-to-face to model [16]. In the

face-to-face model robots cannot communicate until they are both at the same point in the domain. Combinations of inputs in Search Theory are endless and will continue to grow as the field advances.

1.3 Model and Main Results

Here we introduce a particular search problem of the gathering kind denoted as $\mathbf{SRCH}_{n,k}$ in which we have n mobile agents searching for two immobile items hidden on a unit circle. Henceforth robots, mobile agents and searchers will be used interchangeably. Similarly, immobile members, items, objects and treasures will be synonymous. The movement of immobile members is restricted by the number of robots required to carry them. The number of robots needed to carry immobile items is k (where $k \leq n$). Robots search for treasures and termination occurs once at least k robots and all treasures are in the same location. Robots' visibility is minimal and they are unable to see the treasures unless they are standing directly in front of the object. In this problem, robots can communicate wirelessly and instantly with one another to announce the discovery of treasures and thus are not required to search together. Robots move at the same speed of 1. Our solutions are provided with a deterministic approach and our goal is to minimize the worst case time of locating the objects and bringing them together.

Worst-case performance analysis of an algorithm has proven to solicit complex approaches in search problems [3, 4]. In the chapters that follow we attempt to derive the optimal worst-case time it takes to find two objects and bring them together in $\mathbf{SRCH}_{n,k}$. As will be seen in Chapters 2,3 and 4, this is difficult to achieve and instead we aim to enclose this optimal value by providing upper and lower bounds. Lower bounds indicate that no matter what algorithm is presented, it will not be able to beat that bound for the worst case. Whereas for upper bounds we are interested in determining a tight upper bound for a particular algorithm. This is useful as it provides insight as to how poorly the algorithm can run in the worst case. Furthermore, if the lower and upper bounds match, then we have determined an algorithm that provides the optimal worst-case completion time.

It is also important to note that not all bounds are useful bounds. For example, in $\mathbf{SRCH}_{n,k}$ since the robots start and the center of a unit disk, move with speed 1, and must move to the perimeter to begin searching, then an obvious lower bound for any algorithm would be 1. Clearly, this is not useful as it doesn't take into account any search on the

perimeter and is too obvious to provide any insight. So we can improve on this bound. We present our main results in Table 1 for our problem $\mathbf{SRCH}_{n,k}$. Our results provide some bounds for searching with a small number of robots as well as an asymptotically large number of number of robots.

Search on a Disk with two Treasures			
n	k	Upper Bound	Lower Bound
1	1	$1+2\pi$	$1+2\pi$
2	1	$1+\pi$	$1+\pi$
$n \geq 3$	1	$1+2\frac{\pi}{n}+\sin\left(\lfloor\frac{n}{2}\rfloor\frac{\pi}{n}\right)$	$2+\frac{\pi}{n}$
2	2	6.7518	$3+\frac{\pi}{2}$
$n \geq 3$	2	$3+\frac{2\pi}{n}+\sin\frac{\pi}{n}$	$3+\frac{\pi}{n}$

Table 1.1: Main Results for $\mathbf{SRCH}_{n,k}$

1.4 Related Work

While search problems assume a modern flair they were first introduced and discussed in the sixties by Bellman [7] and independently proposed by Beck [6]. Since that time the field of Search Theory has continued to grow its branches by the countless variations available, some of which were discussed above. There are basic search problems where termination is determined once all treasures are found [16]. There are evacuation problems where termination occurs once all mobile agents find the exit located at an unknown location in the domain.

A more particular evacuation problem is priority evacuation first considered in [12, 13]. The problem introduced $n+1$ robots searching on the perimeter of a disk for an exit at an unknown location. The goal was for any of the $n+1$ robots to locate the exit so that the distinguished robot given priority (called the queen) could exit. The queen was able to participate in the search but could also not participate. The role of a distinguished agent was new in search problems. The relevance of the paper is modeled in real life situations where evacuation is critical and in which certain members of society are given priority over others due to class, hierarchical position or even demographic differences. In many research problems both the wireless and face-to-face models are reviewed for a small number as well as an asymptotically large number of robots [13, 17, 22].

Traditionally in evacuation problems all agents must exit for the algorithm to terminate [10]. This research will present work with some parallels in terms of domain and number of mobile agents. There are many similarities between search problems and yet small variances provide large discrepancies in achieving bounds. In reviewing an evacuation problem with two robots on a disk [9], a worst case upper bound of 5.625 is achieved. Changing the problem to a gathering model, increases upper bounds significantly as was seen with our results. Similarly, some variances decrease upper bounds, such as in [14] where an upper bound of 4.779 is achieved for evacuation on a disk for two non-faulty robots and is optimal.

There are several less typical domains in evacuation problems such as polygons that have been researched [19]. Robots in non-typical domains such as squares and triangles with sides of equal length of 1 were explored in [15]. In some search problems robots may be in an

unfamiliar domain that requires mapping instead of target hunting and the environment may be modeled by a strongly connected graph [1, 18]. To map an unknown area, robots must visit and keep track of all nodes and edges [2]. The well known game of Cops and Robbers studied on graphs has multifarious research beginning with Quilliot [24] and independently by Nowakoski and Winkler [21] who both explored a cop number of 1. The cop number is the number of cops required to catch robbers in the domain and is well explored in the book The Game of Cops and Robbers on a Graph [8]. Research in search theory is extensive in the field of graph theory and often involves probabilistic measures for locating targets. Meyniel's famous conjecture announcing an upper bound for the cop number has been studied extensively and was recently confirmed for the binomial random graph by Pralat and Wormald [23].

1.5 Motivation

Search Theory is applicable to an ample amount of real life problems. It is evident in search-and-rescue operations and has been explored in [16, 17] when robots receive distress signals. Robots may only be given limited information about the exact location of the signal. All robots are located at a base camp which is synonymous with points of origin in the mathematical models and the signals are originating from a distant specific point. Multiple points of distress can imply that they lie on concentric circles. When the points are equidistant from the origin then the robots will only need to search the area of a disk which parallels our problem [16, 17]. Search-and-rescue operations occur in a numerous concerning scenarios such as in rough terrains (mountains), when planes go missing in oceans and even in combat.

More modern applications conjugate to Machine Learning and AI advancements. Deep learning algorithms can have circular sub-functions and are require to locate target solutions and time complexity. We are motivated by both benevolent and concomitant future technological applications and aim to provide bounds that can determine performance if our algorithms are applied.

Chapter 2

Problem Definition and Preliminaries

Problem $\mathbf{SRCH}_{n,k}$ is a gathering problem in which n robots are searching for two objects located somewhere on the perimeter of a unit disk and where each object must be moved by k robots (where $k \leq n$). All robots begin the search from the center of the circle and are able to move anywhere in the domain with an identical speed of 1. Robots are able to communicate from a distance to notify one another when an object, also known as an item or treasure, is found. Robot trajectories are described by families of algorithms and the performance of these algorithms are analyzed so as to determine the completion time of the search. The completion time is the time it takes until all objects are in the same location. Feasible solutions are defined as robot trajectories that bring the two treasures together. The goal is to design an algorithm describing the trajectories of the robots so as to minimize the worst case completion time over all instances.

We will review several variations of this problem, however we focus more particularly on a special case where we have two objects, and both $n, k = 2$, ($\mathbf{SRCH}_{2,2}$). We then generalize these results and analyze the larger case where we have $n \geq 3$ robots and $k \in \{1, 2\}$, to determine optimal trajectories. In Mathematics and Theoretical Computer Science performance analysis of an algorithm is typically measured by the worst case and the average case [4]. We will present the worst case analysis of all variations of the problem. Worst case analysis regarding completion time is of concern since we want to know how much time the robots need to

locate and gather the objects. Through algorithmic design and the subsequent performance analysis we aim to enclose the optimal solutions by providing strong upper bounds and lower bounds for both the general case and the special case. Upper bounds can be obtained by the analysis of any particular algorithm for a specified problem. The optimal possible solution will always be less than or equal to that upper bound. Whereas lower bounds provide a benchmark for all families of algorithms of a problem. Optimal solutions are identified when the upper and lower bounds match.

2.1 Notation

We begin by exploring less complex algorithms that follow more obvious trajectories for the robots, such as the Cooperative Algorithm and the Splitting Algorithm introduced in section 2.4. These provide evidence and need for more complex behavior of robot trajectories to enhance algorithmic performance. For simplicity, our domain, the unit circle will be centered at $(0, 0)$ in the Cartesian plane. The locations of hidden objects are identified by any $u, v \in [0, 2\pi)$. Object u, is then at point $(\cos u, \sin u) \in \mathbb{R}^2$ denoted by $cycle(u)$. We call the set of all possible placements of objects collectively as the instances $I(u, v)$ and provide detailed analytic support for worst case analysis of the instance.

Consider an algorithm A. The performance of A for input $I(u, v)$ is the time it takes to bring both objects together, which we call the completion time and which is denoted by $C_A(I)$. We design A so as to minimize $\sup_I C_A(I)$. Feasible solutions are robot trajectories that bring both objects together eventually. In our analysis for search when $n = k = 2$ we will specify and denote the shortest arc distance from the $cycle(0)$ to the first object found by x_1 and the shortest arc distance from the $cycle(0)$ to the second object found by x_2. Additionally, we will always denote the robot that finds the first item by R_1 and the other robot will then always be R_2. Robots move with a speed of 1, so any distance they travel will also equal the time it took to get there and the time will be denoted as such.

2.2 Linepoints and Fetching

Prior to introducing algorithms it is necessary to define two functions; **linepoints** and **fetching** which are denoted by $\mathcal{L}(A, B, t)$ and $\mathcal{F}(F, N, S)$ respectively. Both functions are used to describe robot trajectories when searching for objects and once all objects are found. **Linepoints** describes a robot's location when inside the circle and **fetching** is the time it takes robots to bring both objects together once it is announced that all objects are found. Objects are only located on the perimeter of the circle but robots are free to move throughout the domain. So we require a description of robot trajectories not just on the perimeter but within the domain. We define this path describing the robots' location by a function [10] called **linepoints**.

Definition 2.2.1. Consider two distinct points $A = (a_x, a_y)$ and $B = (b_x, b_y) \in \mathbb{R}^2$ on the perimeter of the disk. The location of a robot with speed 1 moving along the vector from point B to point A is given by the following parametric equation, **linepoints**,

$$\mathcal{L}(A, B, t) = \left(\frac{a_x - b_x}{||B - A||} t + b_x, \frac{a_y - b_y}{||B - A||} t + b_y \right). \tag{2.2.1}$$

In Equation (2.2.1) t (where $t \in \mathbb{R}$), is amount of time that a robot has spent moving along the line from point B to A. **Linepoints** is used as an input in **fetching** when one robot is not located on the perimeter but on a line segment connecting some points A and B. In **fetching**, $\mathcal{F}(F, N, S)$, F indicates the location of the second item found and the robot that found it, N indicates the location of the other robot and S indicates the location of the first object found. It is possible that the three points are not distinct. Additionally, fetching is most useful when the items have been located and only one robot has found the second item. Fetching is a function interested in determining the minimum cost of multiple possibilities. Once all treasures are found, robots are faced with decisions regarding the path to take to bring the objects together. This is where the communication model exhibits its usefulness. The ability of the robots to communicate wirelessly allows them to choose the shortest route by advising one another of their respective locations, thus minimizing the

worst case completion time. So when all objects are located it is in the best interest of the robots to determine the shortest path to bringing them together.

Theorem 2.2.2. Consider an instance I. Given that F, N and S are known, there is an algorithm that we call **fetching** that requires $\mathcal{F}(F, N, S)$ extra time to terminate from the moment the second item is found at point F. Where,

$$\mathcal{F}(F, N, S) = ||F - S|| + \min \begin{cases} ||F - N|| \\ \max \left\{ \ ||F - S||, ||N - S|| \ \right\} \end{cases}. \tag{2.2.2}$$

Proof. Consider $\mathbf{SRCH}_{2,2}$ and suppose that both items have been found and the robots were not searching together. The finder of the second object is located at point F, and the other robot is located at point N somewhere in the domain. While the first object is located at point S. Let us recall, that all three points need not be distinct. Fetching is the time it takes the two robots to bring together the two objects given the three points of interest. So either item one needs to be carried to item two or item two needs to be carried to item one. Consider the first option. If item one is carried to item two then both robots need to travel back to the first item. It would take time $\max\{||F - S||, ||N - S||\}$ for the last robot to arrive at the first object. It would then take an additional time of $||F - S||$ to carry the first item to the second item. Thus the total time to carry the first item to the second item is $\max\{||F - S||, ||N - S||\} + ||F - S||$.

Now consider the second option; carrying the second item to the first item. Recall that the finder of the second item is with the second item, while the non-finder was located anywhere in the domain. So it takes a time of $||F - N||$ for the non-finder to arrive at the second item and an additional time of $||F - S||$ for both to carry the second item back to the first item. Therefore, the total time required to carry the second item to the first is $||F - N|| + ||F - S||$. Hence the time is take to bring both items together, i.e. to fetch is,

$$\mathcal{F}(F, N, S) = \min \begin{cases} ||F - N|| + ||F - S|| \\ \max \left\{ \ ||F - S||, ||N - S|| \ \right\} + ||F - S|| \end{cases} \tag{2.2.3}$$

Which is clearly equivalent to equation (2.2.2). □

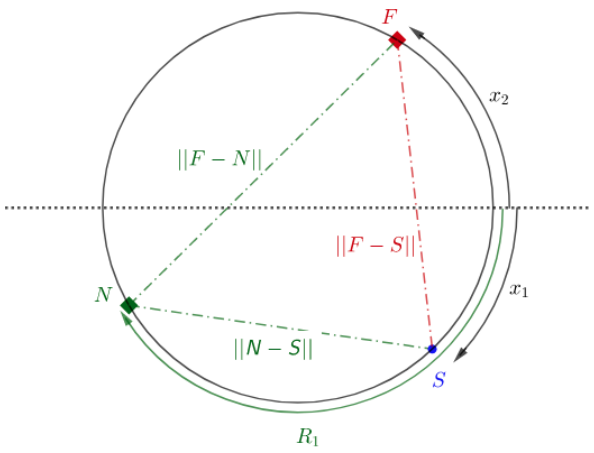

Figure 2.1: An Example of Fetching with two Robots on the Perimeter

Example 2.2.3. In Figure 2.1 we show the three points of interests. We note that the second item is located counterclockwise from $cycle(0)$ at an arc distance of x_2 and the first item is located clockwise from $cycle(0)$ at an arc distance of x_1. Here $F = cycle(x_2)$, $N = cycle(-R_1)$, $S = cycle(-x_1)$ and

$$\begin{aligned} ||F - N|| &= 2\sin\frac{|x_2 + R_1|}{2} \\ ||F - S|| &= 2\sin\frac{|x_2 + x_1|}{2} \\ ||N - S|| &= 2\sin\frac{|x_1 - R_1|}{2}. \end{aligned}$$

Since we are looking for the minimum of two paths for the robots, **fetching** becomes

$$\mathcal{F}(F, N, S) = \min\left\{ \begin{array}{l} 2\sin\frac{|x_2+R_1|}{2} + 2\sin\frac{|x_2+x_1|}{2} \\ \max\left\{ 2\sin\frac{|x_2+x_1|}{2}, 2\sin\frac{|x_1-x_2|}{2} \right\} + 2\sin\frac{|x_2+x_1|}{2} \end{array} \right. \tag{2.2.4}$$

If the location of N, the robot who did not find the second object, is not on the perimeter then $N = \mathcal{L}(A, B, t)$ for some $A, B \in \mathbb{R}^2$ and for some $t \in \mathbb{R}$. Furthermore, N need not be distinct from the location of the first or the second object. In fact, we often see that $F = N$ in the algorithms to follow.

2.3 Elementary Bounds

We begin by reviewing some elementary bounds to establish a starting point. We will first consider the special case when $n = k = 1$. In the problem $\mathbf{SRCH}_{1,1}$ we have two immobile objects and only one searcher.

Theorem 2.3.1. The optimal worst case cost of $\mathbf{SRCH}_{1,1}$ is $1 + 2\pi$.

It is clear to see that any trajectory of the searcher that avoids an uninterrupted full tour of the perimeter would increase the completion time. We justify the above theorem by the following. We will start with the upper bound by presenting the following algorithm:

Algorithm 1 Tour Algorithm

1: The robot moves to $\mathbf{cycle}(0)$
2: They travel counterclockwise in search of both objects
3: Once any object is located they carry it with them until the second object is found

The completion time for the robot will be the addition of time 1 to get to the perimeter and the time it takes to find the last object. We denote the arc distance from $cycle(0)$ of the first and second object found by x and y respectively. Thus the worst case completion time is equivalent to $1 + \max\{x, y\}$, where $x, y \in [0, 2\pi]$. Suppose x is the larger value. Then the maximum completion time is $\leq 1 + 2\pi$. Now suppose y is the larger value then the maximum completion time is also $\leq 1 + 2\pi$. Thus the total worst case cost of one robot searching for both items is bounded above by $1 + 2\pi$.

We will now examine the lower bound. Let $\epsilon \in \mathbb{R}$, where $\epsilon > 0$. Fix an arbitrary algorithm A and let A run for $1 + 2\pi - \epsilon$. Then no matter what algorithm is provided the robot has searched at most $2\pi - \epsilon$ of the perimeter. Thus the worst case placement of the objects on the perimeter will always be within the interval of arc distance $(0, \epsilon)$ that the robot has not searched.

Now we consider the problem of two robots and only one robot is needed to carry an item.

Theorem 2.3.2. The optimal worst case cost of $\mathbf{SRCH}_{2,1}$ is $1 + \pi$.

We justify the above theorem by the following.

Algorithm 2 Split Search

1: Both robots moves to **cycle**(0)
2: R_2 travels counterclockwise and R_1 travels clockwise in search of both objects
3: Once any object is located they carry it with them until the second object is found

The completion time for the robot will be the addition of time 1 to get to the perimeter and the time it takes to find the last object. We denote the shortest arc distance from $cycle(0)$ of the first and second object found by x and y respectively. Thus the worst case completion time is equivalent to $1 + \max\{x, y\}$, where $x, y \in [0, \pi]$. Suppose x is the larger value. Then the maximum completion time is $\leq 1 + \pi$. Now suppose y is the larger value then the maximum completion time is also $\leq 1 + \pi$. Thus the total worst case cost is bounded above by $1 + \pi$ no matter how soon in the search the first item is found.

We will now examine the lower bounds. Let $\epsilon \in \mathbb{R}$, where $\epsilon > 0$. Fix an arbitrary algorithm A and let A run for $1 + \pi - \epsilon$. Then no matter what algorithm is provided the robots have searched at most $2\pi - 2\epsilon$ of the perimeter. Thus the worst case placement of the objects on the perimeter will always be within the interval of arc distance $(0, 2\epsilon)$ that the two robots have not searched.

We are now ready to develop some preliminary bounds for the problem of two robots and both robots are required to carry any given object.

2.4 The Problem of Two Robots and Preliminary Bounds

$\mathbf{SRCH}_{2,2}$ is a particular instance of $\mathbf{SRCH}_{n,k}$ where we have two treasures, two robots and both robots are required to move a treasure. An obvious algorithm is for both robots to travel together along the perimeter and carry the first treasure to the second treasure.

Algorithm 3 Cooperative Algorithm

1: Both robots move to **cycle**(0)
2: Travel counterclockwise together
3: Once they locate the first object they carry it with them until the second object is found

The objective is to determine the worst case completion time for the Cooperative Algorithm given all inputs, $I(u, v)$. It takes time 1 for the robots to travel to the perimeter and then they travel together counterclockwise in search of both objects carrying the first one with them when they find it. Thus the total time to locate both objects would be 1 plus the object located the furthest away from their starting point anywhere after $cycle(0)$ and before $cycle(2\pi)$. So we must minimize the following,

$$\begin{aligned} \sup_{u,v} \quad & 1 + \max\{u, v\} \\ \text{where} \quad & 0 \le u, v < 2\pi \end{aligned} \tag{2.4.1}$$

On the unit circle, $cycle(0) = cycle(2\pi)$, so both objects can be located either at $cycle(0)$ or just before $cycle(2\pi)$. Suppose that the first and second object are found at a counterclockwise arc distance of x_1 and x_2 respectively from $cycle(0)$. By adopting an optimization perspective of the algorithm the above equation becomes the following optimization problem:

$$\begin{aligned} \max_{x_1,x_2} \quad & 1 + x_2 \\ \text{s.t.} \quad & 0 \le x_1 \le x_2 \le 2\pi \end{aligned} \tag{2.4.2}$$

Clearly, the solution to the above maximization problem is $x_2 = 2\pi$ and so $C_A(I) = 1 + 2\pi$ which is approximately 7.2832. We ask ourselves now, is there indeed an instance that would produce this cost? There is no instance where the second object found is located exactly

at $cycle(2\pi)$, if so both robots would locate it as soon as they traveled from the center to $cycle(0)$. Which means that the second object found is located at $cycle(2\pi) - \epsilon$, for some arbitrarily small $\epsilon > 0$. However, no matter what value of ϵ is provided there is always a worse instance where the second object is found. Thus $\sup_{u,v} 1 + \max\{u, v\} = \max C_A(I) = 1 + 2\pi$.

The cost of the Cooperative Algorithm is fairly high due to the lack of sophistication of the algorithm. We can now introduce the Splitting Algorithm. In the Splitting Algorithm a minor change in trajectories for the robots adds some complexity in the performance analysis and employs **fetching**. Without loss of generality, the robot who locates the first treasure will be denoted as R_1 and the other robot will be denoted by R_2.

Algorithm 4 Splitting Algorithm

1: R_1 and R_2 move to **cycle**(0)
2: R_1 travels clockwise and R_2 travels counterclockwise searching the disk simultaneously
3: As soon as the second object is located, R_1 and R_2 initiate **fetching**

We claim the following to be proved shortly.

Theorem 2.4.1. The worst case performance of the Splitting Algorithm is $3 + \frac{2\pi}{3} + \sqrt{3}$.

2.4.1 Performance Analysis for the Splitting Algorithm

We will now present a systematic approach for determining the worst case performance of the Splitting Algorithm. In doing so we will be able to contrive a better upper bound than the one determined by the Cooperative Algorithm. Since the robots are searching in parallel and the circle is symmetric, there are two cases to review. In **Case 1** both items are found by the same robot and in **Case 2** each robot finds one item. We will assume as usual, that R_1 finds the first object.

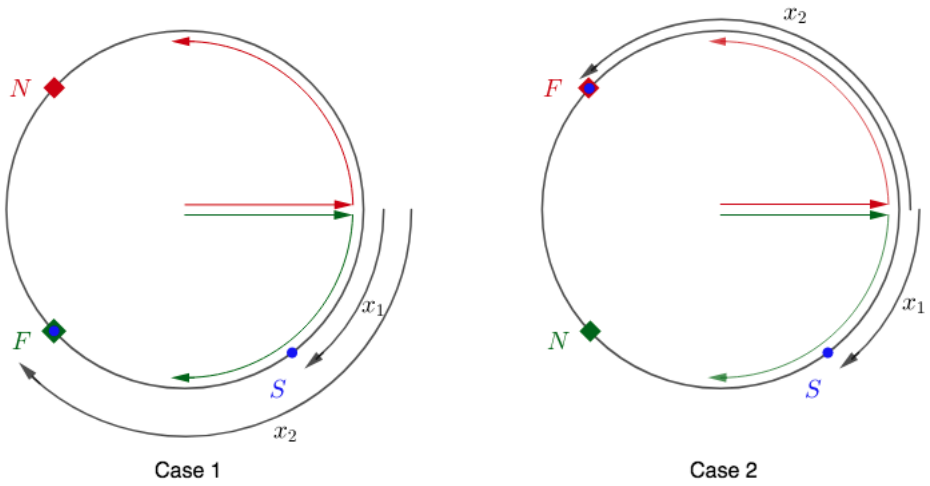

Figure 2.2: The two Cases of the Splitting Algorithm

The performance of the Splitting Algorithm in both cases will be the time it takes for both agents to move from the center of the circle to the perimeter, the additional time required to find the second item to be found and the additional time it takes to complete fetching. Thus the general optimization problem for both cases will be,

$$\begin{aligned} \max_{x_1,x_2} \quad & 1 + x_2 + \mathcal{F}(F,N,S) \\ \text{s.t.} \quad & 0 \le x_1 \le x_2 \le \pi \end{aligned} \tag{2.4.3}$$

The difference in the cases occurs in the values of F, N and S which can be seen in Figure 2.2. For Case 1 $F = cycle(-x_2)$, $N = cycle(x_2)$, $S = cycle(-x_1)$ and

$$\begin{aligned} ||F - N|| &= 2\sin\frac{|-x_2 - x_2|}{2} = 2\sin x_2 \\ ||F - S|| &= 2\sin\frac{|-x_2 - (-x_1)|}{2} = 2\sin\frac{x_2 - x_1}{2} \\ ||N - S|| &= 2\sin\frac{|x_2 - (-x_1)|}{2} = 2\sin\frac{x_2 + x_1}{2} \end{aligned}$$

For Case 2 $F = cycle(x_2)$, $N = cycle(-x_2)$, $S = cycle(-x_1)$ and

$$\|F-N\| = 2\sin\frac{|x_2-(-x_2)|}{2} = 2\sin x_2$$
$$\|F-S\| = 2\sin\frac{|x_2-(-x_1)|}{2} = 2\sin\frac{x_2+x_1}{2}$$
$$\|N-S\| = 2\sin\frac{|-x_2-(-x_1)|}{2} = 2\sin\frac{x_2-x_1}{2}$$

Now we can construct our particular optimization problems for each case respectively. For Case 1,

$$\begin{aligned} \max_{x_1,x_2} \quad & 1+x_2+2\sin\tfrac{x_2-x_1}{2}+2\min\begin{cases}\sin x_2,\\ \max\{\sin\frac{x_2-x_1}{2},\sin\frac{x_2+x_1}{2}\}\end{cases} \\ \text{s.t.} \quad & 0\le x_1\le x_2\le\pi \end{aligned} \tag{2.4.4}$$

and for Case 2,

$$\begin{aligned} \max_{x_1,x_2} \quad & 1+x_2+2\sin\tfrac{x_1+x_2}{2}+2\min\begin{cases}\sin x_2,\\ \max\{\sin\frac{x_2-x_1}{2},\sin\frac{x_2+x_1}{2}\}\end{cases} \\ \text{s.t.} \quad & 0\le x_1\le x_2\le\pi \end{aligned} \tag{2.4.5}$$

Looking at Equations (2.4.4) and (2.4.5), we can observe that we are optimizing functions of minimum and maximum values. Hence, each case yields four distinct non-linear programs with different maximization functions and distinct constraints that require solving. The maximum value of the eight non-linear programs will produce of a cost of $3+\frac{2\pi}{3}+\sqrt{3}$. Since there are two cases of the Splitting Algorithm with four distinct non-linear programs, we will solve each optimization problem and determine the maximum cost performance. Case 1 becomes NLPs 1 to 4 and Case 2 becomes NLP's 5 to 8.

NLP 1

$$\begin{aligned} \max_{x_1,x_2} \quad & 1+x_2+2\sin\tfrac{x_2-x_1}{2}+2\sin x_2 \\ \text{s.t.} \quad & 0\le x_1\le x_2\le\pi \\ & \sin\tfrac{x_2+x_1}{2}\le\sin\tfrac{x_2-x_1}{2} \end{aligned}$$

NLP 2

$$\begin{aligned} \max_{x_1,x_2} \quad & 1+x_2+2\sin\tfrac{x_2-x_1}{2}+2\sin x_2 \\ \text{s.t.} \quad & 0\le x_1\le x_2\le\pi \\ & \sin\tfrac{x_2-x_1}{2}\le\sin\tfrac{x_2+x_1}{2} \end{aligned}$$

NLP 3

$$\begin{aligned} \max_{x_1,x_2} \quad & 1 + x_2 + 2\sin\tfrac{x_2-x_1}{2} + 2\sin\tfrac{x_2+x_1}{2} \\ \text{s.t.} \quad & 0 \le x_1 \le x_2 \le \pi \\ & \sin\tfrac{x_2-x_1}{2} \le \sin\tfrac{x_2+x_1}{2} \le \sin x_2 \end{aligned}$$

NLP 4

$$\begin{aligned} \max_{x_1,x_2} \quad & 1 + x_2 + 4\sin\tfrac{x_2-x_1}{2} \\ \text{s.t.} \quad & 0 \le x_1 \le x_2 \le \pi \\ & \sin\tfrac{x_2+x_1}{2} \le \sin\tfrac{x_2-x_1}{2} \le \sin x_2 \end{aligned}$$

NLP 5

$$\begin{aligned} \max_{x_1,x_2} \quad & 1 + x_2 + 2\sin\tfrac{x_1+x_2}{2} + 2\sin x_2 \\ \text{s.t.} \quad & 0 \le x_1 \le x_2 \le \pi \\ & \sin\tfrac{x_2-x_1}{2} \le \sin\tfrac{x_2+x_1}{2} \end{aligned}$$

NLP 6

$$\begin{aligned} \max_{x_1,x_2} \quad & 1 + x_2 + 2\sin\tfrac{x_1+x_2}{2} + 2\sin x_2 \\ \text{s.t.} \quad & 0 \le x_1 \le x_2 \le \pi \\ & \sin\tfrac{x_2+x_1}{2} \le \sin\tfrac{x_2-x_1}{2} \end{aligned}$$

NLP 7

$$\begin{aligned} \max_{x_1,x_2} \quad & 1 + x_2 + 2\sin\tfrac{x_1+x_2}{2} + 2\sin\tfrac{x_2-x_1}{2} \\ \text{s.t.} \quad & 0 \le x_1 \le x_2 \le \pi \\ & \sin\tfrac{x_2+x_1}{2} \le \sin\tfrac{x_2-x_1}{2} \le \sin x_2 \end{aligned}$$

NLP 8

$$\begin{aligned} \max_{x_1,x_2} \quad & 1 + x_2 + 2\sin\tfrac{x_1+x_2}{2} + 2\sin\tfrac{x_2+x_1}{2} \\ \text{s.t.} \quad & 0 \le x_1 \le x_2 \le \pi \\ & \sin\tfrac{x_2-x_1}{2} \le \sin\tfrac{x_2+x_1}{2} \le \sin x_2 \end{aligned}$$

We are now ready to prove Theorem 2.4.1.

Proof. We claim that an optimal solution to one of the optimization problems above will provide a completion time of $3 + \frac{2\pi}{3} + \sqrt{3}$. We consider Case 2 first and we will commence with **NLP 7**. Let

$$f_7(x_1, x_2) = 1 + x_2 + 2\sin\frac{x_1 + x_2}{2} + 2\sin\frac{x_2 - x_1}{2},$$

and let $u = \frac{x_2+x_1}{2}$ and $w = \frac{x_2-x_1}{2}$. Since $0 \le x_1 \le x_2 \le \pi$ then $w \le u$ and $w \le \frac{\pi}{2}$. So $\pi - w \le u$ and,

$$\pi - \left[\frac{x_2 - x_1}{2}\right] \le \left[\frac{x_2 + x_1}{2}\right]$$

$$\Rightarrow 2\pi - (x_2 - x_1) \le x_2 + x_1$$

$$\Rightarrow \pi \le x_2$$

But $x_2 \leq \pi$. Thus $x_2 = \pi$.

Using the second constraint and the fact that $x_2 = \pi$, we observe that

$$\sin \frac{x_2 + x_1}{2} \leq \sin x_2$$
$$\Rightarrow \sin \frac{\pi + x_1}{2} \leq 0$$

However $\sin \frac{x_2+x_1}{2}$ is a positive value since both x_1 and x_2 are values between zero and π inclusive. So $\sin \frac{\pi+x_1}{2} = 0$. Which implies that $\frac{\pi+x_1}{2} = 0$ or π.

If $\frac{\pi+x_1}{2} = 0$ then $x_1 = -\pi$.

If $\frac{\pi+x_1}{2} = \pi$ then $x_1 = \pi$.

But $0 \leq x_1$, so $x_1 = x_2 = \pi$ and $\max f_3(x_i) = 1 + \pi$ which is approximately 4.1416.

Consider now **NLP 8** and let,

$$f_8(x_1, x_2) = 1 + x_2 + 4 \sin \frac{x_1 + x_2}{2}$$

Observing the second constraint, $\sin \frac{x_1+x_2}{2} \leq \sin x_2$, we can relax **NLP 8** to the following optimization problem:

$$\begin{aligned} \max_{x_1,x_2} \quad & 1 + x_2 + 4 \sin x_2 \\ \text{s.t.} \quad & 0 \leq x_2 \leq \pi \end{aligned}$$

Where we let $g_8(x_1, x_2) = 1 + x_2 + 4 \sin x_2$ and we note that $\max f_8(x_1, x_2) \leq \max g_8(x_2)$ due to feasible region of **NLP 8**.

Taking the first derivative of g_8 we obtain; $g_8'(x_2) = 1 + 4 \cos x_2$. Setting the derivative to 0 we obtain the following solution,

$$x_0 = \arccos\left(-\frac{1}{4}\right).$$

Noting that this solution satisfies the constraints, we know that it is a local optimizer. We

must now check if it is a global optimizer by taking the second derivative.

$$g_8''(x_2) = \left(-4\sin x_2\right)$$

And

$$g_8''(x_0) = \left(-4\sin\arccos\left(-\tfrac{1}{4}\right)\right).$$

We obtain the following associated eigenvalue for the above solution;

$$\lambda_0 = -4\sin\arccos(-\frac{1}{4}),$$

showing that $g_8''(x_0) \leq 0$. Thus x_0 is a global maximizer and the cost of **NLP 8** is no more than 6.6965. Reviewing **NLP 6** we observe the constraints are similar to **NLP 7**. Letting $f_6(x_1, x_2) = 1 + x_2 + 2\sin\frac{x_2+x_1}{2} + 2\sin x_2$, $u = \frac{x_2+x_1}{2}$ and $w = \frac{x_2-x_1}{2}$ again we obtain that $x_2 = \pi$. If $x_2 = \pi$ then we can reduce **NLP 6** to the following optimization problem,

$$\begin{aligned} \max_{x_1,x_2} \quad & 1 + \pi + 2\cos\left(\tfrac{x_1}{2}\right) \\ \text{s.t.} \quad & 0 \leq x_1 \leq \pi \end{aligned}$$

We let $g_6(x_1) = 1+\pi+2\cos\left(\frac{x_1}{2}\right)$ and observe that $\cos\left(\frac{x_1}{2}\right) \leq 1$. Solving for x_1, we get $x_1 \leq 0$. Thus $x_1 = 0$, which satisfies the constraints. So $x_1 = 0$ and $x_2 = \pi$ and $\max f_2(x_i) = 1 + \pi + 2$ which is approximately 6.1416.

For **NLP 5** let,

$$f_5(x_1, x_2) = 1 + x_2 + 2\sin\frac{x_1 + x_2}{2} + 2\sin x_2$$

Then,

$$\nabla f_5(x_1, x_2) = \begin{pmatrix} \cos\frac{x_2+x_1}{2} \\ 1 + \cos\frac{x_2+x_1}{2} + 2\cos x_2 \end{pmatrix}$$

Setting the gradient to 0 we obtain the following system of equations,

$$\begin{pmatrix} \cos\frac{x_2+x_1}{2} \\ 1+\cos\frac{x_2+x_1}{2}+2\cos x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Solving the above system of equations yields the following feasible solution set

$$x_0 = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \frac{\pi}{3} \\ \frac{2\pi}{3} \end{pmatrix}.$$

The solution set satisfies both constraints. Hence, x_0 is a local optimizer. We must now only check if x_0 is a global maximizer.

$$\nabla^2 f_5(x_1, x_2) = \begin{pmatrix} -\frac{1}{2}\sin\frac{x_2+x_1}{2} & -\frac{1}{2}\sin\frac{x_2+x_1}{2} \\ -\frac{1}{2}\sin\frac{x_2+x_1}{2} & 1-\frac{1}{2}\sin\frac{x_2+x_1}{2}+2\sin x_2 \end{pmatrix}$$

Inputting our solution x_0 we obtain,

$$\nabla^2 f_5(x_0) = \begin{pmatrix} -\frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{2}-\frac{\sqrt{3}}{2} \end{pmatrix}.$$

Solving for the eigenvalues of the above Hessian we can see that,

$$\lambda_0 = \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} \frac{1}{4}(-2-\sqrt{3}-\sqrt{7}) \\ \frac{1}{4}(-2-\sqrt{3}+\sqrt{7}) \end{pmatrix}.$$

Thus, $\nabla^2 f_5(x_0) \preccurlyeq 0$ and so x_0 is a global maximizer. Hence,

$$\begin{aligned} f_5(x_1, x_2) &= 1+\frac{2\pi}{3}+2\sin\frac{\frac{2\pi}{3}+\frac{\pi}{3}}{2}+2\sin\frac{2\pi}{3} \\ &= 3+\frac{2\pi}{3}+\sqrt{3}. \end{aligned}$$

Case 1 follows a similar pattern and we present the solutions for both cases in the following table.

Solutions to Cases 1 and 2			
	Case 1		**Case 2**
NLP 1	$3+\pi \approx 6.1416$	**NLP 5**	$3+\frac{2\pi}{3}+\sqrt{3} \approx 6.8264$
NLP 2	≤ 6.8264	**NLP 6**	$3+\pi \approx 6.1416$
NLP 3	≤ 6.6965	**NLP 7**	$1+\pi \approx 4.1416$
NLP 4	≤ 6.6965	**NLP 8**	≤ 6.6965

Table 2.1: Solutions for the Splitting Algorithm

□

We can clearly see that **NLP 5** for Case 2 attains the desired worst case performance for the Splitting Algorithm. In Case 2 each robot locates one object, so the worst case placement of objects is either $I(\frac{2\pi}{3}, -\frac{\pi}{3})$ or $I(\frac{\pi}{3}, -\frac{2\pi}{3})$ both attaining the same worst case cost. The completion time of $3+\frac{2\pi}{3}+\sqrt{3} \approx 6.8264$ is now our benchmark for upper bounds that will be improved upon with subsequent algorithms. We are left now to explore a lower bound for $\mathbf{SRCH}_{2,2}$.

Theorem 2.4.2. There is no algorithm for $\mathbf{SRCH}_{2,2}$ with a performance better than $3+\frac{\pi}{2}$.

Proof. Let $\epsilon \in \mathbb{R}$, where $\epsilon > 0$. Fix an arbitrary algorithm A and let A run for $1+\frac{\pi}{2}-\epsilon$ time. It takes time 1 for both robots to move to the perimeter of the unit circle. After extra time $\frac{\pi}{2}-\epsilon$ the robots have searched at most $2(\frac{\pi}{2}-\epsilon) = \pi - 2\epsilon$ of the perimeter. Consider a diameter l_θ, that forms angle θ with the x-axis. We define $S \subseteq [0, 2\pi)$ as follows, $\theta \in S$ if and only if at least one of the endpoints of l_θ has been explored. If at least one endpoint of all antipodal points has been explored, then $[0, \pi] \subseteq S$. However, that means at least π of the perimeter has been searched, which is a contradiction as the robots have searched at most $\pi - 2\epsilon$ of the perimeter. Therefore at time $1+\frac{\pi}{2}-\epsilon$, there exists two antipodal points on the circle, both of which have not been explored. We place the objects there and observe that it takes at least time 2 to bring the objects together as both robots are required to carry one object. □

Chapter 3

Improving Bounds for two Robots

3.1 The Problem of Improving Bounds

The main contribution of our work is presented in this section and the next, where we explore algorithms with more sophisticated trajectories. As in the Splitting Algorithm, we will be employing a case by case analysis for the performance of the algorithms. We begin by introducing a new family of algorithms named $\alpha - RALLY$ whose performance varies as α changes. We establish $\alpha - RALLY$ to note the difficulty in improving upon the Splitting Algorithm and so that we can inaugurate the best algorithm for $\mathbf{SRCH}_{2,2}$ that our research allowed. The difficulty in improving upper bounds might not seem obvious. Various attempts not presented here were made. We use $\alpha - RALLY$ as an example as it serves as a prerequisite to the algorithm presented in the next section. We must define some new notation that will be used in the following analysis. Consider two robots traveling in opposite directions from $cycle(0)$. Once they both arrive at an arc distance of α one cuts across the circle and the other continues along the perimeter where they both meet at a point. We will define d_α as the length of the chord joining α and the point the robots meet.

Definition 3.1.1. The distance d_α is defined as the unique positive solution to the non-linear equation,

$$d_\alpha = 2\sin\frac{2\alpha + d_\alpha}{2}.$$

Since the robots both travel at the same speed and simultaneously, then the arc distance from α' to the point the robots meet is equivalent to d_α.

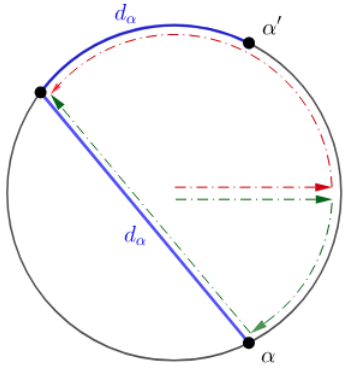

Figure 3.1: Depiction of the distance, d_α

We are now ready to describe the $\alpha - RALLY$ algorithm.

Algorithm 5 $\alpha - RALLY$

1: R_1 and R_2 move to **cycle**(0)
2: If at any time both items are located, R_1 and R_2 initiate $\mathcal{F}(F, N, S)$
3: R_1 travels clockwise and R_2 travels counterclockwise searching α
4: If only one item is found within time α of searching, the finder will travel a distance of d_α and meet the non-finder to continue searching together
5: If no items are found by time α then R_1 and R_2 continue to search independently until both items are found

We approach the analysis with a case by case study of the robot trajectories and locations of the treasures. There are four main cases to observe.

- **Case 1:** When both treasures are located within a time of α
- **Case 2:** When one treasure is located within a time of α and the other after time α and before d_α
- **Case 3:** When one treasure is located within a time of α and the other after time $d_\alpha + \alpha$
- **Case 4:** Both treasures located after time α

Case 1

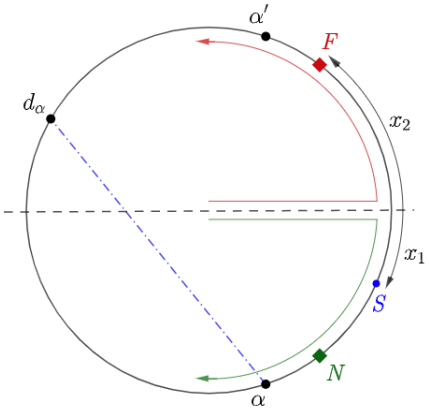

Figure 3.2: Case 1 of $\alpha - RALLY$

In Figure 3.2 we show an instance of Case 1. We can see that the first object is found clockwise at arc distance of x_1 from $cycle(0)$ and the second object is found counterclockwise from $cycle(0)$ and an arc distance of x_2. Both objects are found within a time of α. Just as in the Splitting Algorithm there are two possible placements: each object can be located by a distinct robot as seen in Figure 3.2 and both objects can also be located by the same robot, not pictured in Figure 3.2. To determine the worst case cost of the algorithm we need the maximum value of the two instances. Just as in the Splitting Algorithm we need to determine the maximum value of two optimization problems:

When both objects are found by different robots, NLP Case 1a:

$$\begin{aligned} \max_{x_1,x_2} \quad & 1 + x_2 + \mathcal{F}(\mathbf{cycle}(x_2), \mathbf{cycle}(-x_2), \mathbf{cycle}(-x_1)) \\ \text{s.t.} \quad & 0 \le x_1 \le x_2 \le \alpha \le \pi \end{aligned}$$

When each robot finds one object, NLP Case 1b:

$$\begin{aligned} \max_{x_1,x_2} \quad & 1 + x_2 + \mathcal{F}(\mathbf{cycle}(-x_2), \mathbf{cycle}(x_2), \mathbf{cycle}(-x_1)) \\ \text{s.t.} \quad & 0 \le x_1 \le x_2 \le \alpha \le \pi \end{aligned}$$

Since we are looking for the maximum of value from both optimization problems we require,

$$\max[NLPCase1a, NLPCase1b] \tag{3.1.1}$$

We continue in the same manner and present the following three cases with corresponding optimization problems. Case 2 is represent by Figure 5 and modeled by Equation (3.1.2).

Case 2

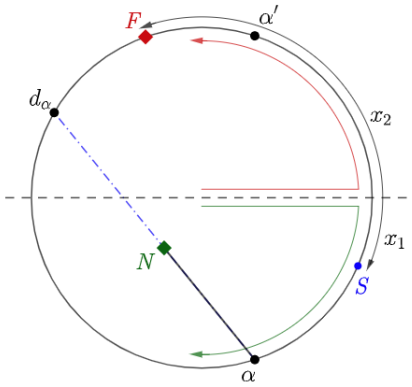

Figure 3.3: Case 2 of $\alpha - RALLY$

In Figure 3.3, we can see that the first object is located at $cycle(-x_1)$ and the second object is found between $cycle(\alpha)$ and $cycle(\alpha + d_\alpha)$ at $cycle(x_2)$. Since the second item is found between time α and $\alpha + d_\alpha$ then the non-finder is traveling along the chord of length d_α once the second item is found.

$$F = cycle(x_2)$$
$$N = \mathcal{L}(cycle(\alpha), cycle(d_\alpha + \alpha), |x_2 - \alpha|)$$
$$S = S = cycle(-x_1)$$

Using the above values and Figure 3.3 to establish constraints we obtain the following

non-linear program,

$$\begin{aligned} \max_{x_1,x_2} \quad & 1 + x_2 + \mathcal{F}[\mathbf{cycle}(x_2), \mathcal{L}(\mathbf{cycle}(-\alpha), \mathbf{cycle}(\alpha + d_\alpha), \mathbf{cycle}(y - \alpha)), \mathbf{cycle}(-x_1)], \\ \text{s.t.} \quad & 0 \le x_1 \le \alpha \\ & \alpha \le x_2 \le \alpha + d_\alpha \end{aligned} \tag{3.1.2}$$

Case 3

We now present the third case and it's points of interests with the aid of Figure 3.4.

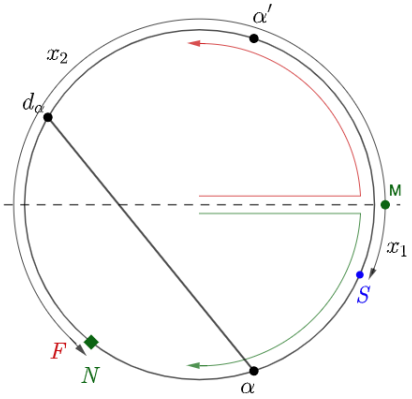

Figure 3.4: Case 3 of $\alpha - RALLY$

In Figure 3.4 we can see that clearly that Case 3 represents when both robots find the second object together in between time $d_\alpha + \alpha$ and $2\pi - \alpha$. The points of interest are

$$F = cycle(x_2)$$

$$N = cycle(x_2)$$

$$S = S = cycle(-x_1).$$

The associated optimization problem is constructed below in Equation (3.1.3).

$$\begin{aligned} \max_{x_1,x_2} \quad & 1 + x_2 + \mathcal{F}[\mathbf{cycle}(x_2), \mathbf{cycle}(x_2), \mathbf{cycle}(-x_1)], \\ \text{s.t.} \quad & 0 \le x_1 \le \alpha \\ & \alpha + d_\alpha \le x_2 \le 2\pi - \alpha \end{aligned} \tag{3.1.3}$$

Case 4

Finally we present Case 4. In case 4 following the same trajectories as the Splitting Algorithm and as Case 1. In Figure 3.5 we show only one instance of Case 4; when each object is found by a different robot. As before, we omit the image of the second instance; when both objects are found by the same robot. In this case, none of the objects are found within time α. So once the robots arrive at α, the decision point, they decide to continue on their respective trajectories. Thus mirroring the Splitting Algorithm.

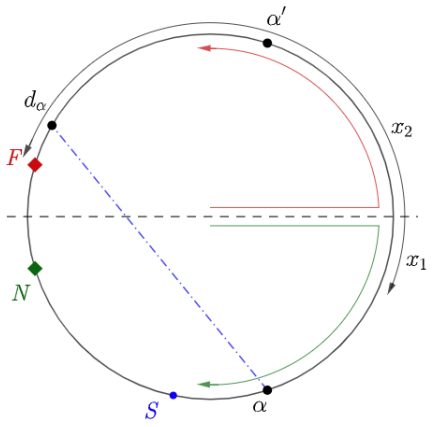

Figure 3.5: Case 4 of $\alpha - RALLY$

We are familiar with interest points in Case 4 and so we establish our two non-linear optimization problems for each instance of Case 4.

When both objects are found by different robots, NLP Case 4a:

$$\begin{aligned} \max_{x_1,x_2} \quad & 1 + x_2 + \mathcal{F}(\mathbf{cycle}(x_2), \mathbf{cycle}(-x_2), \mathbf{cycle}(-x_1)) \\ \text{s.t.} \quad & 0 \le \alpha \le x_1 \le x_2 \le \pi \end{aligned}$$

When each robot finds one object, NLP Case 4b:

$$\begin{aligned} \max_{x_1,x_2} \quad & 1 + x_2 + \mathcal{F}(\mathbf{cycle}(-x_2), \mathbf{cycle}(x_2), \mathbf{cycle}(-x_1)) \\ \text{s.t.} \quad & 0 \leq \alpha \leq x_1 \leq x_2 \leq \pi \end{aligned}$$

Since we require the maximum value from both non-linear programs above, we compute the following

$$\max[NLPCase4a, NLPCase4b] \tag{3.1.4}$$

The role of α in this algorithm serves as a decision point for the robots to either continue on the same trajectories as outlined in the Splitting Algorithm or cut across the circle to save time. Surprisingly, while α as a decision point shows promise it does not in fact save any time. With the aid of software [1] and for the purpose of saving our own time, we determined the cost of the algorithm in all four cases and noted that the cost of $\alpha - RALLY$ is always greater than or equal to the cost of the Splitting Algorithm. While it is obvious that Case 1 and Case 4 behave exactly like the Splitting Algorithm it is not as obvious that Case 3 might sustain higher costs.

The results are summarized in the figures below. In Figure 3.6 we review the worst-case completion time of all instances for $\alpha, \beta - RALLY$. In Figure 3.7 we compare the completion times of the cases $\alpha, \beta - RALLY$ versus the worst case completion time for the Splitting Algorithm. We will soon prove the result that the worst-case completion time for $\alpha - RALLY$ is always greater than or equal to that of the Splitting Algorithm. The promise shown by $\alpha - RALLY$ can be seen in Figures 3.6 and 3.7 where some of the other cases intersect at a completion time below that of the Splitting Algorithm. This led us to believe that a strategic adjustment in the behavior of robots could improve our current standing benchmark.

[1] Mathematica was used to assist in the numerical analysis for Algorithm $\alpha - RALLY$ [20]

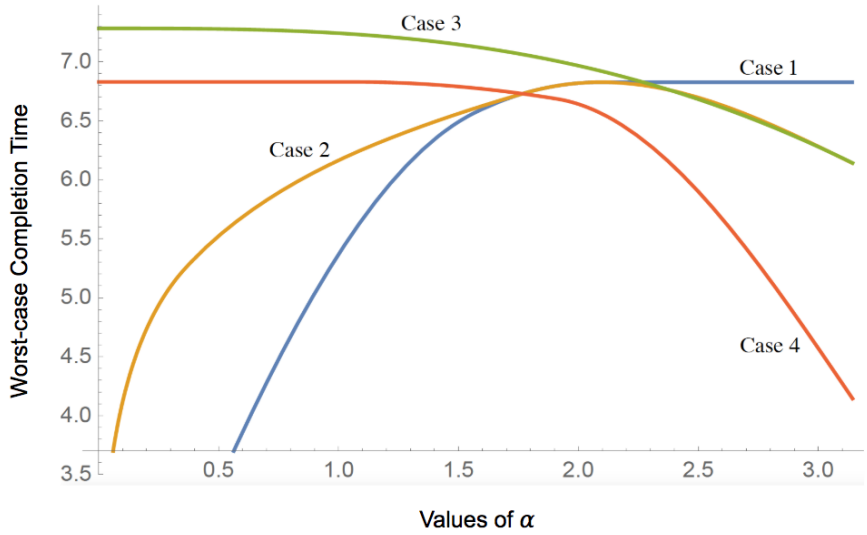

Figure 3.6: Performance of $\alpha - RALLY$

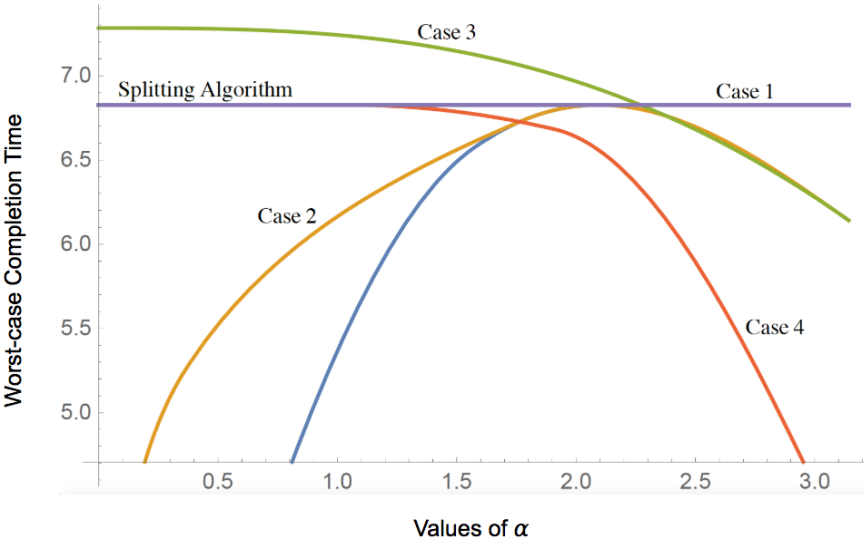

Figure 3.7: Performance of $\alpha - RALLY$ vs. Performance of the Splitting Algorithm

Theorem 3.1.2. $\forall \alpha \in [0, \pi)$ the worst case cost of algorithm $\alpha - RALLY \geq 3 + \frac{2\pi}{3} + \sqrt{3}$.

Proof. Recall that α is a point of juncture for the robots on whether to change trajectories or continue to simulate the Splitting Algorithm. If $\alpha \geq \frac{2\pi}{3}$, then the worst case instance of the Splitting Algorithm induces the same performance in $\alpha - RALLY$.

If $\alpha < \frac{2\pi}{3}$ we prove next that the worst case cost of $\alpha - RALLY > 3 + \frac{2\pi}{3} + \sqrt{3} \approx 6.8264$ (the

Splitting Algorithm). Consider input $[\mathbf{cycle}(-\epsilon), \mathbf{cycle}(-\alpha-\epsilon)]$, for some arbitrarily small $\epsilon > 0$. Then the algorithm performs according to Case 3 with cost,

$$C_A(I_\epsilon) = 1 + y + \mathcal{F}(F, N, S)$$
$$= 1 + 2\pi - \alpha - \epsilon + 2\sin\frac{|2\pi - \alpha - \epsilon - (2\pi - \epsilon)|}{2}.$$

It is trivial that the function is decreasing in α. So

$$\sup_I C_A(I) \geq \sup_\epsilon C_A(I_\epsilon) = 1 + 2\pi - \frac{2\pi}{3} + 2\sin\frac{\pi}{3} = 6.9208 > 3 + \frac{2\pi}{3} + \sqrt{3}.$$

Thus the worst case cost of $\alpha - RALLY$ is always greater than or equal to the Splitting Algorithm. □

So the worst case cost of $\alpha - RALLY$ never beats that of the Splitting Algorithm. We can see from Figures 3.6 and 3.7 that there is room to improve the $\alpha - RALLY$ algorithm in Case 3 - when one item is found before α.

3.2 Two Robots and Tighter Bounds

We now look to an enriched version of the previous algorithm to beat our current benchmark. In $\mathbf{SRCH}_{2,2}$ it is important to benefit from the robots' ability to communicate wirelessly and continuously. So we advance our research with $\alpha, \beta - RALLY$, an algorithm that uses multiple points of decision for the robots. The two decision points are denoted by α and β (where $\beta < \alpha$) and are located on the perimeter. The parameters serve to avoid the larger worse case cost situations that arose in the previous algorithm. The ability of the robots to make optimal decisions at these variable points effectively boosts the performance of $\alpha, \beta - RALLY$ to gain advantage over the Splitting Algorithm. We will quickly introduce the new algorithm and then dive into the analysis.

Algorithm 6 $\alpha, \beta - RALLY$

1: R_1 and R_2 move to **cycle**(0)
2: If at any time both objects are located, R_1 and R_2 initiate $\mathbf{F}(F, N, S)$
3: R_1 travels clockwise and R_2 travels counterclockwise searching until α unless both items are found
4: If only one item is found within time α:
   a: With the item found within time β, then R_1 and R_2 continue on independently
   b: With the item found after time β and before time α then we choose the best candidate to abandon search and travel a distance of d_α to meet the other
5: If no item is found by time α then R_1 and R_2 continue to search independently until both items are found

Theorem 3.2.1. The worst case performance of the $\alpha, \beta - RALLY$ is approximately 6.7518.

We will prove this theorem as follows. First we complete a case by case performance analysis of all instances generated by $\alpha, \beta - RALLY$. Once we establish a series of optimization problems representing the cases we use the help of numerical software [2] to zero in on the best values of α and β. From this we are able to set up a system of non-linear equations dependent on α and β. Solving that system of equations supplies the precise values of α and β that lower the worst case completion time over all instances and beats our benchmark established by the Splitting Algorithm.

[2] Mathematica was used to assist in the numerical analysis for Algorithm $\alpha, \beta - RALLY$ [20]

3.2.1 Performance Analysis

We again, we find it important to observe that Splitting Algorithm makes an appearance in $\alpha, \beta - RALLY$. The algorithm employs the trajectory of the Splitting Algorithm in the following three cases:

- When one item is within time β and the other item is found after time α
- When no items are found within time α
- When both items are found within time α

As before we establish a series of cases accounting for probable placements of all instances to discover the worst case performance of the algorithm. As the algorithm is dependent on two parameters there are many cases to analyze. We will breakdown the analysis into major cases and sub-cases. In each of the analyses we will represent the location of the first object and second object found by x_1 and x_2 which represents the shortest arc distance from $cycle(0)$. There are three main cases to review,

- **Case 0**: 0 items found within α
- **Case 1**: 1 item found within α
- **Case 2**: 2 items found within α

We begin by reviewing Case 0. If 0 items are found within α this means that 0 items are found within β as well. This can be seen below in two sub-cases. Sub-case 0a is when both items are found on the same side and Sub-case 0b is when both items are found on the opposite side.

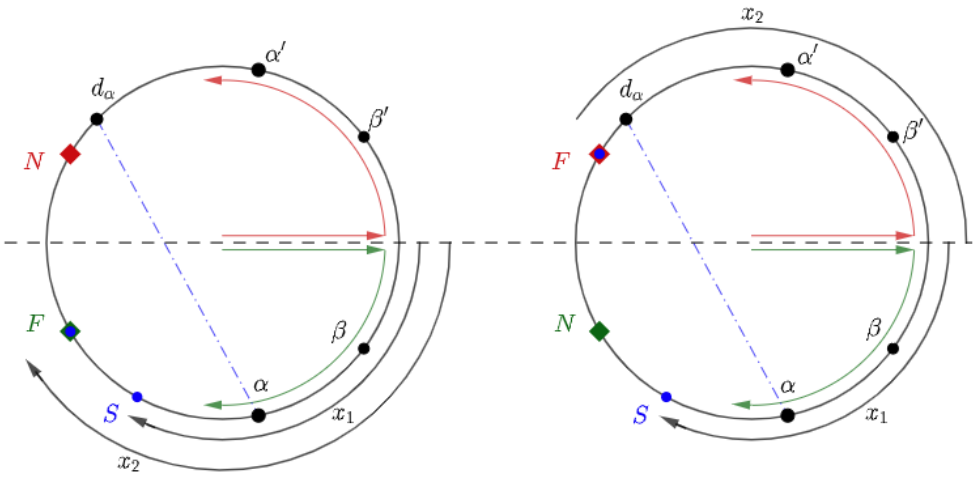

Figure 3.8: Case 0 and its Sub-cases

By now, we are familiar with the process of developping the optimization problem for the cases outlined in Figure 3.8. As in the Splitting Algorithm, we obtain two sub-cases and we require the maximum value case only. We model this here with one optimization problem:

$$\begin{aligned} \max_{x_1,x_2} \quad & 1 + \min\{x_2, 2\pi - x_2\} + 2\sin\tfrac{x_1+x_2}{2} + 2\min\begin{cases} 2\sin x_2, \\ \max\{\sin\tfrac{x_2-x_1}{2}, \sin\tfrac{x_2+x_1}{2}\} \end{cases} \\ \text{s.t.} \quad & \alpha < x_1 \leq \pi \\ & x_1 \leq x_2 \leq 2\pi - x_1 \end{aligned} \tag{3.2.1}$$

We now describe Case 2 and its sub-cases. Case 2 has three plausible scenarios. When both objects are found within time β. When both objects are found after time β and before time α and when one item is found within time β and the second one after time β and before time α. Since the cases and sub-cases are plentiful, for clarity we will show the arc intervals where the objects' arc distances are located in the following table.

Sub-Cases for Case 2 in $\alpha, \beta - RALLY$			
	Sub-case 2a	**Sub-case 2b**	**Sub-case 2c**
Location of first item found	$0 \leq x_1 \leq x_2$	$0 \leq x_1 \leq \beta$	$\beta < x_1 \leq x_2$
Location of second item found	$x_1 \leq x_2 \leq \beta$	$\beta < x_2 \leq \alpha$	$x_1 \leq x_2 \leq \alpha$
Description of Sub-cases	Both items found within time β	one item found in time β and 2nd item found in time α	Both items found after β and before α

Table 3.1: All possible instances of Case 2 for $\alpha, \beta - RALLY$

Since the decision points for trajectory changes for R_1 and R_2 occur at α and β then the above cases are similar to the Splitting Algorithm as well, however with the constraints outlined in Table 3.1 and so an image here is trivial. Sub-cases 2a and 2c can be modeled by one optimization problem where both items are found within time α. We present the below optimization problem describing both sub-cases.

$$\begin{aligned} \max_{x_1,x_2} \quad & 1 + \min\{x_2, 2\pi - x_2\} + 2\sin\tfrac{x_1+x_2}{2} + 2\min\begin{cases} 2\sin x_2, \\ \max\{\sin\tfrac{x_2-x_1}{2}, \sin\tfrac{x_2+x_1}{2}\} \end{cases} \\ \text{s.t.} \quad & 0 < x_1 \leq x_2 \leq \alpha \end{aligned} \tag{3.2.2}$$

Before presenting the optimization problem for Sub-case 2b we must introduce the sub-cases for Case 1. Case 1 is when only one item is found within time α. There are two distinct possibilities; when one item is found within time β and when one item is found within time β and after time α. The first of the sub-cases is when one object is found within β, and will be denoted by Sub-case 1c. This is the sub-case that aligns with Sub-case 2b. In both sub-cases we have one item found within time β and the second item is somewhere else on the perimeter anytime after β. This situation is represented by the following optimization problem:

$$\begin{aligned} \max_{x_1,x_2} \quad & 1+\min\{x_2, 2\pi - x_2\} + 2\sin\tfrac{x_1+x_2}{2} + 2\min\begin{cases} 2\sin x_2, \\ \max[\sin\tfrac{x_2-x_1}{2}, \sin\tfrac{x_2+x_1}{2}] \end{cases} \\ \text{s.t.} \quad & 0 < x_1 \le \beta \\ & \beta \le x_2 \le 2\pi - \beta \end{aligned} \tag{3.2.3}$$

At this time all robot trajectories described above match the Splitting Algorithm with different set of constraints. Since we are familiar with solving these problems we will proceed with establishing the final sub-cases for $\alpha, \beta - RALLY$. We are left with the following four instances in which the trajectories of the robots follow a different path than the one we are used to.

Sub-Cases for Case 1 in $\alpha, \beta - RALLY$		
	Sub-case 1a_1	**Sub-case 1**b_1
Location of first item found	$\beta \le x_1 \le \alpha$	$\beta \le x_1 \le \alpha$
Location of second item found	$\alpha \le x_2 \le \alpha + d_\alpha$	$2\pi - \alpha - d_\alpha \le x_2 \le 2\pi - \alpha$
Description	R_1 finds the first item and R_2 finds the second before R_1 arrives at **cycle**(d_α)	R_1 finds the first item and the second before R_2 arrives at **cycle**($-d_\alpha$)
	Sub-case 1a_2	**Sub-case 1**b_2
Location of first item found	$\beta \le x_1 \le \alpha$	$\beta \le x_1 \le \alpha$
Location of second item found	$\alpha + d_\alpha \le x_2 \le 2\pi - \alpha$	$\alpha \le x_2 \le 2\pi - \alpha - d_\alpha$
Description	R_1 finds the first item, travels along the chord of length d_α and together they find the second item	R_1 finds the first item, R_2 travels along the chord of length d_α and together they find the second item

Table 3.2: Sub-cases for Case 1 of $\alpha, \beta - RALLY$

We can establish four non-linear programs each representing one of the sub-cases above using the location intervals in Table 3.2 to build the feasible regions of the optimization problems.

NLP 1a_1

$$\begin{aligned} \max_{x_1,x_2} \quad & 1 + x_2 + \mathcal{F}[\text{cycle}(x_2), \mathcal{L}[\text{cycle}(-\alpha), \text{cycle}(\alpha + d_\alpha), x_2 - \alpha], \text{cycle}(-x_1)] \\ \text{s.t.} \quad & \beta \leq x_1 \leq \alpha \\ & \alpha \leq x_2 \leq \alpha + d_\alpha \end{aligned}$$

NLP 1a_2

$$\begin{aligned} \max_{x_1,x_2} \quad & 1 + x_2 + 2\sin\tfrac{x_1+x_2}{2} \\ \text{s.t.} \quad & \beta \leq x_1 \leq \alpha \\ & \alpha + d + \alpha \leq x_2 \leq 2\pi - \alpha \end{aligned}$$

NLP 1b_1

$$\begin{aligned} \max_{x_1,x_2} \quad & 1 + x_2 + \mathcal{F}[\text{cycle}(x_2), \mathcal{L}[\text{cycle}(\alpha), \text{cycle}(2\pi - \alpha - d_\alpha), 2\pi - \alpha - x_2], \text{cycle}(-x_1)] \\ \text{s.t.} \quad & \beta \leq x_1 \leq \alpha \\ & 2\pi - \alpha - d_\alpha \leq x_2 \leq 2\pi - \alpha \end{aligned}$$

NLP 1b_2

$$\begin{aligned} \max_{x_1,x_2} \quad & 1 + x_2 + 2\sin\tfrac{x_1+x_2}{2} + 2\sin\tfrac{x_2+x_1}{2} \\ \text{s.t.} \quad & \beta \leq x_1 \leq \alpha \\ & \alpha \leq x_2 \leq 2\pi - \alpha - d_\alpha \end{aligned}$$

We obtain the final optimization problem for Case 1:

$$\max_{x_1,x_2} \quad \min\{\max\{NLP1a_1, NLP1a_2\}, \max\{NLP1b_1, NLP1b_2\}\} \tag{3.2.4}$$

The intention of Case 1 above is to find the minimum value of two possibilities:

- The discoverer of the first item forgoes their search and meets their comrade
- The non-finder of the first item forgoes their search and meets the discoverer

In $\alpha - RALLY$ the robots where not given this extra choice. Equations 3.2.1, 3.2.2, 3.2.3 and 3.2.4 holistically describe all instances of the treasures and the trajectories of the robots

in $\alpha, \beta - RALLY$. Our goal is to choose the best values for α and β that minimize our worst case cost among all instances so that we can achieve a lower upper bound than the one established by the Splitting Algorithm. We chose our α and β with the aid of numerical software [3]. Plotting all instances for all values of α and β and with much trial and error we were able to zero in on the optimal values. We fix $\beta = 0.424$ and observe the behavior of the functions for for various values of α. We see can see clearly in Figure 3.9 below that there are three cases for which the performance of the algorithm is equivalent. Coincidentally, that is also where cost is at a minimum in the worst case among all instances. It is for that performance we choose our best α and β.

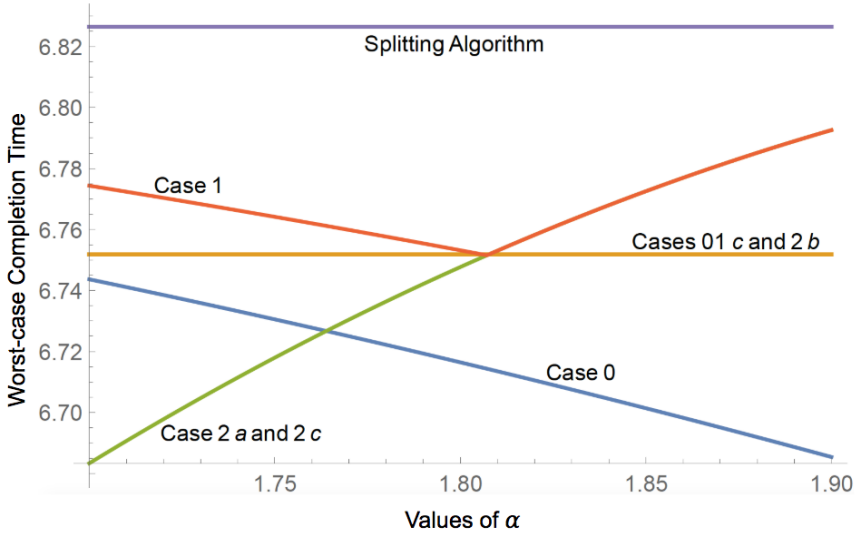

Figure 3.9: Worst-case Analysis Results of the Splitting Algorithm

We identify the cases as Sub-case $1a_1$, Sub-case $2c$ and Sub-case $1c$. Furthermore, we are able to identify the maximizers in these sub-cases that minimize the worst-case cost.

Sub-case $1a_1$

The first item is found after time β and before α. The second item is found at $2\pi - \alpha$.

[3] Mathematica was used to assist in the numerical analysis for Algorithm $\alpha, \beta - RALLY$ [20]

The cost of this instance is:

$$C_1(I) = 1 + 2\pi - \alpha + 2\sin\frac{\alpha - \beta}{2}$$

Sub-case $2c$

Both items are found within time α. The first item is found within time $\pi - \alpha$ and the second item is found at time α. The cost of this instance is:

$$C_2(I) = 2 + \alpha + 2\sin\alpha$$

Sub-case $1c$

The first item is found at time β. The second item is found at some time y, where $\beta \leq y \leq 2\pi - \beta$. The cost of this instance is:

$$C_3(I) = 1 + y + 2\sin y + 2\sin\frac{y + \beta}{2}$$

Both $C_1(I)$ and $C_2(I)$ are functions of α and β. To find the optimal value of y at the precise intersection point above we must solve for y by letting the derivative equal 0. We obtain the following system of non-linear equations,

$$\begin{pmatrix} 1 + 2\cos y + \cos\frac{y+\beta}{2} == 0 \\ 1 + \alpha + 2\sin\alpha + 2 == 1 + 2\pi - \alpha + 2\sin\frac{\alpha-\beta}{2} \\ 1 + 2\pi - \alpha + 2\sin\frac{\alpha-\beta}{2} == 1 + y + 2\sin y + 2\sin\frac{y+\beta}{2} \end{pmatrix}.$$

The numerical results are summarized in Table 3.3.

Optimal Solution for $\alpha, \beta - RALLY$	
Value of y	2.2381
Value of α	1.8076
Value of β	0.4236
Cost of Algorithm	6.7519

Table 3.3: Results for $\alpha, \beta - RALLY$

Therefore we choose α = 1.8076 and β = 0.4236 so that the worst case performance of $\alpha, \beta - RALLY$ is less than the Splitting Algorithm at approximately 6.7519.

Chapter 4

The General Problem

We complete our research with generalizing our model and providing some bounds for asymptotic values of n. Let us recall our model $\mathbf{SRCH}_{n,k}$ in which we have n mobile agents searching for two immobile items hidden on a circle. It is a gathering problem where the movement of objects is restricted by the number of agents required to carry them. The number of mobile agents needed to carry objects is k (where $k \leq n$). The mobile agents are robots searching for objects and termination occurs once at least k robots and all treasures are in the same location. The domain is a unit disk with both treasures hidden on the perimeter. Robots can communicate wirelessly and instantly with one another to announce the discovery of treasures and thus are not required to travel together. All robots move at the same speed of 1 and always begin searching at the center of disk. We generalize the problem by letting n be any integer, while examining only two values for k, $k = 1$ or $k = 2$. We again explore the cost, the time it takes to locate both treasures and bring them together. Our search is completed once both items are in the same location.

4.1 Upper Bounds for the General Problem

Let us begin the exploration of the general case with an example. As usual, consider two robots and two treasures. Suppose now that only one robot is needed to carry any given treasure. In particular this problem is denoted by $\mathbf{SRCH}_{2,1}$.

Theorem 4.1.1. There exists an algorithm for $\mathbf{SRCH}_{2,1}$ with a worst-case performance of $2 + \pi$.

Algorithm 7 Solo Hunt

1: Both robots start at the center of the disk
2: R_1 moves to **cycle**(π) and R_2 moves to **cycle**(0)
3: Both R_1 and R_2 travel counterclockwise
4: Whenever either robot finds an item they carry it with them
5: The robots meet in the center once they've both searched an arc distance of π

We will now prove Theorem 4.1.1.

Proof. It takes time 1 to move to the perimeter of the disk. Robots must search an arc distance of π regardless if they find both items early on or not. The robots are both starting their respective search at antipodal points, travel in the same direction and move at the same speed. Therefore they are always at antipodal points from one another. Once time π has passed, the robots have searched the entire perimeter of the disk and have carried the items independently. At the end of their journeys they are both at the antipodal point from whence they commenced. Thus the total time passed for the algorithm to terminate is $1 + \pi + 1 = 2 + \pi$. □

The above algorithm is not the most sophisticated for $\mathbf{SRCH}_{2,1}$. We could suggest that once any item is found the finder cuts across the disk to meet their companion and they continue the search together. However, we can instantly see that the worst case performance here, would indeed be larger than the simple algorithm above. We could imagine the first item being placed just a short distance counterclockwise from $cycle(\pi)$ and the next one small distance from $cycle(0)$ in the clockwise direction. In this case the robots would not

find the second item until they have already searched the majority of perimeter, and so the worst case cost would approach $1 + 2\pi$!

Additionally, consider an algorithm the same as above, except when both items are found the robots decide to meet along the shortest chord length between them. However, if that was the case the worst case placement of the objects would occur at the last two antipodal points possible; once each robot has already searched a distance of π. The worst case cost would still be the same as in the Solo Hunt Algorithm! When $k = 1$, and n grows asymptotically large, we will observe that a minimal worst case performance is often achieved with an unsophisticated algorithm. We now present our first general problem and an algorithm that complements the problem.

Problem 1: SRCH$_{n,1}$

Algorithm 8 Scatter Search 1

1: Robots $R_1, R_2, ..., R_n$ start at the center of the disk, move to the perimeter and scatter at an arc distance of $\gamma = \frac{2\pi}{n}$ from each other
2: All robots travel counterclockwise in search of both items
3: When any item is found it is carried by the finder and the robots continue searching in the same direction until the second item is found
4: Once the second item is found the finders meet as soon as possible

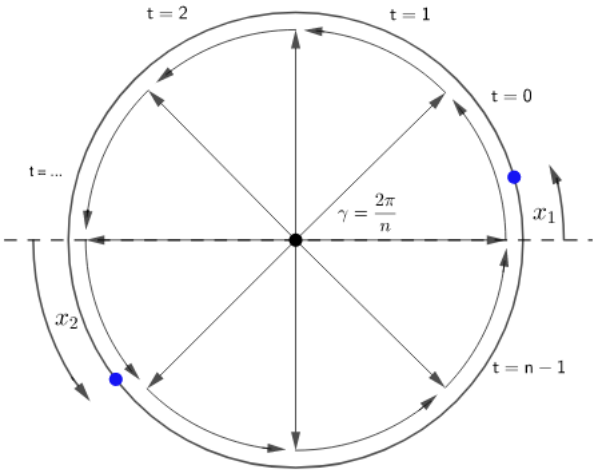

Figure 4.1: Scatter Search 1

Above we show the movement of n robots in the algorithm Scatter Search 1 and the position once all objects are found. All n robots start at the center and travel to the perimeter at distance of $\gamma = \frac{2\pi}{n}$ from one another. We show an instance where the first object is found in section $t = 0$ after a time of x_1 of searching and the second object is found in section $t > 1$ after time x_2 of searching, where $x_2 > x_1$.

Theorem 4.1.2. The worst case performance of algorithm Scatter Search 1 is $1+\frac{2\pi}{n}+\sin\left[\lfloor\frac{n}{2}\rfloor\frac{\pi}{n}\right]$ for $n \geq 2$ and for $k = 1$.

Proof. Consider algorithm Scatter Search 1. It takes time 1 for n robots to move simultaneously from the circle center to the perimeter. In Scatter Search 1, all robots are located at an angle distance of γ from each other and search a maximum distance of γ, (where $\gamma = \frac{2\pi}{n}$). These trajectories divide the circle into n equally long arcs which we will call sections. WLOG let the first item be found in section 0. Then the second item will be located after the first item in section $t = 0, ..., n-1$. Then the time it takes for Scatter Search 1 to terminate is,

$$\begin{aligned} \sup_{x_1,x_2,t} \quad & 1 + x_2 + \sin t\tfrac{\gamma}{2} \\ \text{s.t.} \quad & 0 \leq x_2 < \gamma \\ & t = 0, 1, ..., n-1 \end{aligned}$$

As usual we will adopt an optimization perspective for the problem.

$$\begin{aligned} \max_{x_1,x_2} \quad & 1 + x_2 + \sin t\tfrac{\pi}{n} \\ \text{s.t.} \quad & 0 \leq x_2 \leq \gamma \\ & t = 0, 1, ..., n-1 \end{aligned} \tag{4.1.1}$$

The maximum value x_2 can take is γ. So our optimization problem becomes:

$$\begin{aligned} \max_{t} \quad & 1 + \tfrac{2\pi}{n} + \sin t\tfrac{\pi}{n} \\ \text{s.t.} \quad & t = 0, 1, ..., n-1 \end{aligned}$$

We are looking for values of t that maximize the argument, $\arg\max_t \sin\left(t\frac{\pi}{n}\right)$, where $t =$

$0, ..., n-1$. Let $f(t) = \sin(t\frac{\pi}{n})$. We must consider values of t when n is odd and when n is even. Suppose n is odd, then $n = 2s+1$, where $s \in \mathbb{N}$. If $t\frac{\pi}{n} \leq \frac{\pi}{2}$ then $f(t)$ is increasing and $t \leq \frac{n}{2} = s + \frac{1}{2}$. If $t\frac{\pi}{n} \geq \frac{\pi}{2}$ then $f(t)$ is decreasing and $t \geq \frac{n}{2} = s + \frac{1}{2}$. But $t \in \mathbb{Z}$ so,

$$\lfloor \frac{n}{2} \rfloor \leq t \leq \lceil \frac{n}{2} \rceil \text{ or } s \leq t \leq s+1$$

Since t is an integer then we know that either $t = \lfloor \frac{n}{2} \rfloor$ or $t = \lceil \frac{n}{2} \rceil$. We must show that

$$\sin(s\frac{\pi}{n}) = \sin((s+1)\frac{\pi}{n}).$$

But this is true, if and only if

$$\frac{\pi}{2} - s\frac{\pi}{n} = (s+1)\frac{\pi}{n} - \frac{\pi}{2}$$

If and only iff

$$s\frac{\pi}{n} + (s+1)\frac{\pi}{n} = \pi$$

So,

$$s\frac{\pi}{n} + (s+1)\frac{\pi}{n} = s\frac{\pi}{n} + s\frac{\pi}{n} + \frac{\pi}{n} = 2s\frac{\pi}{n} + \frac{\pi}{n} = (2s+1)\frac{\pi}{n} = \pi$$

Now suppose n is even, then $n = 2s$, where $s \in \mathbb{N}$. If $t\frac{\pi}{n} \leq \frac{\pi}{2}$ then $f(t)$ is increasing and $t \leq \frac{n}{2} = s$. If $t\frac{\pi}{n} \geq \frac{\pi}{2}$ then $f(t)$ is decreasing and $t \geq \frac{n}{2} = s$. Thus $t = \frac{n}{2}$. In either case $t = \lfloor \frac{n}{2} \rfloor$ and so the cost of the algorithm is $1 + \frac{2\pi}{n} + \sin\lfloor \frac{n}{2} \rfloor \frac{\pi}{n}$. □

As previously noted, in the example with the algorithm Solo Hunt, it is not necessary to have the finders meet as soon as both items are found. The worst case cost is the same in both cases. In Scatter Search 1 all robots could search an arc distance of γ regardless of when the objects are found and the worst case performance of the algorithm would remain unchanged. We leave the exploration of a lower to the next section and we continue on to our second general problem.

Problem 2: $\mathbf{SRCH}_{n,2}$

Algorithm 9 Scatter Search 2

1: Robots $R_1, R_2, ..., R_n$ start at the center of the disk, move to the perimeter and scatter at an arc distance of $\gamma = \frac{2\pi}{n}$ from each other
2: All robots travel counterclockwise in search of both items
3: WLOG R_1 finds the first item and announces it's location to the other robots
4: Then the second item is found by R_i where $i = 1, 2, ..., n$
5: Once the second item is found by R_i, R_{i+1} meets R_i and together they travel the shortest distance to the first item

We begin as usual by modeling the time complexity of the algorithm with an optimization problem. Visualizing the problem we obtain the following placement of objects and robots once both objects have been located.

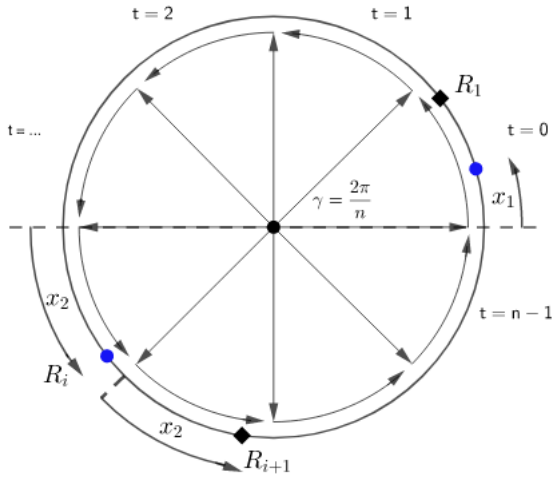

Figure 4.2: Scatter Search 2

In Figure 4.2 we show the movement of n robots in the algorithm Scatter Search 2 and their positions once all objects are found. As in Scatter Search 1, all n robots start at the center and travel to the perimeter at distance of $\gamma = \frac{2\pi}{n}$ from one another. We show a possible instance of the objects where the first object is found in the first section counterclockwise at an arc distance of x_1 from $cycle(0)$ and the second object is found in another section counterclockwise at an arc distance of x_2 from $cycle(2\pi)$. Note that since all robots travel at

the same speed they remain separated at an arc distance of γ from one another. Thus when the second object is found they are all still separated by γ.

Theorem 4.1.3. The worst case performance of Scatter Search 2 is $3 + \frac{2\pi}{n} + \sin\frac{\pi}{n}$.

Proof. Consider algorithm Scatter Search 2. It takes time 1 for n robots to move simultaneously from the circle center to the perimeter. All n robots are constantly moving at a distance of γ from each other and search a maximum distance of γ, (where $\gamma = \frac{2\pi}{n}$). WLOG let the first item be found in section $t = 0$. Then the second item will be located after the first item in section $t = 0, ..., n-1$. Then the time it takes for Scatter Search 2 to terminate is,

$$\begin{aligned} \max_{x_1,x_2,t} \quad & 1 + x_2 + 2\sin\tfrac{\gamma}{2} + 2\sin\tfrac{x_2 - x_1 + t\gamma}{2} \\ \text{s.t.} \quad & 0 \le x_1 \le x_2 \le \gamma \\ & t \in \{0, 1, 2, ..., n-1\} \end{aligned} \tag{4.1.2}$$

Let $f(x_1, x_2, t) = 1 + x_2 + 2\sin\frac{\gamma}{2} + 2\sin\frac{x_2 - x_1 + t\gamma}{2}$, where t represents the section in which the second object is found. Note that the maximum value for x_2 is γ and the maximum value that $\sin\frac{x_2 - x_1 + t\gamma}{2}$ can assume is 1.

We give an instance when n is both odd and even that produces the worst case completion time of $3 + \frac{2\pi}{n} + \sin\frac{\pi}{n}$. If n is odd consider the following instance, $x_2 = \frac{2\pi}{n}$, $x_1 = \frac{\pi}{n}$ and $t = \lfloor \frac{n}{2} \rfloor$. Let $n = 2s + 1$ where $s \in \mathbb{N}$. Inputting this instance into f, we obtain

$$\begin{aligned} f &= 1 + \frac{2\pi}{n} + 2\sin\frac{\pi}{n} + 2\sin\frac{\frac{\pi}{n} + s\frac{2\pi}{n}}{2} \\ &= 1 + \frac{2\pi}{n} + 2\sin\frac{\pi}{n} + 2\sin\frac{\frac{\pi}{n}(1 + 2s)}{2} \\ &= 1 + \frac{2\pi}{n} + 2\sin\frac{\pi}{n} + 2\sin\frac{\pi}{2} \\ &= 3 + \frac{2\pi}{n} + 2\sin\frac{\pi}{n} \end{aligned}$$

Suppose now that n is even and so $n = 2s$, for some $s \in \mathbb{N}$. Considering the following instance where $x_2 = x_1 = \frac{2\pi}{n}$ and $t = \lfloor \frac{n}{2} \rfloor$, we see that

$$
\begin{aligned}
f &= 1 + \frac{2\pi}{n} + 2\sin\frac{\pi}{n} + 2\sin\frac{s\frac{2\pi}{2s}}{2} \\
&= 1 + \frac{2\pi}{n} + 2\sin\frac{\pi}{n} + 2\sin\frac{\pi}{2} \\
&= 3 + \frac{2\pi}{n} + 2\sin\frac{\pi}{n}
\end{aligned}
$$

□

4.2 Lower Bounds for the General Problem

Finally, we explore lower bounds for the general cases where $n > 3$ and $k = 1, 2$.

Theorem 4.2.1. For all $n \geq 2$ and $k = 1$ the performance of any algorithm is at least $2 + \frac{\pi}{n}$.

Proof. Let $\epsilon \in \mathbb{R}$, where $\epsilon > 0$. Fix an arbitrary algorithm A and let A run for $1 + \frac{\pi}{2} - \epsilon$ time. It takes time 1 for both robots to move to the perimeter of the unit circle. After time $\frac{\pi}{2} - \epsilon$ the robots have searched at most $n(\frac{\pi}{n} - \epsilon) = \pi - n\epsilon$ of the perimeter. Consider a diameter l_θ, that forms angle θ with the x-axis. We define $S \subseteq [0, 2\pi)$ as follows, $\theta \in S$ if and only if at least one of the endpoints of l_θ has been explored. $\theta \in S$ if and only if at least one of the endpoints of l_θ has been explored. If at least one endpoint of all antipodal points has been explored, then $[0, \pi] \subseteq S$. However, that means at least π of the perimeter has been searched. Which is a contradiction as the robots have searched at most $\pi - n\epsilon$ of the perimeter. Therefore at time $1 + \frac{\pi}{2} - \epsilon$, $\exists$ two antipodal points on the circle, both of which have not been explored. We place the objects there and observe that it takes at least time 1 to bring the objects together as only one robot is needed to carry one object. □

The proof of the following theorem is analogous to that of Theorem 2.3.2 and Theorem 4.2.1.

Theorem 4.2.2. For all $n \geq 3$ and $k = 2$ the performance of any algorithm is at least $3 + \frac{\pi}{n}$.

Chapter 5

Conclusion and Open Problems

We introduced a particular gathering problem and some fundamental bounds. $\mathbf{SRCH}_{n,k}$ is a gathering problem centered around bringing two hidden objects together and where mobile agents can communicate wirelessly. We constructed upper and lower bounds covering all values of n and two values k. An evident next step would be to tighten and close the gap between said bounds. Admittedly, this addition would not aid in the construction of an exhaustive compendium. There are several other related and compelling problems one might tackle:

1. The same gathering problem in the face-to-face model
2. Exploring the competitive ratio analysis for $\mathbf{SRCH}_{n,k}$
3. Generalizing the model for asymptotic values of k, where $k \leq n$
4. Assigning weights to objects by cogitating different values of k for each object
5. Exploring a polygon domain with m sides of equal length, and letting $m \longrightarrow \infty$ to see if comparable with the disk version

We note again the considerable combinations that can be derived. Another logical direction would be to authorize randomness in terms of robot trajectories with the intent of improving bounds.

Bibliography

[1] Susanne Albers and Monika R Henzinger. Exploring unknown environments. *SIAM Journal on Computing*, 29(4):1164–1188, 2000.

[2] Susanne Albers, Klaus Kursawe, and Sven Schuierer. Exploring unknown environments with obstacles. *Algorithmica*, 32(1):123–143, 2002.

[3] Steve Alpern, Robbert Fokkink, Leszek Gasieniec, Roy Lindelauf, and V. S. Subrahmanian. *Search Theory: A Game Theoretic Perspective*. Springer New York, 2013.

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

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

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

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

[8] Anthony Bonato. *The game of cops and robbers on graphs*. American Mathematical Soc., 2011.

[9] Sebastian Brandt, Felix Laufenberg, Yuezhou Lv, David Stolz, and Roger Wattenhofer. Collaboration without communication: Evacuating two robots from a disk. In *International Conference on Algorithms and Complexity*, pages 104–115. Springer, 2017.

[10] Huda Chuangpishit, Konstantinos Georgiou, and Preeti Sharma. Average case-worst case tradeoffs for evacuating 2 robots from the disk in the face-to-face model. In *International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics*, pages 62–82. Springer, 2018.

[11] Jurek Czyzowicz, Konstantinos Georgiou, Maxime Godon, Evangelos Kranakis, Danny Krizanc, Wojciech Rytter, and Michał Włodarczyk. Evacuation from a disc in the presence of a faulty robot. In *International Colloquium on Structural Information and Communication Complexity*, pages 158–173. Springer, 2017.

[12] Jurek Czyzowicz, Konstantinos Georgiou, Ryan Killick, Evangelos Kranakis, Danny Krizanc, Lata Narayanan, Jaroslav Opatrny, and Sunil Shende. God save the queen. *arXiv preprint arXiv:1804.06011*, 2018.

[13] Jurek Czyzowicz, Konstantinos Georgiou, Ryan Killick, Evangelos Kranakis, Danny Krizanc, Lata Narayanan, Jaroslav Opatrny, and Sunil Shende. Priority evacuation from a disk using mobile robots. In *International Colloquium on Structural Information and Communication Complexity*, pages 392–407. Springer, 2018.

[14] Jurek Czyzowicz, Evangelos Kranakis, Danny Krizanc, Lata Narayanan, and Jaroslav Opatrny. Search on a line with faulty robots. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, pages 405–414. ACM, 2016.

[15] Jurek Czyzowicz, Evangelos Kranakis, Danny Krizanc, Lata Narayanan, Jaroslav Opatrny, and Sunil Shende. Wireless autonomous robot evacuation from equilateral triangles and squares. In *International Conference on Ad-Hoc Networks and Wireless*, pages 181–194. Springer, 2015.

[16] Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. *Distributed Computing by Mobile Entities: Current Research in Moving and Computing*, volume 11340. Springer, 2019.

[17] Konstantinos Georgiou, George Karakostas, and Evangelos Kranakis. Search-and-fetch with 2 robots on a disk: wireless and face-to-face communication models. *arXiv preprint arXiv:1611.10208*, 2016.

[18] Attila Hideg, Tamás Lukovszki, and Bertalan Forstner. Filling arbitrary connected areas by silent robots with minimum visibility range. In *International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics*, pages 193–205. Springer, 2018.

[19] Frank Hoffmann, Christian Icking, Rolf Klein, and Klaus Kriegel. The polygon exploration problem. *SIAM J COMPUT*, 31(2):577–600, 2002.

[20] Wolfram Research, Inc. Mathematica, Version 12.0. Champaign, IL, 2019.

[21] Richard Nowakowski and Peter Winkler. Vertex-to-vertex pursuit in a graph. *Discrete Mathematics*, 43(2-3):235–239, 1983.

[22] Debasish Pattanayak, H Ramesh, Partha Sarathi Mandal, and Stefan Schmid. Evacuating two robots from two unknown exits on the perimeter of a disk with wireless communication. In *Proceedings of the 19th International Conference on Distributed Computing and Networking*, page 20. ACM, 2018.

[23] Paweł Prałat and Nicholas Wormald. Meyniel's conjecture holds for random graphs. *Random Structures & Algorithms*, 48(2):396–421, 2016.

[24] Alain Quilliot. Jeux et pointes fixes sur les graphes. *PhD Dissertation*, 1978.