Searching with Two-Speed Autonomous Mobile Robots

by

Fraser Lachlan MACQUARRIE

*A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs
in partial fulfillment of the requirements for the degree of*

Doctor of Philosophy

in

Computer Science

CARLETON UNIVERSITY

OTTAWA, ONTARIO

© 2015

Fraser Lachlan MacQuarrie

Abstract

We introduce and study various search problems using mobile robots. Specifically we study the problems of exploration and patrolling using a new *two-speed* model of robots, as well as studying the problems of rendezvous and evacuation using a more traditional one-speed model of mobile robots. The problems are executed on a continuous finite domain by a swarm of n robots $r_1, r_2, \dots, r_n$. For the two-speed model, each robot r_i has a *walking speed* w_i and a *searching speed* s_i , where $s_i < w_i$. At each moment a robot either moves through the domain with a speed not exceeding w_i , or it performs a more elaborate task (such as searching) on the domain with speed not exceeding s_i .

A search of an interval is completed when each of its points have been searched by at least one of the n two-speed robots. We want to develop efficient *mobility schedules* (algorithms) for the robots which complete the search of the interval as quickly as possible. More exactly, we want to maximize the *speed* of the mobility schedule – defined as the ratio of the interval length compared to the time of completion of the schedule.

For patrolling, a unit interval is to be patrolled collectively by n two-speed robots. A robot patrols a portion of the domain by searching it. Each robot is allowed to patrol only in one of the two directions (not necessarily the same for all robots). We want to schedule the perpetual movements of the robots to minimize the *idle time*, defined as the maximal time interval any point is not visited by some patrolling robot.

We also study randomized rendezvous on a circle: Two one-speed robots begin at different locations on a circle and want to minimize the time required to find each other. The robots have different speeds, are unaware of their own speed, are equipped with identical chronometers, and each have access to one or more random bits.

Finally, we study the evacuation problem: Distributed on a unit circle are k exits; two identical one-speed robots are placed on the circle. The goal of the *evacuation problem* is to give an algorithm for the robots which minimizes the time required for both robots to reach an exit, in the worst case.

Acknowledgements

First and foremost, I would like to extend my deepest gratitude to my supervisor, Dr. Evangelos Kranakis, for his encouragement and advice during the course of my studies. His passion for computer science, extensive knowledge and positive outlook has been inspirational and helped me grow as researcher and person.

I would also like to sincerely thank everyone who has collaborated on the research presented in this thesis.

Finally, I am very thankful for the love and support of my friends and family who have given me the confidence and determination to achieve my academic and non-academic goals.

Contents

Abstract	ii
Acknowledgements	iii
Contents	iv
List of Tables	vii
List of Figures	viii
List of Algorithms	ix
1 Introduction	1
1.1 Searching with Two-Speed Robots	2
1.2 Two-Speed Robot Model	4
1.3 Problem Domains	5
1.4 Problems Studied and Summary of Results	6
1.5 Related Work	12
1.6 Publications	22
2 Searching	25
2.1 Introduction	25
2.1.1 Preliminaries and Notation	26
2.1.2 Search Problems	28
2.1.3 Robot Model for Searching	30
2.1.4 Outline and Results	30
2.2 The Beachcombers' Problem	32
2.2.1 Properties of Optimal Schedules	32
2.2.2 The Optimal Schedule	40
2.2.3 Closed Formulas for the Optimal Schedule	42

2.3	The Online Beachcombers' Problem	45
2.3.1	Competitiveness of the Online Searching	48
2.3.2	Online Searching with W-uniform swarms	51
2.3.3	Online Searching for Totally Uniform Swarms	56
2.4	The Multi-source Beachcombers' Problem	60
2.4.1	t -SBP as a Partitioning Problem and its Hardness	60
2.4.1.1	A Deterministic Approximation Guarantee for 2-SBP	65
2.4.1.2	An Exact Solution for t -SBP with S-uniform Swarms	68
2.4.2	A Randomized Algorithm for t -SBP	72
2.5	Conclusion and Open Problems	78
3	Patrolling	80
3.1	Introduction	80
3.1.1	Preliminaries and Notation	82
3.1.2	Patrolling Problems	82
3.1.3	Robot Model for Patrolling	84
3.1.4	Outline and Results	85
3.2	Optimal Fence Patrolling Centralized Algorithm	86
3.2.1	Schedule Obtained by a Centralized Algorithm	86
3.2.2	Lower Bound of the Centralized Schedule	87
3.2.3	Application to Transportation: Scheduling with Regular Delivery	89
3.3	Fence Patrolling in the Distributed Model	90
3.3.1	Distributed Optimal Schedule for Two Robots	91
3.3.2	Distributed Schedule for n Primitive Robots	95
3.3.3	Dynamics for Regular Systems of Primitive Robots	97
3.3.4	Monotone Robot Collections, and Convergence	104
3.4	Conclusion	111
4	Rendezvous	112
4.1	Introduction	112
4.1.1	Preliminaries and Notation	113
4.1.2	Rendezvous Problem	113
4.1.3	Robot Model for Rendezvous	114
4.1.4	Outline and Results	114
4.2	Single Random Bit	115
4.3	$k + 1$ Random Bits	124
4.4	Rendezvous and Knowledge	133
4.4.1	Unknown u and L	134
4.4.2	Known u and L	134
4.4.3	Unknown u and Known L	138
4.4.4	Known u and Unknown L	140
4.5	Conclusion	141

5	Evacuation	143
5.1	Introduction	143
5.1.1	Preliminaries	144
5.1.2	Evacuation Problems	145
5.1.3	Robot Model for Evacuation	145
5.1.4	Outline and Results	147
5.2	Evacuation with a Single Exit	148
5.3	Evacuation with k Exits	154
5.4	k -exit Evacuation with Arbitrary D and L	157
5.5	Conclusion	166
6	Conclusion and Future Work	167
6.1	Discussion	167
6.2	Open Problems	172
6.3	Future Work	172
	 Abbreviations	 176
	 Nomenclature	 177
	 Bibliography	 178
	 Index	 187

List of Tables

2.1	Competitive ratio of <code>LeapFrog</code> for collections of robots of size 2,3,4. .	57
4.1	Summary of performance of randomized rendezvous algorithms for two robots in a ring.	115
5.1	Costs of <code>Evacuate_L</code> (D) for the Evacuation Problem with Fixed Initial Distance.	146
5.2	Costs of <code>Evacuate</code> (D) for the Evacuation Problem with Chosen Initial Distance.	146

List of Figures

2.1	Reducing the time of the first searcher as described in Lemma 2.4. . . .	33
2.2	Reducing the time of subsequent searchers as described in Lemma 2.4. .	34
2.3	Search configurations described in Lemma 2.5	36
2.4	Search configurations described in Lemma 2.6.	37
2.5	Schedules produced by (a) Comb and (b) LeapFrog.	49
2.6	Competitive ratio of LeapFrog as a function of the number of robots .	60
2.7	Effect of swapping a and b on outer intervals	70
3.1	Two robots r_1, r_2 patrolling in opposite directions.	91
3.2	Two robots r_1, r_2 perpetually alternate between $x(0)$ and $x(1)$	95
3.3	Two time consecutive bounces between robots r_i, r_{i+1}	98
4.1	Initial configurations A_a and B_a for distance $a \leq L$ between robots . . .	117
5.1	Example of an evacuation problem with 4 exits.	144
5.2	Single exit evacuation of colocated robots.	149
5.3	Two robots starting at distance L explore the circle up to time $t = \frac{1}{4} + \frac{L}{2}$. .	152
5.4	k -exit evacuation of colocated robots.	154
5.5	Transforming a 4-exit evacuation with equally spaced exits to a single exit problem.	157
5.6	k -exit evacuation for $\text{Evacuate}_L(D)$	159
5.7	Performance of $\text{Evacuate}(D)$ for $k = 2$ exits.	163

List of Algorithms

2.1	Comb	41
2.2	SwarmSpeed	47
2.3	LeapFrog	47
2.4	Swap-and-Expand	66
2.5	GreedyPartition	69
2.6	ObliviousRandomized	73
3.1	CentralizedPatrolling	86
3.2	OnlinePatrolling	92
3.3	DistributedPatrolling	96
3.4	DecideConverge	103
4.1	Rendezvous ₀	116
4.2	Rendezvous ₀ (t)	118
4.3	Rendezvous ₀ ($\vec{t} = (t_1, t_2, \dots, t_{k-1})$)	123
4.4	Rendezvous ₁ (k, t)	125
4.5	Rendezvous ₂ (t)	134
4.6	Rendezvous ₃	138
4.7	Rendezvous ₄ (k)	138
4.8	Rendezvous ₅	141
5.1	Evacuate ₀	149
5.2	Evacuate _{L}	151
5.3	Evacuate ₀ (D)	155
5.4	Evacuate _{L} (D)	160
5.5	Evacuate(D)	162

Chapter 1

Introduction

The term mobile agents encompasses many different implementations and areas of study: Software agents exist that can move from node to node in a network; the emergent intelligence of swarms of low-powered devices is studied in relation to artificial intelligence; mobile sensor networks allow for sensors to deploy in an intelligent manner to improve performance.

We approach our investigation of mobile agents from the perspective of distributed computing, and so we will adopt our terminology and models from this field. Mobile agent computing has become a significant area of study in the field of distributed computing. Common tasks and problems solved by mobile robots include: *gathering*, *scattering*, *pattern formation*, and *flocking* (see [Flocchini et al. \[2012\]](#)). In this thesis, we focus on another common task for mobile robots: *searching*.

There are two models for the universe in which mobile agent protocols operate: *discrete* and *continuous*. In a discrete model, the agents operate on a *graph*. This is useful when considering, for instance, mobile software agents which operate on a communication network. In the *continuous* model, which we adopt in this thesis, the agents operate in a

metric space (usually a connected subset of $\mathbb{R}^k$). In this model, it is most convenient to consider the agents as mobile robots moving through a geometric domain. Henceforth, we will refer to autonomous mobile agents as simply *robots*.

Frequently it is assumed that all robots are identical in their capabilities. Furthermore, it is often the case that these robots are *anonymous*, which leads to many interesting problems on how to break the symmetry in this uniform set of robots. Breaking symmetry is not the focus of this thesis, and therefore the assumption of anonymous robots is not made in this thesis. In fact most of the problems we examine specifically study how differences in capabilities of robots can be leveraged to obtain efficient solutions to search problems.

This approach is not new. Recent attention has been given to fleets of robots with differing capabilities – specifically the ability to move with different speeds. However, even in these studies, each robot is limited to a single speed. We wish introduce and motivate a study where each robot has a unique *pair* of speeds. We refer to these robots as *two-speed* robots.

1.1 Searching with Two-Speed Robots

A prisoner escapes from a prison, and the warden instructs the prison guards to search the surrounding area to recapture him. The guards think about the best way to do this and settle on the following approach: They determine how far the prisoner could have travelled, move to and establish a perimeter at that distance, and begin searching back towards the prison. This is an intuitive search strategy, however, it relies on the assumption that the guards are able to move to the perimeter more quickly than they would be able to conduct a search of the area up to the perimeter.

This seems like an entirely reasonable and natural assumption to make. Travelling by car to the perimeter is much faster than taking the time to carefully search through the terrain, buildings, etc., where the prisoner could be potentially be hiding. Conversely, there is no way to thoroughly conduct a search of the area while travelling at high speed in a car.

There are many other situations where *two-speed* searching is a sensible way to approach exploration of various domains. For example *foraging* or *harvesting* a field may take longer than walking across. Consider a more computer science related scenario, where we want to search the nodes of a linked list for a specific data pattern. Obviously a simple traversal of the list will be much faster than examining each node before moving to the next. In general, searching in data structures would be suitable to a two-speed model. Other computer science problems such as *indexing*, *code inspection*, and *packet sniffing* also require a more involved inspection process. Similar problems arise in many other domains.

Consider now a more theoretical search problem. We want to coordinate a group of n robots are situated at the beginning of a unit interval to search the unit interval. Each robot r_i has a unique speed s_i with which it can move. We assume that the robots are able to search while moving at full speed. The solution to this problem is trivial: The robot with the greatest speed will search the entire interval, while the remaining robots do nothing. It does not matter how many robots there are, the solution is entirely dependant on maximum speed of the robots.

Does it make sense that no other robot except the fastest is able to contribute anything to the search? If the robots are truly able to search at full speed then, yes, this makes sense. However, as we argue above, there are many instances we can imagine where the act of searching will slow down the robots – even if this slow-down is small. We therefore arrive at our proposed model of two-speed robots.

1.2 Two-Speed Robot Model

Each problem addressed in this thesis will be executed on a continuous domain by a swarm of n two-speed autonomous mobile robots $r_1, r_2, \dots, r_n$. We may alternately refer to the swarm of robots as a collection, group, fleet, etc., all with the same implied meaning.

To make our results as general as possible, we assume a weak model for the robots. We add capabilities only as needed for a specific problem, or in order to examine how these added capabilities improve performance. Therefore, unless specified otherwise, each robot has the following properties:

autonomous

Robots do not receive any instructions from an outside entity.

deterministic

Robots will change states based solely on stimuli from their environment.

mobile

Each robot has a *walking speed* w_i and a *searching speed* s_i , where $s_i < w_i$.

oblivious

Robots have no ability to remember any information about their previous states, or their environment.

silent

Robots do not have the ability to communicate with each other.

synchronized

We assume that the robots all have chronometers and begin executing the algorithm simultaneously. Therefore, the robots are *fully synchronous*.

two-state

At any moment, a robot will either be *walking* through the domain with a speed not exceeding its walking speed w_i , or *searching* the domain with speed not exceeding s_i .

1.3 Problem Domains

For the purpose of this thesis, we will restrict ourselves to one dimensional domains. Particularly, we will examine the problems either on an *open fence* or a *closed fence*, which are defined as follows:

open fence

An open fence is any Jordan arc, i.e., a curve that does not intersect itself and is not missing any points. It is defined formally as follows:

Let $(x_1, y_1), (x_2, y_2) \in \mathbb{R}^2$. Let $\mathcal{P} : [0, 1] \rightarrow \mathbb{R}^2$ be a path from (x_1, y_1) to (x_2, y_2) . $\mathcal{P}$ is a Jordan arc (and therefore an open fence) if and only if $\mathcal{P}$ is an injection.

closed fence

A closed fence is any Jordan curve, i.e., any *closed* curve that does not intersect itself and is not missing any points. It is defined formally as follows:

Let $(x_1, y_1), (x_2, y_2) \in \mathbb{R}^2$. Let $\mathcal{P} : [0, 1] \rightarrow \mathbb{R}^2$ be a path from (x_1, y_1) to (x_2, y_2) . $\mathcal{P}$ is a Jordan curve (and therefore a closed fence) if and only if $(x_1, y_1) = (x_2, y_2)$, and $\mathcal{P} \setminus \{(x_2, y_2)\}$ is an injection.

In other words, $\mathcal{P}$ is a Jordan curve if and only if $\mathcal{P}$ is a homeomorphic image of the unit circle.

We note that the relationship between open fences and the unit interval $[0, 1]$, and the relationship between closed fences and the unit circle means that any results we obtain for each of these domains can easily be extended to their respective families of fences.

Accordingly, throughout this thesis we may refer to an open fence as an interval or (line) segment where convenient. Similarly, we may refer to a closed fence as a circle. In all cases where this may occur it is assumed that we are talking about a corresponding open or closed fence.

1.4 Problems Studied and Summary of Results

As mentioned previously, the problems we study are in one-dimensional domains. We will refer to the analogy of prison guards and prisoners to motivate the various problems we search. We will give a brief overview of the problems addressed and results obtained in the thesis. More detailed and formal problem statements and results may be found in the relevant chapters.

Searching with Two-Speed Robots

Once the prison guards have apprehended the escaped prisoners, they determine that the prisoners escaped through a hole in the fence. The guards do not know where the hole is located (or if there is possibly more than one hole), and so they need to conduct a thorough examination of the fence to find the locations of all potential holes. This would be an application of the search problem studied in this thesis.

In Chapter 2 we study problems relating to searching an open fence of length L using a new *two-speed* model of autonomous mobile robots. We use the analogy of the robots

as *beachcombers* to emphasize that when searching a domain (e.g. a beach looking for things of value), robots would need to move slower than if they were simply traversing the domain.

We study two variants of the problem where the robots all begin at one endpoint of the fence: when the robots know the length of the fence, which we refer to as the Beachcombers' Problem; and when the robots do not know the length of the fence, which we refer to as the Online Beachcombers' Problem.

We also study the problem when the robots know the length of the fence, are allowed to divide themselves into a number of groups, and where each group may freely choose its starting location. We refer to this as the Multi-Source Beachcombers' Problem.

For all the problems, we want to develop efficient *mobility schedules* (algorithms) for the robots which complete the search of the segment as fast as possible. More exactly, we want to maximize the *speed* of the mobility schedule – defined as the ratio of the segment length compared to the time of completion of the schedule. A search of the interval $[0, L]$ is completed at the time when each of its points have been searched by at least one of the n robots.

For the Beachcombers' Problem, we give an algorithm producing a mobility schedule for any swarm of two-speed robots and prove its optimality.

We then we propose an algorithm to solve the Online Beachcombers' Problem. We prove that the proposed online algorithm is 2-competitive. The competitive ratio is shown to be better in the case when the robots' walking speeds are all the same, approaching 1.29843 as n goes to infinity.

Finally, for the Multi-Source Beachcombers' Problem, we show that even with only two sources, some swarms create instances which are NP-hard, however, other instances are

solvable in polynomial time. We then turn our attention to the problem with any number of sources, where we propose and analyse a very efficient and simple randomized algorithm with constant approximation.

Patrolling with Two-Speed Robots

To ensure no future breakouts, the warden asks the prison guards to patrol the perimeter fence of the prison. It is known that it will take any escaping prisoners at least some amount of time to create a hole in the fence (or climb the fence), therefore each point in the fence can be left unpatrolled for a small amount of time (less than what is needed to escape), but needs to be regularly checked.

Chapter 3 is devoted to studying the problem of patrolling with two-speed robots. As patrolling is essentially the repeated search of a domain, this is a natural problem to examine in a two-speed model. Consider patrols of security guards searching for intruders, where ensuring a search area contains no intruders will take more time than uninvolved territory traversal.

The problem is such: an open fence, represented by a unit interval is to be patrolled collectively by n robots. A robot patrols a portion of the domain by searching it. Each robot is allowed to patrol only in one of the two directions (not necessarily the same for all robots). We want to schedule the perpetual movements of the robots to minimize the *idle time*, defined as the maximal time interval any point is not visited by some patrolling robot.

First, we give a centralized algorithm constructing schedules with optimal idleness, and subsequently we show a nice application to a transportation problem concerning *Scheduling with Regular Delivery*.

The main contribution of the chapter is the study of distributed, dynamical schedules for patrolling robots with only primitive capabilities. Surprisingly we are able to design a dynamic schedule for very weak collections of two robots (silent, oblivious, passively mobile), achieving the optimal idleness. Our algorithm defines a dynamical system of memoryless robots moving back and forth in an interval. In general, analysis of the system dynamics is very complex.

Part of our contribution is a very technical analysis of the dynamics of special families of dynamical systems of n robots that we call *regular*. For such systems we also, i.e. such that two robots may meet only when travelling in opposite directions, propose a highly non-trivial $O(n^2)$ algorithm to decide whether or not robots converge to a stable configuration thus verifying if the dynamic schedule is optimal. It turns out that a very natural family of dynamical systems that we call *monotone* systems such that $w_n > w_{n-1} > \dots > w_1 > p_1 > p_2 > \dots > p_n$, $p_i < p_j \implies w_i > w_j$, where p_i, w_i are the patrolling and walking speeds of the i th robot, respectively, can be shown to be regular. Further, for $n \leq 4$, such monotone dynamical systems are shown to converge to stable configurations.

Our approach also allows us to solve the following transportation problem concerning *Scheduling with Regular Delivery*. Suppose that a commodity produced at one endpoint of a path is being delivered to the other endpoint by a set of robots. Suppose also, that each robot may transport at most one item of the commodity and its transportation speed is lower than its speed when travelling with no charge. What is the smallest value I such that a new item of the commodity is delivered regularly at time intervals I ?

Randomized Rendezvous with One-Speed Robots

While attempting another break out, the prisoners might free themselves from their different cells, make their way to the fence and need to meet in order to escape. This would be an application of rendezvous.

We investigate randomized rendezvous problems in Chapter 4. Rendezvous is a special case (with only two robots) of the more general *gathering* problem: the goal is for all the robots to meet at a common point. One can think of rendezvous as a search problem where the robots are searching for each other. Since rendezvous where the robots have different speeds is not well studied, we do not apply the two-speed robot model, instead we investigate a model where the robots each have a single speed. The robots have different speeds $u, 1$, where $u > 1$, but are unaware of their own speed.

We provide randomized rendezvous algorithms for two synchronous robots located on a closed fence of length L .

The first algorithm, Rendezvous_0 , works with one random bit per robot and consists of a single round: after choosing their initial directions the robots never change direction. Rendezvous is established in $\frac{u \cdot L}{2(u^2 - 1)}$ expected time and this is shown to be optimal among all randomized algorithms employing a single random bit during their execution.

The second algorithm, $\text{Rendezvous}_1(k)$, has the two robots alternate for $k + 1$ rounds between choosing the direction at random followed by walking for a predetermined time $\frac{L}{u+1}$; in the last step the robots walk until rendezvous. Among all algorithms that use $k + 1$ random bits we establish a sharp threshold; for $u \leq 2$, $\text{Rendezvous}_1(k)$ is optimal in terms of expected rendezvous time while for $u > 2$, Rendezvous_0 is optimal.

Further, we provide new randomized rendezvous algorithms employing more random bits and analyse their expected rendezvous time depending on the knowledge of the robots about the length L of the ring and their speeds ($u > 1$).

Evacuation with Identical Robots

Finally, the prisoners might come up with a plan to secretly dig one or more hidden tunnels with openings at the perimeter of the fence. However, during the chaos of a breakout they get disoriented while getting to the fence. They need to all find a hole to escape, and need to do it fast, before the guards notice they're missing. This would be an application of the evacuation problem, which we consider in Chapter 5.

As is the case with the rendezvous problem, since the evacuation problem is not well studied, we do not consider the two-speed robot model. In fact, for this problem, we assume the robots have identical speeds.

The evacuation problem we consider is the following. Distributed on a closed fence are k exits. Two autonomous mobile robots with unit speed and wireless communication, are placed on the fence. The robots have a map of the domain, including exits, (i.e. the robots know the relative position of the exits), but do not have knowledge of their own initial locations on the domain, rather they only know their relative distance.

The goal of the *evacuation problem* is to give an algorithm for the robots which minimizes the time required for both robots to reach an exit, in the worst case. We consider two variations of the problem depending on whether the two robots have control over their initial distance.

When the initial distance of the robots is part of the input (i.e. no control), we show that simple algorithms exist which achieve optimal worst case evacuation times for the

cases where: the robots begin colocated with an arbitrary distribution of the exits; and when the exits are either colocated or evenly spaced, with arbitrary starting positions of the robots.

We also give upper and lower bounds on the problem with arbitrary exit distribution and starting positions of the robots. For the problem where robots can choose their initial distance (with knowledge of, but not control over the distribution of exits), we propose a natural family of algorithms parametrized by the maximum distance between any two exits.

1.5 Related Work

Since searching is a primitive form of all the problems studied, we will approach the related work by first examining the work leading up to and surrounding the two-speed search results presented in the thesis. We will then examine the more specific work related to each of the remaining problems. We note that while some of the problems have been studied in the context of robots with different speed, the study in this thesis is the first one assuming two-speed robots for the problem of searching or patrolling.

Searching

The study of graph searching started with the work of [Koopman \[1946\]](#), who defined the goal of a searcher as minimizing the time required to find a target object. The efficiency of a searching or exploration algorithm is usually measured by the time used by the mobile agent, often proportional to the distance travelled. The work of [Stone \[1975\]](#) focuses on the problem of optimizing the allocation of effort (by the searcher) required to search for a target. The author takes a Bayesian approach, assuming there is a prior

distribution for the target location (known to the searcher) as well as a function relating the conditional probability of detecting a target given it is located at a point to the effort applied.

Many papers followed studying searching and exploration of graphs (e.g. [Deng and Papadimitriou \[1990\]](#), [Fomin and Thilikos \[2008\]](#)) or geometric environments, (e.g. [Albers and Henzinger \[2000\]](#), [Alpern and Gal \[2002\]](#), [Baeza-Yates et al. \[1993\]](#), [Czyzowicz et al. \[2013a\]](#), [Deng et al. \[1991\]](#)). The purpose of these studies was usually either to learn (map) an unknown environment (e.g. [Deng and Papadimitriou \[1990\]](#)) or to search it, looking for a target (motionless or mobile) (cf. [Fomin and Thilikos \[2008\]](#)).

Searching problems have been studied from a game-theoretic viewpoint. [Alpern and Gal \[2002\]](#) presented an approach to rendezvous and searching, when two mobile players either collaborate in order to find each other, or they compete against each other - one willing to meet and the second one attempting to avoid the meeting. Searching one-dimensional environments (segments, lines, semi-lines), similar to the results in this chapter, despite the simplicity of the environment, often led to interesting results in this model (cf. [Beck \[1964\]](#), [Bellman \[1963a\]](#), [Demaine et al. \[2006\]](#)).

Many searching and especially exploration algorithms are *online*, i.e. they concern a priori unknown environments, cf. [Albers \[2003\]](#), [Albers and Schmelzer \[2011\]](#). Performance of such algorithms is usually expressed by the *competitive ratio*, i.e. the proportion of the time spent by the online algorithm versus the time of the optimal offline algorithm, which assumes the knowledge of the environment (cf. [Berman \[1998\]](#), [Fleischer et al. \[2008\]](#)). Most exploration algorithms (e.g. [Czyzowicz et al. \[2013a\]](#), [Deng et al. \[1991\]](#), [Dereniowski et al. \[2013\]](#) and several search algorithms (cf. [Albers \[2003\]](#), [Albers and Schmelzer \[2011\]](#), [Demaine et al. \[2006\]](#)) use the competitive ratio to measure their performance.

Most of the above mentioned work concerned single robots. The problem where autonomous, co-operative mobile agents are *searching* for a particular node or item in the network originated in the work of Shannon [1951] and was studied in many other works, e.g. Chalopin et al. [2010], Das et al. [2007], Fraigniaud et al. [2006], Higashikawa et al. [2012]. Most recently Dereniowski et al. [2013] studied tradeoffs between the number of robots and the time of exploration showing how a polynomial number of robots may search the graph optimally.

Some research studying mobile robots assumes distinct robot speeds. Varying mobile sensor speed was used in Wang et al. [2011] for the purpose of sensor energy efficiency. Beauquier et al. [2010] was utilizing distinct robot speeds to design fast converging protocols, e.g. for gathering.

The Ants Nearby Treasure Search (ANTS) problem Feinerman et al. [2012], though different, is somewhat related to our study. In this problem, k identical mobile robots, all beginning at a start location, are collectively searching for a treasure in the two-dimensional plane. The treasure is placed at a target location by an adversary, and the goal is to find it as fast as possible (as a function of both k and D , where D is the distance between the start location and the target). This is a generalization of the search model proposed in Baeza-Yates et al. [1993], in which the cost of the search is proportional to the distance of the next probe position (relative to the current position) and the goal is to minimize this cost. Related is the w -lane cow-path problem (see Kao and Littman [1997], Kao et al. [1993]), in which a cow is standing at a crossroads with w paths leading off into unknown territory. There is a grazing field on one of the paths, while the other paths go on forever. Further, the cow won't know the field has been found until she is standing in it. The goal is to find the field while travelling the least distance possible. Layered graph traversal, as investigated by Fiat et al. [1991], Papadimitriou and Yannakakis [1991], is similar to the cow-path problem, however

it allows for short-cuts between paths without going through the origin. Research in [Angelopoulos et al. \[2011\]](#) is concerned with exploring m concurrent semi-lines (rays) using a single searcher, where a potential target may be located in each semi-line. The goal is to design efficient search strategies for locating t targets (with $t \leq m$). Another model studied in [Beauquier et al. \[2010\]](#) introduces a notion of speed of the agents to study the gathering problem, in which there is an unknown number of anonymous agents that have values they should deliver to a base station (without replications).

When we allow the robots to begin searching from multiple chosen locations, such as we do for the Multi-source Beachcombers' Problem, we are looking at a combination of two tasks: *scheduling* and *partitioning*. Scheduling jobs with non-identical capacity requirements or sizes, on single batch processing, to minimize total completion time and makespan, as well as variants of this problem, are studied in several papers including [Brucker et al. \[1998\]](#), [Potts and Kovalyov \[2000\]](#), [Uzsoy \[1994\]](#) and the survey paper [Allahverdi et al. \[2008\]](#). However, they all differ from our investigations in that they do not consider the interplay and trade-offs between walking and searching. It is the partitioning aspect that seems to account for the hardness of t -SBP. This aspect of t -SBP can be reduced to the problem of grouping n items into m subsets $S_1, \dots, S_m$ to minimize an objective function $g(S_1, \dots, S_m)$. This is a well-studied problem (c.f. [Anily and Federgruen \[1991\]](#), [Chakravarty et al. \[1985a,b\]](#)), with applications in operations research for inventory control.

From an algorithmic perspective, our work is closely related to Set Partition Problems with Additive Objective (SPAO), see also Section 2.4.1. A special case of these problems are the so-called Combinatorial Allocation Problems (CAP), where one is given a function $g : 2^n \mapsto \mathbb{R}_+$ and a fixed integer t , and the objective is to maximize $\sum_{i=1}^t g(A_i)$ over all partitions $A_1, \dots, A_t$ of n items. We will later prove that t -SBP is a SPAO and

in fact a CAP where the function g is sub-modular. A $1 - (1 - 1/t)^t$ approximation algorithm is already known [Dobzinski and Schapira \[2006\]](#) for CAPs (matching our performance of Theorem 2.33). More recently, [Feige and Vondrák \[2010\]](#) showed that the same problem is APX-hard, and they improved upon [Dobzinski and Schapira \[2006\]](#) (for large enough values of t) by presenting a $1 - 1/e + \epsilon$ approximation algorithm for $\epsilon \approx 10^{-5}$. Both these are randomized linear programming (LP)-based algorithms, which utilize the solution of an LP with $t \cdot n$ many variables in order to allocate at random the n robots to the t locations (using the language of our problem). As a result, the running time of the previous algorithms is dominated by the running time to solve this LP. In contrast, we propose a much simpler (and oblivious) randomized allocation rule that can be implemented in linear time, and that achieves a $1 - (1 - 1/t)^t$ approximation in the general case. To achieve this, we heavily rely on the special structure of our function g , which also allows us to establish improved approximations, or even exact solutions, for more restricted yet interesting variations of our problem.

Patrolling

Patrolling is the act of surveillance, consisting of walking perpetually around an area in order to protect or supervise it, and has been studied intensively in robotics [Almeida et al. \[2004\]](#), [Chevaleyre \[2004\]](#), [Elmaliach et al. \[2008, 2009\]](#), [Hazon and Kaminka \[2008\]](#), [Machado et al. \[2002\]](#), [Yanovski et al. \[2003\]](#) where it is often viewed as a version of terrain *coverage*, a central task in robotics. It is useful in ecological monitoring, detecting intrusion, monitoring and locating objects or humans (that may need to be rescued from a disaster), detecting network failures or even discovering web pages which need to be indexed by search engines [Machado et al. \[2002\]](#). Boundary and area patrolling have been studied in [Agmon et al. \[2008\]](#), [Elmaliach et al. \[2008, 2009\]](#),

[Pasqualetti et al. \[2010\]](#) with approaches placing more emphasis on experimental results.

Idleness is the accepted measure of algorithmic efficiency of patrolling and is related to the frequency with which the points of the environment are visited [Almeida et al. \[2004\]](#), [Alpern et al. \[2011\]](#), [Chevaleyre \[2004\]](#), [Elmaliach et al. \[2008, 2009\]](#), [Machado et al. \[2002\]](#)) (this last criterion was first introduced in [Machado et al. \[2002\]](#)). Diverse approaches to patrolling based on idleness criteria are discussed in [Almeida et al. \[2004\]](#). Patrolling as a game between patrollers and intruder is studied in [Alpern et al. \[2009, 2011\]](#), [Amigoni et al. \[2010\]](#). Elsewhere patrolling is studied based on swarm or ant-based algorithms [Elor and Bruckstein \[2010\]](#), [Marino et al. \[2009\]](#), [Yanovski et al. \[2003\]](#). Robots are memoryless (or having small memory), decentralized [Marino et al. \[2009\]](#) with no explicit communication permitted either with other robots or the central station, and may have local sensing capabilities [Elor and Bruckstein \[2010\]](#). Ant-like algorithms usually mark the visited nodes of the graph and [Yanovski et al. \[2003\]](#) presents an evolutionary process. This thesis shows that a team of memoryless robots, by leaving marks at the nodes while walking through them, after relatively short time stabilizes to the patrolling scheme in which the frequency of the traversed edges is uniform to a factor of two (i.e., the number of traversals of the most often visited edge is at most twice the number of traversal of the least visited one).

Theoretical graph-based approaches to patrolling can be found in [Chevaleyre \[2004\]](#). In [Pasqualetti et al. \[2010\]](#), polynomial-time patrolling solutions for lines and trees are proposed. For the case of cyclic graphs, [Pasqualetti et al. \[2010\]](#) proves the NP-hardness of the problem and a constant-factor approximation is proposed.

Optimal patrolling with same-speed robots in mixed domains, where regions to be traversed are fragmented by components that do not need to be monitored, is studied in [Collins et al. \[2013\]](#). Patrolling with robots that do not necessarily have identical speeds

offers several surprises both in terms of the difficulty of the problem as well as in terms of the algorithmic results obtained. Such a study has been initiated in [Czyzowicz et al. \[2011\]](#) and investigated further in [Dumitrescu et al. \[2014\]](#), [Kawamura and Kobayashi \[2012\]](#). The partition strategy, where each robot patrols and walks along a separate area, has been proven to work for two robots in [Czyzowicz et al. \[2011\]](#), and for three in [Kawamura and Kobayashi \[2012\]](#).

Standard capabilities of mobile robots usually include communication, computation, and environment perception. For many reasons (e.g. production cost, limited or specific applications) one may wish to deal with robots of reduced ability, especially if they are needed in large numbers. In such cases, feasibility issues, rather than computation efficiency are sought [Angluin et al. \[2006, 2007\]](#), [Beauquier et al. \[2010\]](#), [Cieliebak et al. \[2012\]](#). [Angluin et al. \[2006\]](#) introduced *population protocols* (see also [Angluin et al. \[2007\]](#), [Beauquier et al. \[2010\]](#)), where robots are subject to *passive mobility*, also used in this thesis. Passive mobility aims to model volatile environments like water flow, wind or unstable mobility of agents' carriers. Further, [Beauquier et al. \[2010\]](#) considered different speed of such agents. (cf. also [Czyzowicz et al. \[2013c\]](#)).

Rendezvous

Rendezvous is an important primitive in robotics and communications whenever meeting of different processes is required so as to exchange information. The problem of rendezvous has been studied extensively in many topologies (or domains) [Bampas et al. \[2010\]](#), [Chalopin et al. \[2007\]](#), [Czyzowicz et al. \[2013b\]](#) and under various assumptions on system synchronicity and movement of the agents [De Marco et al. \[2006\]](#), [Dessmark et al. \[2003\]](#) as a dynamic symmetry breaking problem [Yu and Yung \[1996\]](#) as well as

in operations research [Alpern \[1995\]](#) in order to understand the limitations of search theory.

It is known that knowledge of potential “asymmetric” characteristics of the underlying domain (e.g., node degrees, edge weights, etc) over which a rendezvous algorithm is performed can often be exploited to simplify solutions and reduce the resulting rendezvous time. (For example, the agents can rendezvous on a node of a given degree, if this value is unique.) Because of its nature as a symmetry breaking problem, rendezvous under its various “model instantiations” becomes harder and more interesting the more symmetric the graph is. In particular, this makes its analysis on the ring important for understanding the complexity of the rendezvous problem. This is one of the main reasons that in this thesis, we will restrict our study to rendezvous on the ring.

There are several papers in the scientific literature dealing with the rendezvous problem for two or more robots in a ring. For a monograph study on rendezvous in the ring and related work see [Kranakis et al. \[2010\]](#) where a variety of models and rendezvous algorithms are discussed. Algorithms are studied taking into account various assumptions and parameters including: rendezvous time (see [Flocchini et al. \[2004\]](#), [Kranakis et al. \[2003\]](#)) memory used (see [Kranakis et al. \[2008\]](#)), detecting the feasibility of rendezvous (see [Czyzowicz et al. \[2008\]](#)), game theoretic tradeoffs (see [Alpern and Gal \[2002\]](#)), tokens (see [Kranakis et al. \[2006\]](#)), randomization (see [Métivier et al. \[2003\]](#)), etc. Specifically, for robots with uniform speeds, randomized rendezvous on the ring has previously been studied in [Alpern \[2000\]](#), [Howard \[1999\]](#).

In this thesis we are concerned with a different model of rendezvous in a ring for two mobile robots first introduced in a deterministic setting by [Feinerman et al. \[2014\]](#). Consider a bi-directional ring with perimeter of length L (L is a real number) and two synchronous robots which can move with constant respective speeds $u, 1$, where $u > 1$. We assume a continuous model: robots have arbitrary starting positions and can

rendezvous at any point in the ring if they are found at the same point at the same time; a rendezvous can happen because either the two robots are moving in opposite directions or in the same direction and one of the robots (faster) overtakes the other (slower).

While some models allow the robots to communicate indirectly by leaving markings or tokens on the ring (e.g. [Bender et al. \[2002\]](#), [Blum and Kozen \[1978\]](#)), our model does not allow any communication or tokens whatsoever between robots until rendezvous has been achieved.

There has been recent interest in the study of mobile agent problems such as searching [Czyzowicz et al. \[2014b\]](#), patrolling [Czyzowicz et al. \[2011\]](#), [Kawamura and Kobayashi \[2012\]](#) and gathering [Beauquier et al. \[2010\]](#) where the agents have different speeds. However, little research has been carried out for rendezvous with different speeds. The two speed deterministic model for rendezvous in a ring was first considered in [Feinerman et al. \[2014\]](#). The authors show that $\frac{u \cdot L}{u^2 - 1}$ is a tight deterministic upper and lower bound for rendezvous between two robots in this setting. Their basic deterministic algorithm is as follows: 1) the robots move in their chosen direction and according to their given speed until they have covered a distance $\frac{u \cdot L}{u^2 - 1}$ from their starting positions (note the robots reach this distance at different times depending on their speeds); 2) if they did not rendezvous before or when reaching this distance, they reverse direction and move until they rendezvous. It is clear that this algorithm requires that the robots have knowledge of the quantity $\frac{u \cdot L}{u^2 - 1}$.

The algorithms in [Feinerman et al. \[2014\]](#) differ from our algorithms in two respects. First, their algorithms are deterministic, while ours are randomized (as such our algorithm has faster rendezvous time in expectation). Second, their robots are equipped with *pedometers* for measuring distance, while our robots are equipped with *chronometers* for measuring time.

Evacuation

When the map of their environment is unknown to the robots at the beginning of the search, a common problem is that of *exploring* the environment [Albers and Henzinger \[2000\]](#), [Albers et al. \[2002\]](#), [Deng et al. \[1991\]](#), [Hoffmann et al. \[2001\]](#). However, it is common that the goal of the robots is different, and exploration is only necessary insofar as it facilitates the achievement of that goal [Kleinberg \[1994\]](#), [Papadimitriou and Yannakakis \[1991\]](#).

When a map of the environment is known to the robots, problems often involve searching for a target located somewhere in the environment. That target might be stationary, as in the cow path problem [Beck \[1964\]](#), [Bellman \[1963b\]](#) or the plane searching problem [Baeza-Yates and Schott \[1995\]](#), [Baeza Yates et al. \[1993\]](#). The target might also be mobile, where the problems are often presented as search games, commonly called cops and robbers or pursuit-evasion games [Bonato and Nowakowski \[2011\]](#), [Chung et al. \[2011\]](#), [Fomin and Thilikos \[2008\]](#), [Nahin \[2012\]](#).

Problems relating to evacuation have been studied in a centralized setting, with the goal of creating evacuation plans [Baumann and Skutella \[2009\]](#), [Fekete et al. \[2010\]](#). Evacuation in a distributed setting, as is the case in this thesis, has been studied on a disc in [Czyzowicz et al. \[2014a\]](#) (and the follow-up work [Czyzowicz et al. \[2015b\]](#)), which is similar to the *lost at sea* problem [Gluss \[1961\]](#); and on a line in [Chrobak et al. \[2015\]](#). In [Czyzowicz et al. \[2014a\]](#) the authors introduced and studied the evacuation problem of k robots through one exit located on the perimeter of a disk when the robots have wireless communication, and when they do not. The major difference between this work and ours is that we only allow robots to move on the perimeter of the disk, whereas robots in [Czyzowicz et al. \[2014a, 2015b\]](#) can move anywhere on the perimeter or in the interior of the disk. As a result, our model allows us to distinguish the cases when the

robots do not have a free choice of their initial arc-distance on the perimeter of the disk (which we denote by L), as well as when they do. Moreover, under this topology, the only communication model we are motivated to consider is wireless, since if restricted to face-to-face communication our problem trivializes. Additionally, where [Czyzowicz et al. \[2014a\]](#) studies the problem for k robots and one exit, we consider the problem with two robots and k exits.

1.6 Publications

Publications upon which the thesis is based

Refereed Conference Papers

- E. Kranakis, D. Krizanc, **F. MacQuarrie**, and S. Shende. Randomized rendezvous algorithms for agents on a ring with different speeds. In *Proceedings of the 16th International Conference on Distributed Computing and Networking (ICDCN 2015)*, pages 9:1 – 9:10. ACM, 2015. **Awarded best paper.**
- J. Czyzowicz, L. Gąsieniec, K. Georgiou, E. Kranakis, and **F. MacQuarrie**. The beachcombers' problem: Walking and searching with mobile robots. In *Proceedings of the 21st International Colloquium on Structural Information and Communication Complexity (SIROCCO 2014)*, pages 23–36. Springer International Publishing, 2014. **Awarded best student paper.** Also an invited submission to a special issue of *Theoretical Computer Science*.
- J. Czyzowicz, L. Gąsieniec, K. Georgiou, E. Kranakis, and **F. MacQuarrie**. The multi-source beachcombers' problem. In *Proceedings of the 10th International*

Symposium on Algorithms and Experiments for Sensor Systems (ALGOSENSORS 2014), pages 3–21. Springer Berlin Heidelberg, 2014.

Submitted Papers

- J. Czyzowicz, K. Georgiou, E. Kranakis, **F. MacQuarrie**, and D. Pajak. Fence patrolling with two-speed robots.
- J. Czyzowicz, S. Dobrev, K. Georgiou, E. Kranakis, and **F. MacQuarrie**. Evacuating two robots from multiple unknown exits in a circle.

Other publications by the author

Journal Papers

- E. Kranakis, **F. MacQuarrie**, and O. Morales Ponce. Connectivity and stretch factor trade-offs in wireless sensor networks with directional antennae. *Theoretical Computer Science*, 590:55–72, 2015. Also appeared in proceedings of FCT 2013 and COCOA 2012.
- M. Eftekhari Hesari, E. Kranakis, O. Morales Ponce, **F. MacQuarrie**, and L. Narayanan. Strong Connectivity of Sensor Networks with Double Antennae. *Theoretical Computer Science*, in press. Also appeared in proceedings of SIROCCO 2012.

Refereed Conference Papers

- E. Kranakis, **F. MacQuarrie**, I-K. Travizani Maffra, and O. Morales Ponce. Strong Connectivity of Wireless Sensor Networks with Double Directional Antennae in 3D. In *Proceedings of 12th International Conference on Ad Hoc Networks and Wireless (ADHOC-NOW 2013)*, pages 257–268. Springer, 2013.
- E. Kranakis, **F. MacQuarrie**, O. Morales Ponce, and J. Urrutia. Uninterrupted Coverage of a Planar Region with Rotating Directional Antennae. In *Proceedings of 11th International Conference on Ad Hoc Networks and Wireless (ADHOC-NOW 2012)*, pages 56–68. Springer, 2012.

Chapter 2

Searching

2.1 Introduction

A fence is to be explored collectively by n robots. At every time moment a robot may perform either of the two different activities of *walking* and *searching*. While walking, each robot may *traverse* the domain with a speed not exceeding its maximal *walking speed*. While searching, the robot performs a more elaborate task on the domain and may not exceed its maximal *searching speed*. The bounds on the walking and searching speeds may be distinct for different robots, but we always assume that each robot can walk with greater maximal speed than it can search. We use the analogy of the robots as *beachcombers* to emphasize that when searching a domain (e.g. a beach looking for items of value), robots move slower than if they were simply traversing the domain.

In our problem, the searchers collaborate in order to terminate the searching process as quickly as possible. Our algorithms generate *mobility schedules* i.e. sequences of

¹This chapter contains results which first appeared in [Czyzowicz et al. \[2014b\]](#) and [Czyzowicz et al. \[2015a\]](#)

moves of the robots, which assure that every point of the environment is inspected by at least one robot while this robot was performing the searching activity.

2.1.1 Preliminaries and Notation

The input to our problem will always contain a swarm, R , of n robots. Each robot i has a *searching speed* s_i and a *walking speed* w_i , where $s_i < w_i$. A swarm is denoted by the tuple of speed vectors $(\mathbf{s}, \mathbf{w})$. A swarm where the walking speeds are all the same is called *W-uniform*. Similarly in an *S-uniform* swarm, all searching speeds are the same.

Robots are placed initially at a *source*, which determines the initial position and direction of movement for the robot. There can be two sources at any given point on the domain (considered in this chapter to be an interval), one for each direction of movement. This definition is the reason the robots in 1-SBP must start at an endpoint: since there is only one source, if the source is an interior point of the interval, only points that lie on the interval in the direction of movement of the robots will ever be searched. At any moment a robot r_i can be: *searching* or *walking*. When searching or walking, it is moving with speed no greater than s_i and w_i respectively. Robots can switch states instantaneously, as many times as needed, and at any time. We assume that the robots' schedules are controlled by a centralized scheduler, that the robots start at the same time, and that they can cross over each other during their operation.

Let I_L denote the interval $[0, L]$ for any positive number L . A *searching schedule* of I_L is defined by an increasing sequence of moments $t_0 = 0, t_1, \dots, t_z$, such that in each interval $[t_j, t_{j+1}]$, for $0 \leq j < z$, every robot r_i either walks along some sub-segment of I_L not exceeding its walking speed w_i , or searches some sub-segment of I_L not exceeding its searching speed s_i . The searching schedule is *correct* if for each point $p \in I_L$ there is some $j \geq 1$ and some robot r_i , such that during the interval $[t_j, t_{j+1}]$

robot r_i searches the sub-segment of I_L containing point p . We assume that there is no communication between robots during the execution of the algorithm.

A *mobility schedule*, denoted by $\mathcal{K}$, consists of (a) a partition of the swarm into t groups, (b) a choice of sources, one for each of the groups, and (c) a searching schedule which specifies at each moment the state of any robot, i.e. searching or walking. A mobility schedule is called *feasible* if there is a *finishing time* $T_{\mathcal{K}}$ after which all points of the domain have been searched by at least one robot. We note that $T_{\mathcal{K}} = t_z$ of the corresponding searching schedule. Hereafter, all schedules referred to will be assumed to be feasible mobility schedules.

The speed $S_{\mathcal{K}}(L)$ of schedule $\mathcal{K}$ searching interval I_L is defined as $S_{\mathcal{K}}(L) = L/T_{\mathcal{K}}$. The searching schedule is optimal if there does not exist any other correct searching schedule having a speed larger than $S_{\mathcal{K}}(L)$.

It is easy to see that the schedule speed maximization criterion is equivalent to its finishing time minimization when the segment length is given or to the searched segment length maximization when the time bound is set in advance. However the speed maximization criterion applies better to the online problem when the objective of the schedule is to perform searching of an unknown-length segment or a ray. Such a schedule successively searches the intervals I_L for the increasing values of L . The speed of such a schedule is defined as $S_{\mathcal{K}} = \inf_L S_{\mathcal{K}}(L)$.

Observe that any searching schedule may be converted to another one, which has the property that all sub-segments which were being searched (during some time intervals $[t_j, t_{j+1}]$ by some robots) have pairwise disjoint interiors. Indeed, if some sub-segment is being searched by two different robots (or twice by the same robot), the second searching may be replaced by the walk through it by the robot involved. Since the walking speed of any robot is always larger than its searching speed, the speed of such

converted schedule is not smaller than the original one. Therefore, when looking for the optimal searching schedule, it is sufficient to restrict consideration to schedules whose searched sub-segments may only intersect at their endpoints. All searching schedules in this chapter will have this property.

Notice as well, that, when looking for the most efficient schedule, we may restrict our consideration to schedules such that at any time moment a robot r_i is either searching using its maximal searching speed s_i , or walking with maximal allowed speed w_i . Indeed, whenever r_i searches (or walks) during a time interval $[t_j, t_{j+1}]$ using a non-maximal and not necessarily constant searching speed (respectively walking speed) we may replace it with a search (respectively walk) using the maximal allowed speed. It is easy to see that the search time of any point is never longer for the modified schedule, so the speed of such a schedule is not decreased.

We assume that all the robots start their exploration at the same time and that they are allowed to cross over each other. We study the offline and online versions of the problem with a single source, hereafter referred to the (1-Source) Beachcombers' Problem. We also study the offline version of the problem with t sources, hereafter referred to as the t -Source Beachcombers' Problem.

2.1.2 Search Problems

We consider the following search problems in this thesis:

The Beachcombers' Problem (1-SBP)

Consider an interval $I_L = [0, L]$ and n robots $r_1, r_2, \dots, r_n$, initially placed at its endpoint 0, each robot r_i having searching speed s_i and walking speed w_i , such that $s_i < w_i$.

The Beachcombers' Problem consists of finding an efficient correct schedule $\mathcal{K}$ of I_L . The speed $S_{\mathcal{K}}$ of the solution to the Beachcombers' Problem equals $S_{\mathcal{K}} = L/T_{\mathcal{K}}$, where $T_{\mathcal{K}}$ is the finishing time of $\mathcal{K}$. The robots know the value of L .

The Online Beachcombers' Problem

Placed at the origin of the interval $I_L = [0, L]$ is a group of n robots $r_1, r_2, \dots, r_n$. Each robot r_i has searching speed s_i and walking speed w_i , such that $s_i < w_i$. The Online Beachcombers' Problem consists of finding an efficient correct schedule $\mathcal{K}$ of I . The cost $S_{\mathcal{K}}$ of the solution to the Online Beachcombers' Problem, called the *speed of $\mathcal{K}$* , equals $S_{\mathcal{K}} = \inf_L S_{\mathcal{K}}(L) = \inf_L \frac{L}{T_{\mathcal{K}}(L)}$ where $L = \mu \cdot k$, for any number μ and any positive integer k , and $T_{\mathcal{K}}(L)$ denotes the time when the search of the segment I_L is completed. The robots know the value of μ , but do not have knowledge of the value of k .¹

The Multi-Source Beachcombers' Problem (t -SBP)

Consider an interval $I_L = [0, L]$ and n robots $r_1, r_2, \dots, r_n$. Each robot r_i has searching speed s_i and walking speed w_i , such that $s_i < w_i$. The t -Source Beachcombers' Problem consists of finding a correct schedule $\mathcal{K}$ of I_L which has at most t unique sources, and which maximizes the *speed* with which the robots search the interval. The speed $S_{\mathcal{K}}$ of the solution to the Beachcombers' Problem is defined to be $S_{\mathcal{K}} = L/T_{\mathcal{K}}$, where $T_{\mathcal{K}}$ is the finishing time of $\mathcal{K}$.² Whenever it is clear from the context, we may silently assume the normalization $L = 1$.

¹In our analysis, we will assume w.l.o.g. that $\mu = 1$. Note that it is common in competitive analysis that certain bounds to the input are known, as otherwise the competitive ratio may be unbounded. Also, our results can be easily adapted to work for intervals of arbitrary length, in which case any competitive ratio we achieve is penalized additively by ϵ , for every $\epsilon > 0$ that can be chosen to be arbitrarily small.

²Note that maximizing the speed of a feasible schedule is equivalent to minimizing its finishing time.

2.1.3 Robot Model for Searching

We add the following capabilities to our basic robot model in order to execute the search algorithms.

knowledge

Each individual robot has knowledge of:

- Their own searching and walking speeds,
- the direction in which they should search,
- the amount of time they should spend walking, and
- the amount of time they should spend searching.

The knowledge of time durations and search directions is given to the robots from a centralized scheduler who is aware of the number of robots in the fleet, and the speeds of each robot. We also assume that the scheduler is able to partition the robots into groups and place these groups at locations on the domain.

In the offline problem only, the scheduler also knows the length of the interval to be searched.

2.1.4 Outline and Results

We begin by studying the Beachcombers' Problem with one source in Section 2.2. In Subsection 2.2.1 we first study the properties of optimal mobility schedules. We then propose `Comb`, an optimal algorithm for the Beachcombers' Problem, which requires $O(n \log n)$ computational steps, and prove its correctness. Subsection 2.3 is devoted to online searching, where the length of the segment to be searched is not known in

advance. In this section we propose the online searching algorithm `LeapFrog`, prove its correctness and analyse its efficiency. We prove that the `LeapFrog` algorithm is 2-competitive. The competitive ratio is shown to be reduced to 1.29843 in the case when all robots' walking speeds are the same.

We then study the more general Multi-source Beachcombers' Problem in Section 2.4. Multi-source Beachcombers' Problem t -SBP, when compared to 1-SBP, add the additional algorithmic complication of partitioning to the existing scheduling problem. In Subsection 2.4.1, we explore how the added complexity of partitioning makes the problem NP-hard. In contrast, we prove that some instances of t -SBP admit polynomial time solutions.

Theorem 2.1. *It is NP-hard to find the optimal solution for arbitrary instances of 2-SBP, while W-Uniform 2-SBP instances with search speeds $s_1, \dots, s_n$ for which $\frac{1}{1-s_i} \in \mathbb{N}$ admit a $1/(1 + \epsilon^2)$ -approximation which requires $O(\frac{n^3 \log m}{\epsilon})$ steps, where $m = \max(1/(1 - s_i))$. The solution for 1-SBP is a $\frac{1}{2}$ -approximation for 2-SBP, and deterministic 0.5569-approximation for 2-SBP is also possible that requires $O(n \log n)$ many computational steps.*

Theorem 2.2. *`GreedyPartition` (Algorithm 2.5) solves instances of t -SBP with S -uniform swarms optimally in $O(n \log n)$ many steps.*

We then turn our attention to general t -SBP problems. Based on results established in Section 2.2 for 1-SBP, in Subsection 2.4.2 we explore a very efficient and elegantly simple randomized algorithm (Algorithm 2.6) with constant approximation:

Theorem 2.3. *There exists a randomized algorithm for t -SBP with expected approximation ratio $1 - (1 - \frac{1}{t})^t$ which needs $O(n \log(t))$ random bits, and runs in $O(n \log n)$ many steps.*

2.2 The Beachcombers' Problem

We proceed by first identifying in Subsection 2.2.1 a number of structural properties exhibited by every optimal solution to the Beachcombers' Problem. This will allow us to conclude in Subsection 2.2.2 that the Beachcombers' Problem can be solved efficiently.

2.2.1 Properties of Optimal Schedules

In the following lemmas we identify some properties of optimal schedules. By the observation made in the preliminaries we suppose that the segment may be divided into a sequence of sub-intervals $\sigma_1, \sigma_2, \dots, \sigma_k$, each sub-interval σ_i being solely and entirely searched by the same robot.

Lemma 2.4. *In every optimal schedule all robots terminate their work simultaneously and each robot completes its work by searching some non-empty part of the segment.*

Proof. Suppose, to the contrary, that the finishing time of some optimal schedule $\mathcal{K}$ is $T_{\mathcal{K}}$ and some robot r_e finishes its work at an earlier time $T_{\mathcal{K}} - \epsilon$, for $\epsilon > 0$. We show that it is possible to rearrange the schedule (i.e. the times of walking and searching of all robots) so that (i) the rearranged schedule $\mathcal{K}'$ is correct, (ii) it completes within time $T_{\mathcal{K}}$, (iii) robot r_e is the one which searches the very first sub-interval of the segment, and (iv) r_e finishes its work at time $\leq T_{\mathcal{K}} - \frac{\epsilon}{2}$.

Let $[0, A]$ be the first sub-interval to be searched entirely by any particular robot in $\mathcal{K}$. If it is r_e which searches $[0, A]$ in $\mathcal{K}$, no rearrangement is necessary as the desired schedule properties already hold.

Assume then that it is robot $r_f \neq r_e$ which searches $[0, A]$ in $\mathcal{K}$, as illustrated in Figure 2.1a. We make robot r_e search interval $X = [0, \min(\frac{\epsilon}{2}(w_e - s_e), A)]$ in $\mathcal{K}'$ while

FIGURE 2.1: Reducing the time of the first searcher as described in Lemma 2.4.

The top lines represent the search patterns of robots r_j and r_k . The dotted lines represent where the robots are walking and the solid lines represent where the robots are searching.

robot r_f in $\mathcal{K}$ is switched to walking over X in $\mathcal{K}'$, as shown in Figure 2.1b. Observe that, we made r_e search more than in $\mathcal{K}$, but still finishing within time $T_{\mathcal{K}} - \frac{\epsilon}{2} < T_{\mathcal{K}}$. However this permits robot r_f to switch from searching to walking in some non-empty sub-interval. As r_f was completing its work in schedule $\mathcal{K}$ within at most time T and $w_{r_f} > s_{r_f}$ robot r_f finishes now its work in time $T_{\mathcal{K}} - \epsilon_1$ for some $\epsilon_1 > 0$. Note as well that robot r_e is now searching the first sub-interval of the segment.

As a result of the above rearrangement we have a correct schedule in which a sequence of subsequent sub-intervals $\sigma_1, \sigma_2, \dots, \sigma_k$ are searched by the robots, such that the robot searching σ_1 finishes its work before time $T_{\mathcal{K}}$ (i.e. σ_1 is searched by r_e finishing not later than $T_{\mathcal{K}} - \frac{\epsilon}{2}$).

Consider now some sub-interval $\sigma_i = [A, B]$ which is searched by some robot r_g which completes its work in time $T_{\mathcal{K}} - \epsilon$ in $\mathcal{K}$ and $\sigma_{i+1} = [B, C]$ be subsequent sub-interval searched by a different robot $r_h \neq r_g$, with $0 < A < B < 1$. This is illustrated in Figure 2.2a.

FIGURE 2.2: Reducing the time of subsequent searchers as described in Lemma 2.4.

The top lines represent the search patterns of robots r_j and r_k . The dotted lines represent where the robots are walking and the solid lines represent where the robots are searching.

Similarly as above, we can make robot r_g search into sub-interval σ_{i+1} for time $\frac{\epsilon}{2}$, so that it additionally searches interval $Y = [0, \min(B + \frac{\epsilon}{2}(w_g - s_g), C)]$ permitting robot r_h to switch to walking over interval Y and so finish its work at time $T_{\mathcal{K}} - \epsilon_1$, as shown in Figure 2.2b.

Therefore, we have shown that if any robot finishes its work before time $T_{\mathcal{K}}$, we rearrange the schedule so that the finishing time of the robot searching interval σ_1 is $T_{\mathcal{K}} - \epsilon_1$. We have also shown that if interval σ_i is searched by a robot r_j which finishes its work in time $T_{\mathcal{K}} - \epsilon_i$, then we can rearrange the schedule so that r_j finishes in time $T_{\mathcal{K}} - \epsilon_i/2$ and the robot searching interval σ_{i+1} finishes its work in time $T_{\mathcal{K}} - \epsilon_{i+1}$. By induction, we can thus reduce the finishing times of all robots contradicting that schedule $\mathcal{K}$ was optimal.

In case when some robot does not participate in searching or if it finishes its work by walking, we can reduce its makespan by cutting off the last walk without affecting the correctness of the schedule. We then repeat the same argument as above. ■

Lemma 2.5. *In every optimal schedule, each robot searches a continuous interval.*

Proof. Suppose, to the contrary, that in an optimal schedule $\mathcal{K}$ some robot searches more than one interval from $\sigma_1, \sigma_2, \dots, \sigma_m$ (the set of intervals search by all robots). Let σ_q be the last interval of this sequence, such that some robot which searches σ_q , say r_j , also searches σ_p , for $1 \leq p < q - 1$. Such a schedule is illustrated in Figure 2.3a. Denote by λ the length of the segments between σ_p and σ_q , i.e. $\lambda = \sum_{i=p+1}^{q-1} |\sigma_i|$. Consider moving interval σ_p distance λ to the right and each σ_i , for $i = p+1, \dots, q-1$ distance $|\sigma_p|$ to the left. By movement of an interval we mean that the robot searching this interval changes to the searching mode in the new position of the interval and to the walking mode in the old position (except in the intersection of the intervals). This new schedule is illustrated in Figure 2.3b. Observe that such rearranged schedule $\mathcal{K}'$ is correct and all robots each search and walk the exact same distances as in $\mathcal{K}$, and so both schedules have the same finishing time. Note, however, that the robot which was searching interval σ_{q-1} in $\mathcal{K}$, say r_k , now finishes its schedule in $\mathcal{K}'$ by walking. By Lemma 2.4, an optimal schedule must have all robots conclude their work by searching a non-empty interval. Therefore $\mathcal{K}'$ is not optimal, and neither is $\mathcal{K}$ since it has the same finishing time as $\mathcal{K}'$. ■

An implication of Lemmas 2.4 and 2.5 is that every optimal schedule is defined by a sequence of intervals $\sigma_1, \sigma_2, \dots, \sigma_n$, such that robot r_i walks at maximal speed through intervals $\sigma_1, \dots, \sigma_{i-1}$ and finishes its work by searching σ_i (we suppose w.l.o.g. that we rename the robots with respect to the order of the intervals). The next lemma shows that such robots must be arranged in this sequence by the non-decreasing order of their walking speeds.

Lemma 2.6. *In every optimal schedule, for any two robots r_i, r_j with $w_i < w_j$, robot r_i searches a sub-interval closer to the starting point than the sub-interval of robot r_j .*

FIGURE 2.3: Search configurations described in Lemma 2.5

The top lines represent the search patterns of robots r_j and r_k . The dotted lines represent where the robots are walking and the solid lines represent where the robots are searching.

Proof. Suppose, to the contrary, that there is an optimal schedule $\mathcal{K}$ where the robots are not arranged by the non-decreasing walking speed. Then there must exist a pair of consecutive robots r_i, r_{i+1} in $\mathcal{K}$ with $w_i > w_{i+1}$, searching, respectively, the consecutive intervals σ_i, σ_{i+1} . In what follows, we investigate the effect of switching the order of the search intervals of two robots r_i, r_{i+1} , so that the union of the intervals remains unchanged. In particular we will redistribute the portions of the union of the two intervals that each robot will search. Since we will only redistribute the union of intervals σ_i, σ_{i+1} , the remaining sub-intervals will remain the same, and so will the finishing search times of the remaining robots.

For the sake of notational convenience, let $|\sigma_i| = c_i$ and $|\sigma_{i+1}| = c_{i+1}$. We may also assume that $c_i + c_{i+1} = 1$, and therefore that $c_i = \lambda$ and that $c_{i+1} = 1 - \lambda$, after proper scaling of the intervals. This scenario is illustrated in Figure 2.4a.

Note that robot r_i searches σ_i while robot r_{i+1} walks within σ_i and searches σ_{i+1} . Each robot will have to walk the distance $x \geq 0$ between the start point and the leftmost point

FIGURE 2.4: Search configurations described in Lemma 2.6.

The top lines represent the search patterns of robots r_i and r_{i+1} . The dotted lines represent where the robots are walking and the solid lines represent where the robots are searching.

of σ_i . By Lemma 2.4 all robots have the same finishing time, so we have $\frac{x}{w_i} + \frac{\lambda}{s_i} = \frac{x}{w_{i+1}} + \frac{\lambda}{w_{i+1}} + \frac{1-\lambda}{s_{i+1}}$. Solving for λ gives $\lambda = \left(\frac{x}{w_{i+1}} - \frac{x}{w_i} + \frac{1}{s_{i+1}} \right) / \left(\frac{1}{s_i} + \frac{1}{s_{i+1}} - \frac{1}{w_{i+1}} \right)$. Therefore, as $\lambda \geq 0$ (by Lemmas 2.4, 2.5), the finishing time for $\mathcal{K}$ is

$$T_{\mathcal{K}} = \frac{x}{w_i} + \frac{\lambda}{s_i} = \frac{x}{w_i} + \frac{\frac{x}{s_i} \left(\frac{1}{w_{i+1}} - \frac{1}{w_i} \right)}{\frac{1}{s_i} + \frac{1}{s_{i+1}} - \frac{1}{w_{i+1}}} + \frac{1}{\left(\frac{1}{s_i} + \frac{1}{s_{i+1}} - \frac{1}{w_{i+1}} \right) s_i s_{i+1}}.$$

We now reschedule the robots to a new schedule $\mathcal{K}'$ where robot r_{i+1} searches first, say a μ portion of $c_i + c_{i+1} = 1$, and robot r_i searches the remaining sub-interval of length $(1 - \mu)(c_i + c_{i+1}) = 1 - \mu$. This new configuration is illustrated in Figure 2.4b. This means that robot r_i will now walk the interval of length μ . Since by Lemma 2.4 the two robots must finish simultaneously, the same reasoning shows that $\frac{x}{w_i} + \frac{\mu}{w_i} + \frac{1-\mu}{s_i} = \frac{x}{w_{i+1}} + \frac{\mu}{s_{i+1}}$. As before, we can eliminate μ , so as to conclude that

$$T'_{\mathcal{K}} = \frac{x}{w_{i+1}} + \frac{\mu}{s_{i+1}} = \frac{x}{w_{i+1}} + \frac{\frac{x}{s_{i+1}} \left(\frac{1}{w_i} - \frac{1}{w_{i+1}} \right)}{\frac{1}{s_i} + \frac{1}{s_{i+1}} - \frac{1}{w_i}} + \frac{1}{\left(\frac{1}{s_i} + \frac{1}{s_{i+1}} - \frac{1}{w_i} \right) s_i s_{i+1}}.$$

We show below that $T_{\mathcal{K}} > T'_{\mathcal{K}}$. Observe first that, since $w_i > w_{i+1}$ we have

$$Q = \left(\frac{1}{\left(\frac{1}{s_i} + \frac{1}{s_{i+1}} - \frac{1}{w_{i+1}}\right) s_i s_{i+1}} - \frac{1}{\left(\frac{1}{s_i} + \frac{1}{s_{i+1}} - \frac{1}{w_i}\right) s_i s_{i+1}} \right) > 0$$

Hence

$$\begin{aligned} T_{\mathcal{K}} - T'_{\mathcal{K}} &= \left(\frac{x}{w_i} + \frac{\frac{x}{s_i} \left(\frac{1}{w_{i+1}} - \frac{1}{w_i} \right)}{\frac{1}{s_i} + \frac{1}{s_{i+1}} - \frac{1}{w_{i+1}}} + \frac{1}{\left(\frac{1}{s_i} + \frac{1}{s_{i+1}} - \frac{1}{w_{i+1}}\right) s_i s_{i+1}} \right) \\ &\quad - \left(\frac{x}{w_{i+1}} + \frac{\frac{x}{s_{i+1}} \left(\frac{1}{w_i} - \frac{1}{w_{i+1}} \right)}{\frac{1}{s_i} + \frac{1}{s_{i+1}} - \frac{1}{w_i}} + \frac{1}{\left(\frac{1}{s_i} + \frac{1}{s_{i+1}} - \frac{1}{w_i}\right) s_i s_{i+1}} \right) \\ &= x \left(\left(\frac{1}{w_i} - \frac{1}{w_{i+1}} \right) + \frac{\frac{1}{s_i} \left(\frac{1}{w_{i+1}} - \frac{1}{w_i} \right)}{\frac{1}{s_i} + \frac{1}{s_{i+1}} - \frac{1}{w_{i+1}}} + \frac{\frac{1}{s_{i+1}} \left(\frac{1}{w_i} - \frac{1}{w_{i+1}} \right)}{\frac{1}{s_i} + \frac{1}{s_{i+1}} - \frac{1}{w_i}} \right) \\ &\quad + Q \\ &> x \left(- \left(\frac{1}{w_{i+1}} - \frac{1}{w_i} \right) + \frac{\frac{1}{s_i} \left(\frac{1}{w_{i+1}} - \frac{1}{w_i} \right)}{\frac{1}{s_i} + \frac{1}{s_{i+1}}} + \frac{\frac{1}{s_{i+1}} \left(\frac{1}{w_i} - \frac{1}{w_{i+1}} \right)}{\frac{1}{s_i} + \frac{1}{s_{i+1}}} \right) \\ &\quad + Q \end{aligned}$$

where the last term above equals Q which is > 0 . Therefore in schedule $\mathcal{K}'$ the finishing time of both robots r_i, r_j is smaller than in $\mathcal{K}$, which by Lemma 2.4 contradicts the optimality of $\mathcal{K}$. ■

By using the properties obtained in Lemmas 2.4, 2.5 and 2.6, we determine a useful recurrence for the sub-intervals searched by robots in an optimal schedule.

Lemma 2.7. *Let the robots $r_1, r_2, \dots, r_n$ be ordered in non-decreasing walking speed, and suppose that T_{opt} is the time of the optimal schedule. Then, the segment to be searched may be partitioned into successive sub-segments of lengths $c_1, c_2, \dots, c_n$ and*

the optimal schedule assigns to robot r_i the i^{th} interval of length c_i , where the length c_i satisfies the following recursive formula, and where we assume, without loss of generality, that $w_0 = 0$, $s_0 = 1$, and $w_1 = 1$.³

$$c_0 = 0; \quad c_k = \frac{s_k}{w_k} \left(\left(\frac{w_{k-1}}{s_{k-1}} - 1 \right) c_{k-1} + T_{opt}(w_k - w_{k-1}) \right), \quad k \geq 1 \quad (2.1)$$

Proof. From Lemma 2.5 we know that all robots search contiguous intervals. Since by Lemma 2.4 we need to utilize all robots, it follows that the optimal schedule defines a partition of the unit domain into n sub-intervals. Finally by Lemma 2.6, if we order the robots in non-decreasing walking speed, then robot r_i will search the i^{th} in a row interval, showing the first claim of the lemma.

Now, from Lemma 2.4, we know that all robots finish at the same time, T_{opt} . Since all robots start processing the domain at the same time, robot k will walk its initial sub-interval of length $\sum_{i=1}^{k-1} c_i$ in time proportional to $1/w_k$, and in the remaining time it will search the interval of length c_k . Hence

$$c_k = s_k \left(T_{opt} - \frac{\sum_{i=1}^{k-1} c_i}{w_k} \right), \quad (2.2)$$

from which we easily see that the recursion holds for c_1 . We will now show that the recursion holds for $k \geq 2$. First, we know from (2.2) that the value of c_{k-1} is

$$c_{k-1} = s_{k-1} \left(T_{opt} - \frac{\sum_{i=1}^{k-2} c_i}{w_{k-1}} \right),$$

³We set $w_0 = 0$, $s_0 = 1$ and $w_1 = 1$ for notational convenience, so that (2.1) holds. Note that w_0 and s_0 does not correspond to any robot, so it does not matter that $s_0 > w_0$. We note that w_1 is the walking speed of the robot, which will search the first sub-interval and so will never enter walking mode. Hence, w_1 does not affect our solution.

which we can rearrange to show

$$\sum_{i=1}^{k-2} c_i = w_{k-1} \left(T_{\text{opt}} - \frac{c_{k-1}}{s_{k-1}} \right). \quad (2.3)$$

We can rewrite (2.2) as

$$\begin{aligned} c_k &= s_k \left(T_{\text{opt}} - \frac{c_{k-1} + \sum_{i=1}^{k-2} c_i}{w_k} \right) \\ &\stackrel{(2.3)}{=} s_k \left(T_{\text{opt}} - \frac{c_{k-1} + w_{k-1} \left(T_{\text{opt}} - \frac{c_{k-1}}{s_{k-1}} \right)}{w_k} \right) \\ &= \frac{s_k}{w_k} \left(\left(\frac{w_{k-1}}{s_{k-1}} - 1 \right) c_{k-1} + T_{\text{opt}}(w_k - w_{k-1}) \right). \end{aligned}$$

■

2.2.2 The Optimal Schedule

As a consequence of Lemmas 2.4, 2.5 and 2.6 we have the following offline algorithm `Comb` producing an optimal schedule. The algorithm is parametrized by the real values c_i equal to the sizes of intervals to be searched by each robot r_i .

Figure 2.5a illustrates the schedule produced by algorithm `Comb` for a set of five robots. The trajectory of each robot (except the first one) is formed by a segment of walking (thin line) followed by a segment of searching (bold). The dashed line corresponds to the slope representing the speed S_{opt} of the schedule. Each sub-segment C_i in Fig. 1(a) has length c_i stated in Lemma 2.7.

We can now prove the following theorem:

Theorem 2.8. *The Beachcombers' Problem can be solved optimally in $O(n \log n)$ many steps.*

Proof. By Lemma 2.7 we need to order the robots by non-decreasing walking speed, which requires $O(n \log n)$ many steps. We then show how to compute all c_i in linear number of steps, modulo the arithmetic operations that depend on the encoding sizes of w_i, s_i .

Consider an imaginary unit time period. Starting with the slowest, for each robot, we use equation (2.1) to compute (in constant time) the sub-interval y_i it would search if it were to remain active for the unit time period. Consequently, we can compute in $O(n)$ steps the total length $\sum_{i=1}^n y_i$ of the interval that the collection of robots can search within a unit time period. This schedule, scaled to a unit domain, will have finishing time $T = 1 / \sum_{i=1}^n y_i$. The length of the interval that robot r_k will search is then $c_k = y_k / \sum_{i=1}^n y_i$. ■

Algorithm 2.1: Comb

Input: Set of robots R with associated walking and patrolling speeds

Output: Schedule of R

```

1 Sort the robots in non-decreasing walking speeds
2 Let  $c_i \leftarrow \frac{s_k}{w_k} ((w_{k-1}/s_{k-1} - 1) c_{k-1} + w_k - w_{k-1})$  for  $i = 1, \dots, n$ 
3  $L \leftarrow \sum_{i=1}^n c_i$ 
4 for  $i = 1, \dots, n$  do
5   |  $c_i \leftarrow c_i / L$ 
6 end
7 for  $i = 1, \dots, n$  do
8   | for  $j = 1, \dots, i - 1$  do
9     | WALK( $c_j$ )
10    | SEARCH( $c_i$ )
11   | end
12 end

```

2.2.3 Closed Formulas for the Optimal Schedule

From the proof of Theorem 2.8 we can implicitly derive the time (and the speed) of an optimal solution to the Beachcombers' Problem. In what follows, we assume that the robots are ordered in non-decreasing walking speeds, that $w_0 = 0$, that $s_0 = 1$, and that $w_1 = 1$. These assumptions are justified on the same basis as in Lemma 2.7.

Lemma 2.9. *Consider a set of robots such that in the optimal schedule each robot finishes searching in time T_{opt} . Robot r_k will search a sub-interval of length c_k , such that*

$$\frac{c_k}{T_{opt}} = s_k - \frac{s_k}{w_k} \sum_{r=1}^{k-1} s_r \prod_{j=r+1}^{k-1} \left(1 - \frac{s_j}{w_j}\right) \quad (2.4)$$

Proof. We begin by noting that when $k = 1$ the summation evaluates to zero, since the beginning index ($r = 1$) comes after the final index ($k - 1 = 0$). Similar reasoning results in the product evaluating to one when $r + 1 > k - 1$.

We will now prove (2.4) by induction on k . For $k = 1$, the above observation along with knowing that $w_1 = 1$ reduces (2.4) to $\frac{c_k}{T_{opt}} = s_k$, which follows immediately from equation (2.1). Suppose, by inductive hypothesis, that

$$\frac{c_{k-1}}{T_{opt}} = s_{k-1} - \frac{s_{k-1}}{w_{k-1}} \sum_{r=1}^{k-2} s_r \prod_{j=r+1}^{k-2} \left(1 - \frac{s_j}{w_j}\right). \quad (2.5)$$

We have

$$\begin{aligned}
\frac{c_k}{T_{\text{opt}}} &\stackrel{(2.1)}{=} \frac{s_k}{w_k} \left(\left(\frac{w_{k-1}}{s_{k-1}} - 1 \right) \frac{c_{k-1}}{T_{\text{opt}}} + (w_k - w_{k-1}) \right) \\
&\stackrel{(2.5)}{=} \frac{s_k}{w_k} \left(\frac{w_{k-1}}{s_{k-1}} - 1 \right) \left(s_{k-1} - \frac{s_{k-1}}{w_{k-1}} \sum_{r=1}^{k-2} s_r \prod_{j=r+1}^{k-2} \left(1 - \frac{s_j}{w_j} \right) \right) \\
&\quad + \frac{s_k}{w_k} (w_k - w_{k-1}) \\
&= \frac{s_k}{w_k} \left(\left(\frac{w_{k-1}}{s_{k-1}} - 1 \right) s_{k-1} + (w_k - w_{k-1}) \right) \\
&\quad - \frac{s_k}{w_k} \left(\left(1 - \frac{s_{k-1}}{w_{k-1}} \right) \sum_{r=1}^{k-2} s_r \prod_{j=r+1}^{k-2} \left(1 - \frac{s_j}{w_j} \right) \right) \\
&= s_k - \frac{s_k}{w_k} \left(s_{k-1} + \sum_{r=1}^{k-2} s_r \prod_{j=r+1}^{k-1} \left(1 - \frac{s_j}{w_j} \right) \right) \\
&= s_k - \frac{s_k}{w_k} \sum_{r=1}^{k-1} s_r \prod_{j=r+1}^{k-1} \left(1 - \frac{s_j}{w_j} \right).
\end{aligned}$$

■

Definition 2.10 (Search Power). Consider a set of n robots $r_1, r_2, \dots, r_n$, with $s_i < w_i$, $i = 1, \dots, n$. We define the *search power* of any subset A of robots using a real function $g : 2^{[n]} \mapsto \mathbb{R}^+$ as follows: For any subset A , first sort the robots in non-decreasing walking speeds w_i , and let $w_1^A, \dots, w_{|A|}^A$ be that ordering (the superscripts just indicate membership in A). Also let $s_1^A, \dots, s_{|A|}^A$ be the corresponding search speeds of the ordered robots of the subset $|A|$ (again, the superscripts indicate membership in A). We define the evaluation function (search power of set A) as

$$g(A) := \sum_{k=1}^{|A|} s_k^A \prod_{j=k+1}^{|A|} \left(1 - \frac{s_j^A}{w_j^A} \right).$$

The search power of any subset of the robots is well defined, and it is always positive (since $s_i < w_i$).

Theorem 2.11. *The speed S_{opt} of the optimal schedule equals the search power of the collection of robots. In other words, if N denotes the set of all robots, then $S_{\text{opt}} = g(N)$.*

Proof. We begin by noting that S_{opt} is equal to the sum of (2.4) for $k = 1, \dots, n$:

$$S_{\text{opt}} = \sum_{k=1}^n \frac{c_k}{T_{\text{opt}}} = \sum_{k=1}^n \left(s_k - \frac{s_k}{w_k} \sum_{r=1}^{k-1} s_r \prod_{j=r+1}^{k-1} \left(1 - \frac{s_j}{w_j} \right) \right).$$

We therefore want to show that

$$\sum_{k=1}^n \left(s_k - \frac{s_k}{w_k} \sum_{r=1}^{k-1} s_r \prod_{j=r+1}^{k-1} \left(1 - \frac{s_j}{w_j} \right) \right) = \sum_{k=1}^n s_k \prod_{j=k+1}^n \left(1 - \frac{s_j}{w_j} \right).$$

We will now show that $S_{\text{opt}} = g(N)$ by induction on the number of robots $n = |N|$.

When $n = 1$, we can easily see that $S_{\text{opt}} = g(N) = s_1$. Suppose by inductive hypothesis that

$$\sum_{k=1}^{n-1} \left(s_k - \frac{s_k}{w_k} \sum_{r=1}^{k-1} s_r \prod_{j=r+1}^{k-1} \left(1 - \frac{s_j}{w_j} \right) \right) = \sum_{k=1}^{n-1} s_k \prod_{j=k+1}^{n-1} \left(1 - \frac{s_j}{w_j} \right). \quad (2.6)$$

We have

$$\begin{aligned}
& \sum_{k=1}^n \left(s_k - \frac{s_k}{w_k} \sum_{r=1}^{k-1} s_r \prod_{j=r+1}^{k-1} \left(1 - \frac{s_j}{w_j} \right) \right) \\
&= s_n - \frac{s_n}{w_n} \sum_{r=1}^{n-1} s_r \prod_{j=r+1}^{n-1} \left(1 - \frac{s_j}{w_j} \right) + \sum_{k=1}^{n-1} \left(s_k - \frac{s_k}{w_k} \sum_{r=1}^{k-1} s_r \prod_{j=r+1}^{k-1} \left(1 - \frac{s_j}{w_j} \right) \right) \\
&\stackrel{(2.6)}{=} s_n - \frac{s_n}{w_n} \sum_{r=1}^{n-1} s_r \prod_{j=r+1}^{n-1} \left(1 - \frac{s_j}{w_j} \right) + \sum_{k=1}^{n-1} s_k \prod_{j=k+1}^{n-1} \left(1 - \frac{s_j}{w_j} \right) \\
&= s_n + \left(1 - \frac{s_n}{w_n} \right) \sum_{k=1}^{n-1} s_k \prod_{j=k+1}^{n-1} \left(1 - \frac{s_j}{w_j} \right) \\
&= s_n + \sum_{k=1}^{n-1} s_k \prod_{j=k+1}^n \left(1 - \frac{s_j}{w_j} \right) \\
&= \sum_{k=1}^n s_k \prod_{j=k+1}^n \left(1 - \frac{s_j}{w_j} \right).
\end{aligned}$$

■

2.3 The Online Beachcombers' Problem

In this section we present an algorithm which produces a searching schedule for a segment of unknown size. For reasons that will become apparent, we require that the length of the interval be some multiple of a number μ . The value of μ is known to the robots in advance, and is used to coordinate their movements. Each robot executes the same sequence of movements for each interval of length μ of the segment. Therefore, the speed of searching each interval of length μ is the same. We call this the *swarm speed*.

Contrary to the offline case, where all robots finish searching at the same time (at different positions), in the online algorithm the robots arrive at the end point of each interval of length μ at the same time. Because of this synchronicity, we require the segment to

be a discrete length, as we cannot guarantee the performance of our algorithm if the end of the segment occurs in the middle of one of the intervals. Robots which cannot contribute to increasing the swarm speed are not used in the schedule. Each utilized robot r_i (called a *swarm robot*) searches a sub-segment of the unit segment of size c_i and walks along the remainder. The lengths of the sub-segments c_i are chosen to synchronize the arrival of all robots at every end point of an interval of length μ . The sub-segments are pairwise interior disjoint and their union covers the entire interval, i.e. $\sum_{i=1}^k c_i = \mu$.

For the remainder of this section, we will assume that $\mu = 1$ (so the intervals considered will be unit intervals), but any solution can be easily scaled to any value of μ .

Theorem 2.12. *Consider a partition of the unit interval $I = [0, 1]$ into k consecutive non-overlapping segments $C_1, C_2, \dots, C_k$, from left to right, of lengths $c_1, c_2, \dots, c_k$, respectively. Assume that all the robots start (at endpoint 0) and finish (at endpoint 1) simultaneously. Further assume that the i_{th} robot r_i searches the segment C_i with speed s_i and walks the rest of the interval $I \setminus C_i$ with speed w_i such that $w_i > s_i$. Then the swarm speed satisfies $S = \frac{\sum_{i=1}^k \frac{1}{\delta_i}}{1 + \sum_{i=1}^k \frac{1}{w_i \delta_i}}$, where $\delta_i := \frac{1}{s_i} - \frac{1}{w_i}$, for $i = 1, 2, \dots, k$.*

Proof. The partition of the interval $[0, 1]$ into segments as prescribed in the statement of the theorem gives rise to the equation

$$c_1 + c_2 + \dots + c_k = 1. \quad (2.7)$$

Let s be the speed of the swarm of n robots. Since all the robots must reach the other endpoint 1 of the interval at the same time, we have the following identities.

$$\frac{c_i}{s_i} + \frac{1 - c_i}{w_i} = \frac{1}{S}, \text{ for } 1 \leq i \leq k, \quad (2.8)$$

where $\frac{c_i}{s_i}$ is the time spent searching and $\frac{1-c_i}{w_i}$ the time spent walking by robot r_i . Using the notation $\delta_i := \frac{1}{s_i} - \frac{1}{w_i}$, and substituting into Equation (2.8), after simplifications we get $c_i = \frac{1}{S\delta_i} - \frac{1}{w_i\delta_i}$, for $1 \leq i \leq k$. Using Equation (2.7) we see that $1 = \sum_{i=1}^k c_i = \sum_{i=1}^k \frac{1}{S\delta_i} - \sum_{i=1}^k \frac{1}{w_i\delta_i} = \frac{1}{S} \sum_{i=1}^k \frac{1}{\delta_i} - \sum_{i=1}^k \frac{1}{w_i\delta_i}$, which implies the theorem, as desired.

■

Algorithm 2.2: SwarmSpeed

Input: Set of robots R with associated walking and patrolling speeds

Output: Maximum swarm speed of R

```

1  $S \leftarrow 0$ ,  $S_{num} \leftarrow 0$ ,  $S_{den} \leftarrow 1$ ,  $\delta$ 
2  $i \leftarrow 1$ 
3 while  $i \leq n$  and  $S < w_i$  do
4    $\delta \leftarrow 1/(\frac{1}{s_i} - \frac{1}{w_i})$ 
5    $S_{num} \leftarrow S_{num} + \delta$ 
6    $S_{den} \leftarrow S_{den} + \delta/w_i$ 
7    $S \leftarrow \frac{S_{num}}{S_{den}}$ 
8    $i \leftarrow i + 1$ 
9 end
10 return  $S$ 

```

Algorithm 2.3: LeapFrog

Input: Set of robots R with associated walking and patrolling speeds, robot $r_j \in R$
Output: Search schedule for robot r_j

```

1  $S \leftarrow \text{SwarmSpeed}()$ 
2 if  $w_j \leq S$  then
3   EXIT // robot  $r_j$  stays motionless
4 else
5   for  $i \leftarrow 1$  to  $j - 1$  do
6      $c_i \leftarrow (\frac{1}{S} - \frac{1}{w_i})/(\frac{1}{s_i} - \frac{1}{w_i})$ 
7     WALK( $c_i$ )
8   end
9    $c_j \leftarrow (\frac{1}{S} - \frac{1}{w_j})/(\frac{1}{s_j} - \frac{1}{w_j})$ 
10  while not at line end do
11    SEARCH( $c_j$ )
12    WALK( $1 - c_j$ )
13  end
14 end

```

We define the procedure `SwarmSpeed` which finds the speed of a swarm in linear time and algorithm `LeapFrog` which defines the swarm. Algorithm `LeapFrog` defines the schedules for a swarm of k robots $r_1, r_2, \dots, r_k$ out of the original n robots such that $w_1 \geq w_2 \geq \dots \geq w_k$.

Once the swarm speed has been computed, we can compute the sub-segments' lengths c_i , that we call the *contribution* of robot r_i - the fraction of the unit interval that r_i is allotted to search.

The schedule produced by algorithm `LeapFrog` is illustrated at Figure 2.5b. We used the same set of robots as in the offline case, i.e. robots having identical walking and searching speeds as those in Figure 2.5a. Observe that robot r_1 , while useful in the Comb algorithm, was not used by `LeapFrog` algorithm in Figure 2.5b, since its walking speed is smaller than the swarm speed. The swarm speed is represented by the slope of the dashed line in Figure 2.5b.

Lemma 2.13. *Algorithm `LeapFrog` is correct (every point of $[0, +\infty)$ is searched).*

Proof. Let $C_j(i)$ denote the sub-segment of $[i, i + 1]$ of length c_j which is searched by robot r_j . The lemma follows from the observation that $\bigcup_{j=1}^k C_j(i) = [i, i + 1]$, for all $i \geq 0$ and $j = 1, \dots, k$. ■

2.3.1 Competitiveness of the Online Searching

In this section we discuss the competitiveness of the `LeapFrog` algorithm. Since competitive ratio is naturally discussed more often for cost optimization (minimization) problems, we assume in this section that we compare the finishing time (rather than speed) of the online versus offline solution. We show first that in the general case the `LeapFrog` algorithm is 2-competitive.

(A) Schedule produced by Comb.

(B) Schedule produced by LeapFrog.

FIGURE 2.5: Schedules produced by (a) Comb and (b) LeapFrog.

The robots used in the algorithms are $\{(\frac{1}{2}, \frac{1}{12}), (\frac{6}{13}, \frac{1}{4}), (1, \frac{1}{3}), (\frac{5}{3}, \frac{1}{12}), (6, \frac{1}{3})\}$, where robot $r_i = (w_i, s_i)$.

Theorem 2.14. *Consider a set of robots $r_1, r_2, \dots, r_n$, ordered by non-decreasing walking speed. If the completion time of the schedule produced by Comb equals T_{opt} then the completion time T_{online} of the searching schedule produced by LeapFrog is such that $T_{online} < 2 \cdot T_{opt}$.*

Proof. As LeapFrog outputs schedules of the same speed for all integer-length segments it is sufficient to analyse its competitiveness for a unit segment. Assume, to

the contrary, that the time T_{online} of the schedule output by LeapFrog is such that $T_{\text{online}} \geq 2 \cdot T_{\text{opt}}$. The swarm speed S of LeapFrog is then at most $S \leq 1/(2 \cdot T_{\text{opt}})$. Consider $C_1, C_2, \dots, C_n$ – the sub-segments searched by robots $r_1, r_2, \dots, r_n$, respectively. Recall that each robot r_i of Comb walks along segments $C_1, C_2, \dots, C_{i-1}$ and searches C_i arriving at its right endpoint at time T_{opt} . Let i^* be the index such that the midpoint $1/2 \in C_{i^*}$ (or point $1/2$ is a common endpoint of C_{i^*} and C_{i^*+1}). Observe, that in time $2 \cdot T_{\text{opt}}$ each robot r_i , such that $i \geq i^*$ could reach the right endpoint of the unit segment, while searching its portion of length $2 \cdot |C_i|$. For each robot r_i , such that $i \geq i^*$, we have $w_i > 1/(2 \cdot T_{\text{opt}}) \geq S$, each such robot is used by LeapFrog in lines 5-9. However, since $\sum_{i=i^*}^n 2 \cdot |C_i| > 1$ all robots r_i , for $i \geq i^*$ search a segment longer than 1, arriving at its right endpoint within time $2 \cdot T_{\text{opt}}$, or $T_{\text{online}} < 2 \cdot T_{\text{opt}}$ for the unit segment. This contradicts the earlier assumption. ■

Observe that, the competitive ratio of 2 may be approached as close as we want.

Proposition 2.15. *For any sufficiently small $\epsilon > 0$ there is a set of two robots for which the LeapFrog algorithm produces a schedule of completion time T_{online} such that $T_{\text{online}} = (2 - \epsilon)T_{\text{opt}}$.*

Proof. Let the speeds of the two robots be $s_1 = 1 - \epsilon/2, w_1 = 1, s_2 = 1, w_2 = (2 - \epsilon)/\epsilon$. As the swarm speed S computed in SwarmSpeed procedure equals 1, the line 2 of the LeapFrog algorithm excludes r_1 from the swarm, so the search is performed uniquely by r_2 with $T_{\text{online}} = 1$. Using Theorem 2.11 we get $S_{\text{opt}} = \sum_{k=1}^2 s_k \prod_{j=k+1}^2 \left(1 - \frac{s_j}{w_j}\right) = \left(1 - \frac{\epsilon}{2}\right) \left(1 - \frac{\epsilon}{2 - \epsilon}\right) + 1 = 2 - \epsilon$. Hence $T_{\text{opt}} = 1/(2 - \epsilon)$ and $T_{\text{online}} = 1 = (2 - \epsilon)T_{\text{opt}}$ ■

We conclude this section by conjecturing that no online algorithm exists which has a competitive ratio better than LeapFrog. If this conjecture is true, it would mean that LeapFrog is optimal.

Conjecture 2.16. *No online algorithm exists which solves the Online Beachcombers' Problem with the competitive ratio of $(2 - \epsilon)$ for any $\epsilon > 0$.*

Our reasoning behind the conjecture is this: Observe that, instead of repeating the same search pattern for each unit-length segment, we could scale down such schedule and repeat it for a segment of arbitrary small value $\epsilon > 0$. This would permit to keep the swarm speed S within an arbitrarily small range also for an interval of length being any real value (greater than 1) thus obtaining a competitive ratio of $(2 - \epsilon)$ for any $\epsilon > 0$. This conjecture is made only for the class of algorithms where the robots cannot communicate. A model which allows communication could possibly beat this ratio, and would be an interesting problem to study.

2.3.2 Online Searching with W-uniform swarms

The following theorem concerns the competitiveness of the LeapFrog algorithm in the special case when all robot walking speeds are the same.

Consider the function f_n defined as

$$f_n(\alpha) := \frac{(\alpha(n-1) + 1)(1 - (1-\alpha)^n)}{\alpha n},$$

and let $\alpha_n = \operatorname{argmax} f_n(\alpha)$. In this section we will prove the following theorem:

Theorem 2.17. *Let be given the collection of robots $r_1, r_2, \dots, r_n$ with the same walking speed $w = w_1 = \dots = w_n$. The LeapFrog algorithm has the competitive ratio $f_n(\alpha_n)$ which is increasing in n . In particular, $f_2(\alpha_2) = 1.115$, $f_3(\alpha_3) \approx 1.17605$, $f_4(\alpha_4) \approx 1.20386$ and $\lim_{n \rightarrow \infty} f_n(\alpha_n) \approx 1.29843$.*

Our strategy for proving Theorem 2.17 is to show that, among all problem instances when walking speeds are the same, the competitive ratio of `LeapFrog` is maximized when all robots' searching speeds are also the same.

We call *W-uniform* the instance of the Beachcombers' Problem in which all robots have the same walking speeds. Moreover if the searching speeds are the same - the problem is *totally uniform*. Clearly all n robots participate in the swarm of the `LeapFrog` algorithm. Considering the speeds of the schedules obtained by the offline and online algorithms given by Theorem 2.11 and Theorem 2.12, the upper bound on the competitive ratio C of the `LeapFrog` algorithm is given by

$$C = \sup_R \frac{T_{\text{online}}}{T_{\text{opt}}} = \sup_R \left(\sum_{k=1}^n s_k \prod_{j=k+1}^n \left(1 - \frac{s_j}{w_j} \right) \right) / \left(\frac{\sum_{k=1}^n \frac{1}{\delta_k}}{1 + \sum_{k=1}^n \frac{1}{w_k \delta_k}} \right)$$

where the supremum of the ratio is taken over all configurations R of robots' speeds and $\delta_i := \frac{1}{s_i} - \frac{1}{w_i}$, for $i = 1, 2, \dots, k$.

Observe that this ratio remains the same if the instance of the problem is constrained to the case where all robots have walking speeds equal to 1. Substituting this value in, and rearranging the above equation, we get

$$C = \sup_R \left(\sum_{k=1}^n s_k \prod_{j=k+1}^n (1 - s_j) \right) \left(1 + \frac{1}{\sum_{k=1}^n \frac{s_k}{1-s_k}} \right).$$

When the terms summation and product are fully expanded (shown in Claim 2.18), they can be re-collected to give the following equation:

$$C = \sup_R \left(1 - \prod_{k=1}^n (1 - s_k) \right) \left(1 + \frac{1}{\sum_{k=1}^n \frac{s_k}{1-s_k}} \right) \quad (2.9)$$

Claim 2.18. $\sum_{k=1}^n s_k \prod_{j=k+1}^n (1 - s_j) = 1 - \prod_{k=1}^n (1 - s_k)$

Proof. The claim will be proven by induction on n . When $n = 1$, both expressions evaluate to s_1 . Suppose by inductive hypothesis that

$$\sum_{k=1}^{n-1} s_k \prod_{j=k+1}^{n-1} (1 - s_j) = 1 - \prod_{k=1}^{n-1} (1 - s_k). \quad (2.10)$$

We have that

$$\begin{aligned} \sum_{k=1}^n s_k \prod_{j=k+1}^n (1 - s_j) &= s_n + \sum_{k=1}^{n-1} s_k \prod_{j=k+1}^n (1 - s_j) \\ &= s_n + (1 - s_n) \sum_{k=1}^{n-1} s_k \prod_{j=k+1}^{n-1} (1 - s_j) \\ &\stackrel{(2.10)}{=} s_n + (1 - s_n) \left(1 - \prod_{k=1}^{n-1} (1 - s_k) \right) \\ &= s_n + 1 - s_n - (1 - s_n) \prod_{k=1}^{n-1} (1 - s_k) \\ &= 1 - \prod_{k=1}^n (1 - s_k). \end{aligned}$$

■

In what follows we compute a numeric upper bound for (2.9). Such a task seems challenging as it involves n many parameters, i.e. the searching speeds. As the expression is symmetric in the parameters, one should expect that it is maximized when all parameters are the same. Effectively, this would mean that competitive ratio of our algorithm is worst only for totally uniform instances, i.e. where all searching speeds are the same and all walking speeds are the same. This is what we make formal in the next technical lemma.

Lemma 2.19. *Given some fixed n , expression (2.9) is maximized for totally uniform instances of Beachcombers' Problem.*

Proof. Consider the function $f : (0, 1)^n \mapsto \mathbb{R}_+$ defined as

$$f(s_1, s_2, \dots, s_n) = \left(1 - \prod_{k=1}^n (1 - s_k)\right) \left(1 + \frac{1}{\sum_{k=1}^n \frac{s_k}{1 - s_k}}\right).$$

A necessary condition for optimality is that $\frac{\partial f}{\partial s_i} = 0$ for $i = 1, \dots, n$. Towards computing the partial derivatives we introduce the shorthands

$$Q := \sum_{k=1}^n \frac{s_k}{1 - s_k} \quad \text{and} \quad P := \prod_{k=1}^n (1 - s_k),$$

and we observe that

$$\frac{\partial}{\partial s_i} Q = \frac{\partial}{\partial s_i} \frac{s_i}{1 - s_i} = \frac{1}{(1 - s_i)^2}$$

and that

$$\frac{\partial}{\partial s_i} P = - \prod_{k=1, \dots, n \text{ \& } k \neq i}^n (1 - s_k) = - \frac{1}{1 - s_i} P.$$

Then, we easily get that

$$\begin{aligned} f &= (1 - P) \left(1 + \frac{1}{Q}\right) \\ \frac{\partial f}{\partial s_i} &= - \frac{\partial P}{\partial s_i} \left(1 + \frac{1}{Q}\right) + (1 - P) \left(\frac{-\frac{\partial Q}{\partial s_i}}{Q^2}\right) \\ &= - \left(-\frac{1}{1 - s_i} P\right) \left(1 + \frac{1}{Q}\right) + (1 - P) \left(\frac{-\frac{1}{(1 - s_i)^2}}{Q^2}\right) \end{aligned}$$

Requiring that the above partial derivative identifies with 0 and solving for s_i gives

$$s_i = 1 - \frac{1 - P}{PQ(Q + 1)}. \quad (2.11)$$

Note that this already shows that all s_i are equal when f is maximized. In order to complete the proof, we need to show that these values of s_i are indeed between 0 and 1.

In order to show this, we do the following trick. Since $s_i = 1 - \frac{1-P}{PQ(Q+1)}$, we also have $1 - s_i = \frac{1-P}{PQ(Q+1)}$ and so

$$\frac{s_i}{1 - s_i} = \frac{1 - \frac{1-P}{PQ(Q+1)}}{\frac{1-P}{PQ(Q+1)}}.$$

Summing the left-hand-side over $i = 1, \dots, n$ gives exactly Q , so we conclude that for the values of s_i that optimize f we have

$$Q = n \frac{PQ(Q+1) - (1-P)}{1-P}. \quad (2.12)$$

Let then $P = \frac{Q+n}{n(Q^2+Q+1)+Q}$, i.e. the value as indicated after we solve for P in (2.12).

Substituting this into (2.11), we get

$$\begin{aligned} s_i &= 1 - \frac{1 - \frac{Q+n}{n(Q^2+Q+1)+Q}}{\left(\frac{Q+n}{n(Q^2+Q+1)+Q}\right) Q(Q+1)} \\ &= 1 - \frac{n(Q^2 + Q + 1) + Q - (Q + n)}{(Q + n)Q(Q + 1)} \\ &= 1 - \frac{n(Q^2 + Q)}{(Q + n)(Q^2 + Q)} \\ &= 1 - \frac{n}{Q + n} \end{aligned} \quad (2.13)$$

Since we know that $n > 0$ and $Q > 0$, we immediately see that $0 < s_i < 1$ holds. We will show that when all values of s_i are the same, the competitive ratio is, in fact, maximized (rather than minimized) in the next section. ■

To conclude, Lemma 2.19 says that in order to determine the competitive ratio of our algorithm for general W-uniform case, it suffices to consider totally uniform instances. This is what we do in the next subsection.

2.3.3 Online Searching for Totally Uniform Swarms

We will now consider the Online Beachcombers' Problem when the swarm of robots is both W-uniform and S-uniform, i.e. All robots have identical walking speeds, w , and identical searching speeds, s , where $s < w$. The analysis in this section will lead to a proof for Theorem 2.17.

For the sake of notational ease, we normalize all speeds so as to have uniform walking speeds $w = 1$ and uniform searching speeds $s = \alpha$.

Following the analysis for W-uniform case (see (2.9)), we know that the competitive ratio of our algorithm for the totally uniform instance as described above is

$$\begin{aligned} C &\stackrel{(2.9)}{=} \sup_R \left(1 - \prod_{k=1}^n (1 - \alpha) \right) \left(1 + \frac{1}{\sum_{k=1}^n \frac{\alpha}{1-\alpha}} \right) \\ &= \sup_R \left(\frac{(\alpha(n-1) + 1)(1 - (1-\alpha)^n)}{\alpha n} \right) \end{aligned}$$

We denote the above expression on α, n by $f_n(\alpha)$.

$$f_n(\alpha) := \frac{(\alpha(n-1) + 1)(1 - (1-\alpha)^n)}{\alpha n} \quad (2.14)$$

From now on we think of $\max_{0 \leq \alpha \leq 1} f_n(\alpha)$ as the competitive ratio of the LeapFrog algorithm with n robots. Note that $\max_{0 \leq \alpha \leq 1} f_n(\alpha)$ is easy to determine for $n = 1, \dots, n$. For one robot, the competitive ratio is indeed 1. For $n \geq 2$, the maximum of $f_n(\alpha)$ when

TABLE 2.1: Competitive ratio of LeapFrog for collections of robots of size 2,3,4.

n	$\max_{\alpha} f_n(\alpha)$	$\operatorname{argmax}_{\alpha} f_n(\alpha)$
2	$\frac{9}{8} = 1.125$	$\frac{1}{2} = 0.5$
3	$\frac{172+7\sqrt{7}}{162} \approx 1.17605$	$\frac{5-\sqrt{7}}{6} \approx 0.392375$
4	≈ 1.20386	≈ 0.322472

$0 \leq \alpha \leq 1$ is achieved either when the derivative of $f_n(\alpha)$ is 0 (w.r.t. α), or when α is 0 or 1. Values 0, 1 can be easily eliminated as positions of global maxima. Hence, $\operatorname{argmax}_{0 \leq \alpha \leq 1} f_n(\alpha)$ can be computed by solving equation $\frac{d}{d\alpha} f_n(\alpha) = 0$, which is a degree n polynomial. For $n = 2, 3, 4$ we have closed formulas for the roots of the polynomials which are summarized in Table 2.1. Once we find $\operatorname{argmax}_{0 \leq \alpha \leq 1} f_n(\alpha)$, it is also easy to compute $\max_{0 \leq \alpha \leq 1} f_n(\alpha)$ for $n = 2, 3, 4$. In what follows we give a detailed analysis of the competitive ratio for $n \geq 5$ robots. For this we need the next technical lemma.

Lemma 2.20. *Let $\alpha_n = \operatorname{argmax} f_n(\alpha)$. Then $f_n(\alpha_n)$ increases with n .*

Proof. Note that Table 2.1 already shows the lemma for $n \leq 4$. Hence, below we may assume that $n \geq 5$. First we show that $f_n(a)$ has a unique maximizer when $\alpha \in (0, 1)$. For this we examine the critical points of $f_n(a)$ by solving $\frac{d}{d\alpha} f_n(\alpha) = 0$, i.e.

$$r_n(\alpha) := (1 - \alpha)^{n-1} (1 + \alpha(n - 1) + \alpha^2 n(n - 1)) - 1 = 0. \quad (2.15)$$

To show that $r_n(\alpha)$ has a unique solution in $(0, 1)$ we again take the derivative with respect to α to find that $\frac{d}{d\alpha} r_n(\alpha) = (1 - \alpha)^{n-2} \alpha n(n - 1)(1 - \alpha(n + 1))$. This means that $r_n(\alpha)$ increases when $\alpha < 1/(n + 1)$ and decreases otherwise. Noting also that $r_n(0) = 0$ and $r_n(1) = -1$, we conclude that $r_n(\alpha)$ has a unique root in $(0, 1)$, i.e. $f_n(\alpha)$ has a unique maximizer α_n over $\alpha \in (0, 1)$, and in particular $\alpha_n > \frac{1}{n+1}$. Next we

will provide a slightly better bound on the roots of $r_n(\alpha)$. For this we observe that

$$r_n\left(\frac{1}{n-1}\right) = \frac{\left(\frac{n-2}{n-1}\right)^n (3n-2)}{n-2} - 1$$

which can be seen to be positive for $n \geq 5$. Hence, by the monotonicity we have already shown for $r_n(\alpha)$, we may assume that its root α_n satisfies

$$\frac{1}{n-1} < \alpha_n < 1. \quad (2.16)$$

Now we turn our attention to $f_n(\alpha_n)$. Since α_n satisfies $r_n(\alpha_n) = 0$, it is easy to see that

$$f_n(\alpha_n) = \frac{(1 + \alpha_n(n-1))^2}{1 + \alpha_n(n-1) + \alpha_n^2 n(n-1)}$$

simply by substituting $(1 - \alpha)^{n-1}$ from (2.15) into (2.14). Recalling that α_n is also a function on n we get that

$$\frac{d f_n(\alpha_n)}{d n} = \frac{\alpha_n(1 - \alpha_n)}{(1 + \alpha_n(n-1) + \alpha_n^2 n(n-1))^2} (1 - \alpha_n^2(n-1)^2) \frac{d \alpha_n}{d n}$$

Due to (2.16) we conclude that $\frac{d f_n(\alpha_n)}{d n}$ has the opposite sign of $\frac{d \alpha_n}{d n}$, i.e. for $n \geq 5$ we have that $f_n(\alpha_n)$ increases with n if and only if α_n decreases with n . So it remains to show that the roots α_n of $r_n(\alpha) = 0$ decrease with n . Observe here that at this point we may restrict consideration to integral values of n .

To conclude the lemma we argue that $\alpha_{n+1} < \alpha_n$. For this we observe that

$$\frac{r_{n+1}(\alpha) + 1}{r_n(\alpha) + 1} = \frac{(1 - \alpha)(1 + \alpha n + \alpha^2 n(n+1))}{1 + \alpha(n-1) + \alpha^2 n(n-1)}$$

which is clearly less than 1 for $\alpha > \frac{1}{n+1}$ (just by solving for α). Effectively this means that the graph of $r_n(\alpha) + 1$ is above the graph of $r_{n+1}(\alpha) + 1$ for every $\alpha > \frac{1}{n+1}$, and

hence

$$r_n(\alpha_{n+1}) > r_{n+1}(\alpha_{n+1}) = 0 = r_n(\alpha_n) > r_{n+1}(\alpha_n).$$

But then, from the monotonicity we have shown for $r_n(\alpha)$ this implies that $\alpha_{n+1} < \alpha_n$ as wanted. ■

The next lemma in combination with Lemma 2.19 prove Theorem 2.17.

Lemma 2.21. *For the totally uniform instances of the Beachcombers' Problem, the LeapFrog algorithm has competitive ratio at most $9/8$ for two robots, and the competitive ratio of at most $\max_c \{(1 + 1/c)(1 - e^{-c})\} \approx 1.29843$ for any number of robots.*

Proof. By the proof of Lemma 2.20, α_2 satisfies $(1 - \alpha_2)(1 + \alpha_2 + 2\alpha_2^2) = 1$, which has the unique solution $\alpha_2 = 1/2$. It is easy to see that $f_2(1/2) = 9/8$.

Next, by Lemma 2.20, the bigger is the number of robots, the higher is the competitive ratio of our algorithm. Hence, we need to determine $\lim_{n \rightarrow \infty} f_n(\alpha_n)$.

To that end we note that if $\alpha_n = o(1/n)$, then $1 - (1 - \alpha)^n \approx \alpha n$, and so

$$\lim_{n \rightarrow \infty} \frac{(\alpha(n-1) + 1)(1 - (1 - \alpha)^n)}{\alpha n} = \lim_{n \rightarrow \infty} \frac{\alpha n - \alpha + 1}{\alpha n} \alpha n = 1.$$

Similarly, if $\alpha_n = \omega(1/n)$, then $1 - (1 - \alpha)^n$ tends to 1 as n goes to infinity. Consequently,

$$\lim_{n \rightarrow \infty} \frac{(\alpha(n-1) + 1)(1 - (1 - \alpha)^n)}{\alpha n} = \lim_{n \rightarrow \infty} \frac{\alpha n - \alpha + 1}{\alpha n} = 1.$$

It remains to check what happens when $\alpha = c/n$ for some $c \in \mathbb{R}_+$. But then

$$\lim_{n \rightarrow \infty} \frac{(\alpha(n-1) + 1)(1 - (1 - \alpha)^n)}{\alpha n} = (1 + 1/c)(1 - e^{-c}).$$

Taking the first derivative of the last expression gives $1 - e^{-c}(c^2 + c + 1)$. Setting this to zero and solving for c gives us $c \approx 1.79328$. The second derivative at $c \approx 1.79328$ is negative, and so we can say that the above limit is maximized when $c \approx 1.79328$ and the value it attains approaches 1.29843. ■

A picture for the rate of growth of the competitive ratio of the crawling algorithm is depicted in Figure 2.6.

FIGURE 2.6: Competitive ratio of LeapFrog as a function of the number of robots

2.4 The Multi-source Beachcombers' Problem

2.4.1 t -SBP as a Partitioning Problem and its Hardness

We begin our examination of t -SBP by considering the problem where robots are already partitioned into groups $A_1, \dots, A_t$. It is not difficult to see that (as in 1-SBP) we have the same necessary conditions of optimality, i.e. robots start searching at the same time, they all finish at the same time, etc. In other words, having fixed a partition, the

optimal solution can be found by solving t many 1-SBP instances. In particular, each group of robots will process a subinterval that is proportional to its search power.

Consider specifically the situation where a set of robots are partitioned into two groups, where each group is placed at the same mid-point (dependant of the relative search power of each group) and search towards opposing endpoints of the interval $[0, L]$. This observation states that the optimal strategy is for each group to each search their portion of the interval as if the problem was 1-SBP. This leads us to the following conjecture:

Conjecture 2.22. *The two-speed search problem on a closed fence is equivalent to the two-speed search problem on an open fence.*

Most importantly, the above observation reduces the problem of finding an optimal schedule for t -SBP into the problem of correctly guessing a partition of the swarm into t groups. With this in mind, one may recall the following family of well-studied problems [Chakravarty et al. \[1985b\]](#).

Definition 2.23 (The Set Partition Problem with Additive Objective (SPAO)).

Consider n robots, a function $g : 2^n \mapsto \mathbb{R}^+$, and a fixed integer t . In the *Set Partition Additive Objective* (SPAO) problem we are looking for a partition of the n robots into t disjoint sets $A_1, \dots, A_t$ that maximizes $\sum_{i=1}^t g(A_i)$.

Note that t -SBP problems are almost a generalization of SPAO, since our objects now have 2-dimensional attributes, i.e. the walking and the searching speeds. We can translate the Beachcombers' Problem into a partitioning problem resembling SPAO:

Observation 2.24 (t -SBP Revisited). Consider n items, each having two attributes s_i , w_i , with $s_i < w_i$. Define the evaluation function $g : 2^{[n]} \mapsto \mathbb{R}^+$ as the search power of subsets of the swarm with the same attributes (See Definition 2.10). Then, in the t -SBP problem we are looking for a partition of $[n]$ into t disjoint sets $A_1, \dots, A_t$ that maximizes $\sum_{i=1}^t g(A_i)$.

One may consider a greedy approach where we first order the items by non-decreasing w_i 's, and then assign robot r into set A_j , chosen from $A_1, \dots, A_t$, where $g(A_j)$ is minimum (before adding r). This idea is equivalent to scheduling a robot to the group which minimizes the amount it walks. Note that, as described by the solution to 1-SBP given in Lemma 2.7, adding item (robot) r to A_j , increases the value of the objective by

$$c_r = s_r \left(1 - \frac{1}{w_r} g(A_j) \right). \quad (2.17)$$

We now recall from the literature a combinatorial problem that plays a crucial role in our upper and lower bounds. In PRODUCTPARTITION we are given positive integers $m_1, \dots, m_n$ (with $m_i > 1$) and asked if there is a partition of them, A, B , such that $\prod_{i \in A} m_i = \prod_{i \in B} m_i$. PRODUCTPARTITION has a natural optimization version, MIN-MAXPRODUCTPARTITION, where we try to minimize $\max\{\prod_{i \in A} m_i, \prod_{i \in B} m_i\}$ over all partitions A, B of the n integers. As [Ng et al. \[2010\]](#) showed, this objective function can be well approximated, while the decision problem is hard. We use this to establish the hardness of 2-SBP for W -uniform swarms by reducing from PRODUCTPARTITION.

Theorem 2.25. *It is NP-hard to solve 2-SBP with a W -uniform swarm.*

Proof. From the hardness result of PRODUCTPARTITION, it suffices to reduce from PRODUCTPARTITION. Our reduction considers a W -uniform swarm and sets the searching speeds as $s_i = 1 - \frac{1}{m_i}$. We argue that if we can solve 2-SBP in polynomial time, then we can decide PRODUCTPARTITION in polynomial time as well.

Given a partition A, B of the W -uniform swarm, we know the best schedule has cost

$$2 - \prod_{i \in A} (1 - s_i) - \prod_{i \in B} (1 - s_i),$$

given by Definition 2.10 (Search Power) and Claim 2.18. Hence, optimizing the finishing time for this instance of 2-SBP is equivalent to finding a partition A, B so as to minimize

$$\prod_{i \in A} (1 - s_i) + \prod_{i \in B} (1 - s_i).$$

Since $\prod_{i \in A \cup B} (1 - s_i)$ is invariant for all partitions, the problem can be translated into minimizing

$$\left| \prod_{i \in A} (1 - s_i) - \prod_{i \in B} (1 - s_i) \right| = \left| \prod_{i \in A} \frac{1}{m_i} - \prod_{i \in B} \frac{1}{m_i} \right|.$$

Thus, optimizing the schedule of the instance finds a partition that solves PRODUCTPARTITION, if one exists. ■

Theorem 2.26. *W-Uniform 2-SBP instances $s_1, \dots, s_n$ for which $\frac{1}{1-s_i} \in \mathbb{N}$, admit a $I/(1+\epsilon^2)$ -approximation which requires $O(\frac{n^3 \log m}{\epsilon})$ steps, where $m = \max(1/(1-s_i))$.*

Proof. The reader might expect that we will use the FPTAS for MINMAXPRODUCTPARTITION as a subroutine. Actually, we do so in the crudest possible way! Given a swarm with respective searching speeds $s_1, \dots, s_n$ with $s_i = 1 - \frac{1}{m_i}$, $m_i \in \mathbb{N}$, we run the FPTAS for MINMAXPRODUCTPARTITION for input $m_1, \dots, m_n$, to obtain a partition $A \subseteq \{1, \dots, n\}$. Let this be an $\alpha := 1 + \epsilon$ approximation to the MINMAXPRODUCTPARTITION problem.

We use this partition to split the robots between the two endpoints, and we schedule each group optimally as described in the solution from Section 2.2. For the ease of notation, we set $m^A := \prod_{i \in A} m_i$ and $M := \prod_{i=1}^n m_i$.

Instead of maximizing the combined searching speed of the groups, we fix the segment to be explored as unit length and (equivalently) wish to minimize the finishing time of the search schedules. Therefore, the cost T of our proposed algorithm and the cost of

the optimal solution T_{OPT} are

$$T = \left(2 - \frac{1}{m^A} - \frac{m^A}{M}\right)^{-1} \quad \text{and} \quad T_{\text{OPT}} = \left(2 - \frac{1}{m^D} - \frac{m^D}{M}\right)^{-1}, \quad (2.18)$$

where D is the optimal underlying partition of the best scheduling. We claim that $T/T_{\text{OPT}} \leq 1 + \epsilon^2$.

In order to break the symmetry, we assume that

$$m^A \geq \sqrt{M} \quad \text{and} \quad m^D \geq \sqrt{M}. \quad (2.19)$$

The key observation towards concluding our claim is that the partition D which minimizes

$$\left(2 - \frac{1}{m^D} - \frac{m^D}{M}\right)^{-1}$$

is the same as the one which minimizes

$$\max\{m^D, \frac{M}{m^D}\} \stackrel{(2.19)}{=} m^D.$$

Since we run the FPTAS for MINMAXPRODUCTPARTITION, the cost of the solution was $\max\{m^A, \frac{M}{m^A}\} \stackrel{(2.19)}{=} m^A$, and in particular

$$m^A \leq \alpha m^D. \quad (2.20)$$

We are now ready to compute the approximation guarantee of our proposed algorithm:

$$\begin{aligned} \frac{T}{T_{\text{OPT}}} &\stackrel{(2.18)}{=} \frac{2 - \frac{1}{m^D} - \frac{m^D}{M}}{2 - \frac{1}{m^A} - \frac{m^A}{M}} \stackrel{(2.20), (2.19)}{\leq} \frac{2 - \frac{1}{m^D} - \frac{m^D}{M}}{2 - \frac{1}{\alpha m^D} - \frac{\alpha m^D}{M}} \\ &\stackrel{(2.19)}{\leq} \frac{2 - \frac{2}{m^D}}{2 - \frac{1}{\alpha m^D} - \frac{\alpha}{m^D}} = \frac{2m^D - 2}{2m^D - \left(\frac{1}{\alpha} + \alpha\right)} \leq \frac{2}{4 - \left(\frac{1}{\alpha} + \alpha\right)}, \end{aligned}$$

where the last inequality is due to the fact that the ratio decreases with D . Plugging in $\alpha = 1 + \epsilon$ it is easy to see then that last ratio is no more than $1 + \epsilon^2/2$.

■

2.4.1.1 A Deterministic Approximation Guarantee for 2-SBP

In this section we show that the solution to 1-SBP is a trivial $\frac{1}{2}$ -approximation for 2-SBP, and present a deterministic 0.5569-approximation for 2-SBP. The algorithm is based on the optimal solution to 1-SBP and a shifting mechanism that trades the walking time of robots for longer searching times. As before, we assume that the interval to be searched is $[0,1]$. If this is not the case, the solution is easily scaled.

Consider the optimal solution to 1-SBP in which robots $1, \dots, n$ are ordered according to their respective search intervals $c_1, \dots, c_n$, where $\sum_{i=1}^n c_i = 1$. We denote by $d_1, \dots, d_n$ distances in the optimal solution that robots walk towards their respective search intervals. In particular, $d_1 = 0$, and $d_i = \sum_{j=1}^{i-1} |c_j|$, for all $1 < i \leq n$.

Lemma 2.27. *The robots $1, \dots, i$ with search intervals $c_1, \dots, c_i$ and which search a combined interval of length d_{i+1} in the optimal solution to 1-SBP cannot search an interval of length greater than $d_i + d_{i+1}$ in 2-SBP.*

Proof. Follows from the fact that even if robots $i-1$ and i operate at different ends of the searched segment in 2-SBP the total contribution at the respective ends cannot exceed values d_i and d_{i+1} . Otherwise these robots would have to be preceded by some extra robots and that would violate the property of the optimal solution in which participating robots must be ordered according to their walking speeds. ■

Corollary 2.28. *The solution for 1-SBP provides a $\frac{1}{2}$ -approximation for 2-SBP.*

Proof. Follows immediately from Lemma 2.27 when considering all robots. ■

Consider any robot i . In the optimal solution to 1-SBP, robot i walks a fixed distance d_i from the front (left end) of the searched segment to its search interval of length x_i . We observe that the length of the searched interval x_i could be longer if i walked a shorter distance. In particular, if i walked distance $\beta \cdot d_i$ instead, for some constant $0 < \beta < 1$, the searched interval can be extended by $y_i = \frac{d_i}{w_i}(1 - \beta)s_i$, where $\frac{d_i}{w_i}$ refers to the walking time of i in the optimal solution to 1-SBP. Since w_i , s_i and d_i are fixed, the extension y_i is a linear function of β . We say that a group X of searched intervals is subject to β -expansion if for each $x_i \in X$, the respective robot i walks a distance at most βd_i from the front of the searched segment. The total gain (extra search distance) of β -expansion is defined as $\sum_{x_i \in X} y_i$. This is also a linear function of β .

We now present our 0.5569-approximation algorithm to 2-SBP, which we will call Swap-and-Expand. For convenience, we have the two groups of robots walking in opposing directions starting from the two endpoints of the interval. The proof of correctness of our algorithm completes the proof of Theorem 2.1.

Algorithm 2.4: Swap-and-Expand

Input: Set of robots R with associated walking and patrolling speeds

Output: Schedule of the swarm

- 1 Sort the robots in non-decreasing walking speed
 - 2 Calculate the search intervals $(c_1, \dots, c_n)$ for Algorithm Comb
 - 3 Find i such that $\sum_{k=1}^{i-1} c_k < 1/2 \leq \sum_{k=1}^i c_k$
 - 4 Assign robots $1 \dots i$ to group A_0 and robots $i + 1 \dots n$ to group A_1
 - 5 Place group A_0 at point 0 with direction *right*
 - 6 Place group A_1 at point 1 with direction *left*
 - 7 Execute Comb with the robots in A_0 , and concurrently with the robots in A_1
-

The algorithm is simple: either moving the robots leads to a significant improvement (large β -expansion), or it does not. In the latter case, we can show that the optimal

solution for 1-SBP is better than a $\frac{1}{2}$ -approximation. We observe that the output from this algorithm solves 2-SBP at least as well as simply using the solution for 1-SBP.

Lemma 2.29. *Swap-and-Expand gives an α -approximation for 2-SBP, where $\alpha = \frac{1}{8} (13 - \sqrt{73}) > 0.5569$.*

Proof. For some $0 \leq \delta < 1/2$, consider two cases referring to the total gain of a $(1/2 + \delta)$ -expansion of A_1 : **Case 1** where the total gain is $\leq \delta$ and the complementary **Case 2**.

Case 1 In this case the total gain of $(1/2 + \delta)$ -expansion is at most δ . We first observe that in this case (the best possible) 0-expansion would give the total gain $\leq \frac{\delta}{1/2 - \delta}$ (follows from linearity of the expansion mechanism). In other words, even if all intervals in A_1 were moved to the beginning of the searched segment (i.e., respective robots did not have to walk at all) the total gain would not exceed $\frac{\delta}{1/2 - \delta}$. Concluding, one can observe that the total length of the search intervals in any 2-SBP solution would not exceed $d_i + d_{i+1} + \sum_{j=i+1}^n |c_j| + \frac{\delta}{1/2 - \delta}$. This is equal to $d_i + 1 + \frac{\delta}{1/2 - \delta}$, and since $d_i < 1/2$ we get the bound $\frac{1}{\frac{3}{2} + \frac{\delta}{1/2 - \delta}} = \frac{2 - 4\delta}{3 - 2\delta}$.

Case 2 In this case the total gain of the $(1/2 + \delta)$ -expansion is larger than δ . This gain will be used by the approximation algorithm in the 2-SBP setting where one is allowed to allocate robots with their search intervals at the two ends of the searched segment. We propose the following solution. From the optimal solution to 1-SBP we remove intervals $c_1, \dots, c_i$. These will be eventually placed in the reversed order at the right end of the searched segment in the 2-SBP setting. This removal leaves a gap of size d_{i+1} at the left end of the searched segment. We shift all remaining intervals (without changing their order) in A_1 maximally towards the left end of the searched segment avoiding overlaps.

(a) If all segments in A_1 experience $(1/2 + \delta)$ -expansion, i.e., if each $c_i \in A_1$ shortens its distance d_i to the left end of the searched segment to $\leq d_i \cdot (1/2 + \delta)$ the total gain is $\geq \delta$. In this case the interval searched in time t (the optimal time for 1-SBP) in 2-SBP is of length at least $1 + \delta$. Since the optimal solution in 1-SBP gives $\frac{1}{2}$ -approximation in 2-SBP, we have the approximation ratio $\frac{1+\delta}{2}$.

(b) Otherwise, if at least one segment (take the one with the smallest index k) $c_k \in A_1$ does not experience $(1/2 + \delta)$ -expansion, its left end is at distance $> d_k \cdot (1/2 + \delta)$ from the left end of the searched segment. In this case the total gain obtained by $c_{i+1}, \dots, c_{k-1}$ is $\geq d_k \cdot (1/2 + \delta) - \sum_{j=i+1}^{k-1} |c_j|$. Now since $\sum_{j=i+1}^{k-1} |c_j| \leq d_k - 1/2$ (since c_i contains the point $1/2$) we conclude that the total gain is $\geq d_k \cdot (1/2 + \delta) - d_k + 1/2 = d_k \cdot (\delta - 1/2) + 1/2$. Since $d_k \leq 1$ the total gain is at least δ . As in **Case 2a**, we get the approximation ratio $\frac{1+\delta}{2}$.

To this point, we have said nothing about the value of δ . However, given an arbitrary swarm either case may hold. Therefore, we want to choose a value of δ which maximizes $\min\left(\frac{2-4\delta}{3-2\delta}, \frac{1+\delta}{2}\right)$. Since the first function is monotonic decreasing (with respect to δ) and the second is monotonic increasing, $\min\left(\frac{2-4\delta}{3-2\delta}, \frac{1+\delta}{2}\right)$ is maximized when $\frac{2-4\delta}{3-2\delta} = \frac{1+\delta}{2}$. We solve this for δ and find its roots, only one of which falls in the interval $[0, 1]$, when $\delta = \frac{9-\sqrt{73}}{4}$, from which we obtain an approximation ratio bounded below by $\frac{1}{8}(13 - \sqrt{73}) \geq 0.5569$. ■

2.4.1.2 An Exact Solution for t -SBP with S -uniform Swarms

We now present GreedyPartition (Algorithm 2.5) and show that it solves instances of t -SBP with S -uniform swarms in $O(n \log n)$ many steps (as stated in Theorem 2.2). We assume that the interval to be searched is $[0,1]$. Again, if this is not the case, the solution is easily scaled.

GreedyPartition works by sorting the robots, separating them into groups in a revolving manner, calculating the search power of the groups, and placing the groups at positions on the interval so that by executing the algorithm for 1-SBP, each group finishes searching their respective subintervals at the same time.

Algorithm 2.5: GreedyPartition

Input: swarm (s, w) , integer t

Output: Schedule of the swarm for t -SBP

```

1 Sort the robots in non-decreasing walking speeds
2 for  $i = 1, \dots, n$  do
3   | Assign robot  $i$  to group  $A_{(i \bmod t)}$ 
4 end
5 for  $i = 1, \dots, t$  do
6   |  $x_j \leftarrow \sum_{k=1}^{|A_i|} \prod_{j=k+1}^{|A_i|} \left(1 - \frac{1}{w_j^{A_i}}\right)$ 
7 end
8 for  $i = 1, \dots, t$  do
9   | Place group  $A_i$  at point  $\frac{\sum_{k=1}^{i-1} x_k}{\sum_{k=1}^t x_k}$  with search direction right
10 end
11 for  $i = 1, \dots, t$  do
12   | Execute Comb with the robots in  $A_i$ 
13 end

```

Proof of Theorem 2.2. It is clear that the algorithm runs in $O(n \log n)$ time - the time required to sort the robots. We note that the robots are passed in sorted order to Comb, so that it runs in $O(1)$ time.

It now remains to show that GreedyPartition is optimal. We begin by noting that no two of the t slowest walking robots will be placed in the same group, and therefore none of them will walk at all. Since they all search at the same rate, they are interchangeable and any solution which has any permutation of the first t robots, but identical otherwise will be equivalent to the solution provided by Algorithm 2.5.

Assume then, that there exists an optimal solution which is not equivalent to the solution provided by GreedyPartition. In this case, we can find two robots $a \in A_j, b \in A_k$

such that $\omega_a < \omega_b$ and $w_a > w_b$ where ω_i the interval walked by robot i . In other words, there exist two robots where the faster walker walks a shorter distance than the slower one. We first observe that:

Lemma 2.30. *Swapping a and b increases the length searched by a and b combined.*

Proof. Since both robots have identical searching speeds, we can simply consider the total time walking - as the length of the combined interval searched will be maximized when the combined time walking is minimized. Both before and after the swap, each robot must walk at least ω_a . Before the swap, a begins searching immediately while b must walk a further $\omega_b - \omega_a$ before it can start searching. So the total time spent walking by both robots (again ignoring the first interval) is $\frac{\omega_a}{w_a} + \frac{\omega_a}{w_b} + \frac{\omega_b - \omega_a}{w_b}$. Similarly, the total time spent walking by both robots after the swap is $\frac{\omega_a}{w_a} + \frac{\omega_a}{w_b} + \frac{\omega_b - \omega_a}{w_a}$. Since $w_a > w_b$, the total time walking is less after the swap. ■

FIGURE 2.7: Effect of swapping a and b on outer intervals

Therefore, swapping a and b must negatively affect other robots, otherwise we have reached a contradiction. Since it does not affect any robots that search intervals closer to their source points than a and b , then there exist robots which search intervals farther from their source points than the intervals searched by a and b . We can assume that the robots searching these outer intervals are arranged optimally, as described by

GreedyPartition. If they are not, then there exist two robots farther from their source points than a and b which have the same properties as a and b , and so we can consider them instead.

We now consider how swapping a and b affects the outer intervals searched by the remaining robots. We denote the intervals searched by robot i before and after the swap as c_i and c'_i , respectively. We assume that $\omega_a + x_a < \omega_b + x_b$. If this is not true, then we also swap all robots which search intervals after a and b , and the rest of the following argument will hold. After the swap, the robots which originally started after a will walk less than they did before the swap. Similarly, the robots after b will walk more than they did before the swap. We note that $a + 1$ walked a shorter distance than $b + 1$ before the swap and so we say that this swap *unbalances* the outer intervals, as the ones that were closer to their start point before the swap become even closer at the expense of the others, which move further away from their start point by a corresponding amount.

Lemma 2.31. *Unbalancing the outer intervals improves the search time.*

Proof. Suppose that the intervals are unbalanced by $\epsilon > 0$. Consider the robots $a + 1$ and $b + 1$ which originally follow both a and b . We observe that as a pair, after the swap, the time it takes them to search intervals of the same combined distance as their original intervals improves by $\frac{\epsilon}{w_{a+1}} - \frac{\epsilon}{w_{b+1}}$. Since $w_{a+1} < w_{b+1}$ we know this is a positive amount. Similarly, we can show that the pairs $(a + 2, b + 2), \dots, (a + k, b + k)$ all improve their search time by the imbalance. We know that there must be at least as many robots which follow a as there are that follow b - since the distribution of these robots follows Comb.

A careful examination of the swap of a and b will reveal that it does not create a strict imbalance as described above - the one set of intervals shift closer to their start point less than the other set shifts farther from theirs. This shift also has the effect of inflating the

intervals shifted closer, so that the first few intervals will be shifted more than the last few. These effects may at first seem to mean that the above argument cannot be applied, however we note that the difference in the amount that the intervals shift is exactly the amount that was gained by swapping a and b . Therefore, if we first consider the swap without the increased search efficiency, we can see that the imbalance improves the search time. If we then add in the improvement, it will again improve the search time.

■

We have therefore shown that no matter where a and b are positioned, we can improve the solution. This contradicts our assumption, and so GreedyPartition is optimal.

■

2.4.2 A Randomized Algorithm for t -SBP

In this section, we propose an oblivious randomized algorithm for t -SBP. We fix some input to the problem, i.e. a swarm with attributes $(s_1, w_1), \dots, (s_n, w_n)$. For the sake of our analysis, we assume that the swarm is ordered in non-decreasing walking speeds w_i . Denote the optimal solution to t -SBP for the above instance by $\text{OPT}_t(\mathbf{s}, \mathbf{w})$. We propose the next randomized algorithm. We note that ObliviousRandomized is identical to GreedyPartition except for how the robots are partitioned into groups.

Lemma 2.32. *The expected value of ObliviousRandomized for t -SBP on input $(\mathbf{s}, \mathbf{w})$ equals $t \cdot \text{OPT}_1(s/t, \mathbf{w})$.*

Proof. ObliviousRandomized assigns each robot to one of the t groups uniformly at random. Denote the induced partition by $A_1, A_2, \dots, A_t$. By symmetry, and due to linearity of expectation, the expected value of ObliviousRandomized is t times the expected searched interval of the group A_1 .

Algorithm 2.6: ObliviousRandomized**Input:** swarm (s, w) , integer t and $O(n \log t)$ random bits**Output:** Schedule of the swarm for t -SBP

```

1 for  $i = 1, \dots, n$  do
2   | Choose  $k$  independently and uniformly at random from  $1, \dots, t$ 
3   | Assign robot  $i$  to group  $A_k$ 
4 end
5 for  $i = 1, \dots, t$  do
6   |  $x_j \leftarrow \sum_{k=1}^{|A_i|} \prod_{j=k+1}^{|A_i|} \left(1 - \frac{1}{w_j^{A_i}}\right)$ 
7 end
8 for  $i = 1, \dots, t$  do
9   | Place group  $A_i$  at point  $\frac{\sum_{k=1}^{i-1} x_k}{\sum_{k=1}^t x_k}$  with search direction right
10 end
11 for  $i = 1, \dots, t$  do
12   | Execute Comb with the robots in  $A_i$ 
13 end

```

Let now B_r denote the random variable that equals the search power of group A_1 , when we only use the first r robots with respect to their walking speeds. Clearly, $\mathbb{E}[B_n]$ is the expected search power of A_1 , and due to the short discussion above, the expected value of ObliviousRandomized is $t \cdot \mathbb{E}[B_n]$.

In order to compute $\mathbb{E}[B_n]$ we set up a recursion. Using the law of total probability, we observe that

$$\mathbb{E}[B_n] = \mathbb{P}[n \in A_1] \mathbb{E}[B_n \mid n \in A_1] + \mathbb{P}[n \notin A_1] \mathbb{E}[B_n \mid n \notin A_1] \quad (2.21)$$

$$= \frac{1}{t} \mathbb{E}[B_n \mid n \in A_1] + \frac{t-1}{t} \mathbb{E}[B_n \mid n \notin A_1] \quad (2.22)$$

$$= \frac{1}{t} \mathbb{E} \left[B_{n-1} + s_n \left(1 - \frac{1}{w_n} B_{n-1} \right) \right] + \frac{t-1}{t} \mathbb{E}[B_{n-1}] \quad (2.23)$$

$$= \frac{1}{t} s_n + \left(1 - \frac{s_n}{t w_n} \right) \mathbb{E}[B_{n-1}], \quad (2.24)$$

where line (2.21) is due to the fact that the fastest walking robot is either in or outside A_1 , line (2.22) is from the fact that robots are assigned to groups uniformly at random,

line (2.23) is due to (2.17) and finally (2.24) is because of linearity of expectation.

We conclude the lemma by recalling that if X_n denotes the optimal solution for one group when we use the first n robots, then $X_n = s_n + \left(1 - \frac{s_n}{w_n}\right) X_{n-1}$ (as it can be easily derived from the closed form solution given in Section 2.2). Hence, $\mathbb{E}[B_n]$ satisfies the exact same recurrence, but searching speeds are scaled by t . ■

By Lemma 2.32, the expected performance of `ObliviousRandomized` is exactly

$$\frac{t \cdot \text{OPT}_1\left(\frac{\mathbf{s}}{t}, \mathbf{w}\right)}{\text{OPT}_t(\mathbf{s}, \mathbf{w})}, \quad (2.25)$$

from which we obtain naive bounds of

$$\text{OPT}_1\left(\frac{\mathbf{s}}{t}, \mathbf{w}\right) \geq \frac{1}{t} \text{OPT}_1(\mathbf{s}, \mathbf{w}),$$

and that

$$\text{OPT}_t(\mathbf{s}, \mathbf{w}) \leq t \cdot \text{OPT}_1(\mathbf{s}, \mathbf{w}).$$

The approximation ratio is then $1/t$, which is the ratio we achieve if we assign all robots to one group. The expected performance should be better, given that we assign robots to random groups. The crux of our analysis is based on the trade-offs between the bounds we have for $\text{OPT}_1\left(\frac{\mathbf{s}}{t}, \mathbf{w}\right)$ and $\text{OPT}_t(\mathbf{s}, \mathbf{w})$, with respect to $\text{OPT}_1(\mathbf{s}, \mathbf{w})$. We now show a restatement of Theorem 2.3.

Theorem 2.33. *For the t -SBP problem on instance $(\mathbf{s}, \mathbf{w})$, `ObliviousRandomized` outputs (in expectation) a schedule of combined search power at least*

$$\left(1 - (1 - 1/t)^t\right) \text{OPT}_t(\mathbf{s}, \mathbf{w}).$$

The trick in our analysis is to first find the worst configuration of swarms that minimizes (2.25) over all swarms that satisfy $\text{OPT}_t(\mathbf{s}, \mathbf{w}) = \alpha \cdot \text{OPT}_1(\mathbf{s}, \mathbf{w})$ for some $1 \leq \alpha \leq n$. Effectively, this determines the worst approximation ratio, call it $R(\alpha)$, for the Oblivious Randomized Algorithm, with the restriction that $\text{OPT}_t(\mathbf{s}, \mathbf{w}) = \alpha \cdot \text{OPT}_1(\mathbf{s}, \mathbf{w})$. Finding $\max_{1 \leq \alpha \leq n} R(\alpha)$ will then determine the desired bound on the approximation ratio (as described in Theorem 2.33). In what follows, we focus on estimating $R(\alpha)$.

Lemma 2.34. *The performance of ObliviousRandomized, $R(\alpha)$, is worst when for each robot i , we have $s_i \approx w_i$.*

Proof. Since $t \geq 2$, there exists a group in $\text{OPT}_t(\mathbf{s}, \mathbf{w})$ in which a robot r does not walk (only searches), but walks in $\text{OPT}_1(\mathbf{s}, \mathbf{w})$. Note that the value w_r appears only the numerator of (2.25). Since in particular we have

$$\text{OPT}_1\left(\frac{\mathbf{s}}{t}, \mathbf{w}\right) = \sum_{k=1}^n \frac{s_k}{t} \prod_{j=k+1}^n \left(1 - \frac{s_j}{tw_j}\right),$$

it is clear that the ratio is minimized when w_r is minimized. On the other hand, $w_r > s_r$, and hence when the approximation ratio is minimized, w_r is infinitesimally close to s_r .

Given that $s_r \approx w_r$, we observe then that $\text{OPT}_1(\mathbf{s}, \mathbf{w}) \approx s_n$, where n is the index of the robot with the fastest walking speed. Effectively, this shows that the approximation ratio we want lower bound is

$$\frac{t \cdot \text{OPT}_1\left(\frac{\mathbf{s}}{t}, \mathbf{w}\right)}{\text{OPT}_t(\mathbf{s}, \mathbf{w})} \approx \frac{t \cdot \text{OPT}_1\left(\frac{\mathbf{s}}{t}, \mathbf{w}\right)}{\alpha \cdot s_n},$$

with the restriction that $\text{OPT}_t(\mathbf{s}, \mathbf{w}) = \alpha \cdot s_n$. But, we further observe that the rest of the walking speeds $w_2, \dots, w_n$ appear only in the numerator of the ratio we want to bound, and (identically to our argument for robot r), the ratio is smallest when the w_i attain as

small values as possible. This shows that $s_i \approx w_i$ for $i = 2, \dots, n$, while the walking speed w_1 of robot 1 is irrelevant to $\text{OPT}_1(\mathbf{s}, \mathbf{w})$. ■

We can now express $R(\alpha)$ as the minimum of a relatively easy rational function, after which we will have all we need to prove Theorem 2.3.

Lemma 2.35. *The performance $R(\alpha)$ of ObliviousRandomized for t -SBP is minimized by a swarm with searching speeds $s_1 \leq s_2 \leq \dots \leq s_t$. Moreover, the value $R(\alpha)$ is given as the infimum of the following optimization problem*

$$\min_{s_1, \dots, s_t} \frac{\sum_{k=1}^t s_k \left(1 - \frac{1}{t}\right)^{t-k}}{\alpha \cdot s_t} \quad \text{subject to} \quad \sum_{k=1}^t s_k = \alpha \cdot s_t$$

Proof. For a swarm $(\mathbf{s}, \mathbf{w})$ that minimizes $R(\alpha)$, let $i_1, \dots, i_t$ be the indices of robots that search the last intervals (i.e. start searching the latest in time) for the t different groups of $\text{OPT}_t(\mathbf{s}, \mathbf{w})$. Below we argue that the exact same robots are scheduled last when computing $\text{OPT}_1\left(\frac{\mathbf{s}}{t}, \mathbf{w}\right)$.

Since robots in each group are always ordered in non-decreasing walking speeds, it also follows that each of the robots $i_1, \dots, i_t$ is the fastest in each of the t groups. By Claim 2.34, we know that we may assume that all walking speeds almost coincide with the searching speeds, and hence

$$\text{OPT}_t(\mathbf{s}, \mathbf{w}) \approx \sum_{s=1}^t s_{i_s}. \quad (2.26)$$

However, by Claim 2.34 for the configuration that determines $R(\alpha)$ we also have

$$t \cdot \text{OPT}_1\left(\frac{\mathbf{s}}{t}, \mathbf{w}\right) \approx \sum_{k=1}^n s_k \left(1 - \frac{1}{t}\right)^{n-k}. \quad (2.27)$$

This immediately implies that $R(\alpha)$ is given as the ratio of expression (2.27) over expression (2.26), with the restriction that $\sum_{s=1}^t s_{i_s} = \alpha \cdot s_n$. Recall that this corresponds to the ratio for a swarm where the searching speed for every robot is infinitesimally close to its walking speed. Also, (2.26) not only gives the value for t -SBP, but also tells us that the robots i_s , $s = 1, \dots, t$ must be the fastest robots in the swarm with respect to their walking speeds (that coincide with the searching speeds). Otherwise, we could improve the value $\text{OPT}_t(s, w)$. But then, the schedule we have for $\text{OPT}_1(\frac{s}{t}, w)$ should also schedule the exact same t robots to search the last t intervals, meaning that the last t robots in the sum (2.27) must be $i_1, i_2, \dots, i_t$ (and without loss of generality, as this does not affect the optimal value of t -SBP, we may assume that $s_{i_1} \leq s_{i_2} \leq \dots \leq s_{i_t}$). But then, just ignoring the contribution of the first $n - t$ robots, we have that

$$\sum_{k=1}^n s_k \left(1 - \frac{1}{t}\right)^{n-k} \geq \sum_{k=n-t+1}^n s_k \left(1 - \frac{1}{t}\right)^{n-k} = \sum_{s=1}^t s_{i_s} \left(1 - \frac{1}{t}\right)^{t-s} \quad (2.28)$$

Combining now (2.28) with (2.27) shows the lemma. ■

Theorem 2.3. When $\text{OPT}_t(s, w) = \alpha \cdot \text{OPT}_1(s, w)$, the expected performance $R(\alpha)$ of ObliviousRandomized is given by Lemma 2.35, which we rewrite as

$$\sum_{k=1}^{t-1} \frac{s_k}{s_t} = \alpha - 1 \quad (2.29)$$

while also, the expected performance is the infimum of the expression

$$\frac{\sum_{k=1}^t s_k \left(1 - \frac{1}{t}\right)^{t-k}}{\alpha \cdot s_t} = \frac{1}{\alpha} \cdot \sum_{k=1}^{t-1} \frac{s_k}{s_t} \left(1 - \frac{1}{t}\right)^{t-k} + \frac{1}{\alpha} \quad (2.30)$$

We note that the value of the above expression is invariant under scaling, so for convenience we may set $s_t = 1$. Next, recall that according to Claim 2.35, the searching speeds satisfy $s_1 \leq s_2 \leq \dots \leq s_{t-1} \leq 1$. Since by (2.29) the speeds s_i sum up to $\alpha - 1$,

and observing the monotonicity of the coefficients of s_i , one can easily show that the sum (2.30) is minimized when $s_1 = s_2 = \dots = s_{t-1} = \frac{\alpha-1}{t-1}$, which is compatible with our restrictions, since $\alpha \leq t$. Then, we have

$$\begin{aligned} R(\alpha) &\stackrel{(2.30)}{=} \frac{1}{\alpha} \left(\frac{\alpha-1}{t-1} \sum_{k=1}^{t-1} \left(1 - \frac{1}{t}\right)^{t-k} + 1 \right) \\ &= \frac{1}{\alpha} \left(\frac{\alpha-1}{t-1} \left(t-1 - t \left(1 - \frac{1}{t}\right)^t \right) + 1 \right) \end{aligned} \quad (2.31)$$

A final observation is that expression (2.31) is decreasing in α (for every fixed t). Since α is no more than t (recall that α measures how much more OPT_t is compared to OPT_1), we conclude that the expected approximation ratio of `ObliviousRandomized` is $\min_{\alpha} R(\alpha) = R(t)$. Substituting $\alpha = t$ in (2.31) gives us what the Theorem claims. ■

2.5 Conclusion and Open Problems

We proposed and analysed offline and online algorithms for addressing the Beachcombers' Problem.

The offline algorithm, `Comb`, is shown to produce the optimal schedule when the size of the segment to search is known in advance. An interested reader may observe that, if all walking speeds are different, then `Comb` is the only one achieving the optimal speed.

The online searching algorithm, `LeapFrog`, is shown to be 2-competitive in general case and 1.29843-competitive when the robots' walking speeds are identical.

For the Multi-Source Beachcombers' Problem, we observe that some (and perhaps all) instances with a W-uniform swarm are NP-hard. However, all instances with S-uniform swarms are solvable in polynomial time.

We then showed that the solution for 1-SBP is a $\frac{1}{2}$ -approximation for 2-SBP, and proposed `Swap-and-Expand` and proved that it is a deterministic 0.5569-approximation for 2-SBP which requires $O(n \log n)$ computational steps.

Finally, we focused on general t -SBP problems and proposed `ObliviousRandomized` a simple randomized algorithm with expected approximation ratio $1 - \left(1 - \frac{1}{t}\right)^t$ which needs $O(n \log t)$ random bits, and runs in $O(n \log n)$ many steps.

Chapter 3

Patrolling

3.1 Introduction

Fence patrolling is the act of perpetual monitoring of a domain, modelled by a unit segment, by a set of mobile robots. As the robots cannot continuously monitor all points of the domain, a standard efficiency measure of patrolling is the notion of *idleness* - the size of the minimal time interval for which all points of the domain are always visited (independently of the start of this interval). Patrolling algorithms attempt to produce the schedules, (i.e. the trajectories of the robots in time) minimizing idleness.

In previous research, the robots were supposed to have either the same or different maximal speeds of their patrolling movements. In this chapter we assume that the patrolling activity requires more elaborate work than just walking within the domain; this is for example the case when foraging or harvesting which takes longer than walking. In computer science *web page indexing, forensic search, code inspection, packet sniffing*

¹This chapter contains results from the submitted paper: J. Czyzowicz, K. Georgiou, E. Kranakis, F. MacQuarrie and D. Pajak, *Fence Patrolling with Two-Speed Robots*

requires a more involved inspection. Consequently, we assume the maximal patrolling speeds to be strictly smaller than the maximal walking speeds.

We suppose that each robot is capable of patrolling in only one direction (arbitrarily chosen) of the segment while walking is permitted in both directions. It is worth noting that the fence problem for robots patrolling in both directions appears to be quite difficult. Indeed, optimal idleness algorithms were found only for fence patrolling of up to three robots, (cf. [Czyzowicz et al. \[2011\]](#), [Dumitrescu et al. \[2014\]](#), [Kawamura and Kobayashi \[2012\]](#)). Such one-way patrolling assumption is natural for humans, e.g. in text inspection (cf. the difficulty of the backward spelling game). Several mechanical devices operate actively in one direction only, e.g. combine and forage harvesters, snowplowers, etc. Also travelling up the hill, swimming against river flow, carrying heavy load - all these actions require slower operating speeds.

We solve optimally the patrolling problem in the centralized scenario. We also show how to apply our algorithm to natural optimization problem in transportation concerning *Scheduling with Regular Delivery*. Next, and more surprisingly, using a very weak collection of mobile robots with only primitive capabilities, we are able to design a distributed strategy converging to the same optimal solution thus achieving the same idleness as the optimal centralized algorithm. Our robots are anonymous, oblivious, silent (no communication permitted) and they cannot process any information, as well as they are not aware of their numbers or their speeds. They are subject to *passive mobility*, i.e. their movement is dynamically controlled by the contacts with the environment. The robots are only capable of walking or patrolling with maximal speed and their perception mechanism permits only to recognize a collision with the environment (i.e. the endpoints of the segment or another robot). This is the first study of the patrolling problem in a decentralized setting thus leading to the design of a distributed,

self-stabilizing algorithm (cf [Dijkstra \[1982\]](#)[EWD386, pp 34-35], which discusses a solution to a cyclic relaxation problem).

3.1.1 Preliminaries and Notation

Consider a set R of n mobile robots $r_1, \dots, r_n$ each associated with some *patrolling speed* p_i and some *walking speed* w_i , where $p_i < w_j$ for $i, j = 1, \dots, n$. Robots perpetually move along the unit segment $[0, 1]$ in both directions. At any moment, a robot r_i may be in a *walking state* in which it moves at its walking speed w_i or in the *patrolling state* and moving at its patrolling speed p_i . Each robot may walk in both directions but its patrolling is always done in the same direction of the interval $[0, 1]$. Hence we can say that each robot is associated with its patrolling direction (positive or negative direction of the interval) which does not change throughout its entire movement.

3.1.2 Patrolling Problems

We consider the following patrolling problems in this thesis:

The Fence Patrolling Problem

Let $\mathcal{K}$ denote a *patrolling schedule*, i.e. an algorithm associated with each of the robots dictating on what speed and in what direction every robot moves at each moment of time. Given a schedule $\mathcal{K}$ produced by algorithm $\mathcal{A}$, the idle time $\mathcal{I}_{\mathcal{K}}(p, t)$ of a point p at time t is defined as the amount of time needed to the next visit of p after time t by any robot of R : $\mathcal{I}_{\mathcal{K}}(p, t) = \min\{t_v > t : \exists i (0 \leq i \leq n \text{ such that } r_i(t_v) = p)\} - t$, where we denote by $r_j(t)$ the point of the segment visited by r_j at time t , for $t \in [0, \infty)$. We are interested in an algorithm minimizing the maximal idle time taken over all points p

and all time moments t . However, as it may be impossible to design an algorithm which achieves the best possible idle time right from the start of the schedule (e.g. robots may be in an unsuitable position to achieve this) we will be allowing any finite delay for checking idle times of segment points. More exactly, the *idleness* of schedule $\mathcal{K}$ is defined as

$$\mathcal{I}_{\mathcal{K}} = \inf_{T \geq 0} \sup_{0 \leq p \leq 1} \mathcal{I}_{\mathcal{K}}(p, T).$$

An optimal patrolling algorithm is a schedule which optimizes the idleness among all possible patrolling algorithms under consideration.

It will be useful to observe that our goal may be viewed as the following equivalent maximization task. Suppose that the robots operate in an infinite line. What is the maximal length segment $[0, L]$ and the perpetual movement of the robots of R , such that in each unit time interval $[t, t + 1]$ each point of $[0, L]$ is visited at least once by a robot in patrolling state. We call such a length L the *patrolling range* of R .

The Distributed Model

In this model, robots have only very primitive capabilities allowing them to execute separate schedules which change according to their perception of the environment. They are *oblivious* and *silent* (cannot communicate) and they cannot process any information. Besides two-speed mobility they can perceive the environment by recognizing obstacles (i.e., endpoints) or other robots that they do not even need to recognize. They are not aware of their patrolling or walking speeds nor of the length of the segment. In the schedules produced by our distributed algorithms the robots function according to the *bouncing-rule*: if a robot collides with another robot or a segment endpoint, then it changes direction as well as moving-state. A patrolling schedule of n robots is in a

stable configuration $(x_1^*, \dots, x_n^*)$ if robot i moves within the interval $[x_{i-1}^*, x_i^*]$ (we set $x_0^* = 0$ and $x_n^* = 1$), bouncing always at its endpoints.

Scheduling with Regular Delivery

Our techniques allows us to solve the following transportation problem. Suppose that at point 1 of the unit segment there is an infinite quantity of a commodity that is transported to point 0 by robots R . Each robot may carry one item of the commodity using a speed not exceeding p_i or it may travel with no load with a speed not exceeding w_i . At any time a robot r_i may drop the item it is carrying at the point currently occupied by r_i or it may pick up an item present at its current position. In particular, when two robots meet at a point an item being carried by one of them may be transferred to the other one. At all times no robot may carry more than one commodity item. What is the smallest value $\mathcal{I}$, such that during any time interval $(k\mathcal{I}, (k+1)\mathcal{I}]$ a new item is delivered to point 0.

3.1.3 Robot Model for Patrolling

While patrolling is performed by searching the domain, in this chapter, we refer to a robot's searching speed as its *patrolling* speed.

We add the following capabilities to our basic robot model for each of the following patrolling algorithms.

Centralized Algorithm

knowledge

Each individual robot has knowledge of:

- Their own patrolling and walking speeds, and
- the leftmost and rightmost points of their assigned patrol region.

The assigned regions are given to the robots from a centralized scheduler who is aware of:

- The number of robots in the fleet, and the speeds of each robot, and
- the length of the interval to be patrolled.

location awareness

Each robot is aware of their location on the interval at all times.

sense of direction

The robots always know in which direction they are oriented.

Decentralized Algorithm**collision detection**

Robots can detect collisions with other robots or the end points of the interval.

local sense of direction

Each robot is aware in which direction they are oriented in relation to their own starting orientation, but the robots do not share a global coordinate system.

3.1.4 Outline and Results

In Section 3.2 we present a centralized algorithm producing optimal patrolling schedules for any number of robots. Section 3.3 studies the distributed case where each robot performs a dynamic schedule controlled solely by its environment through collisions. A dynamic schedule for two robots is proposed and its optimality is proven. For larger

collections of robots, a stronger condition on the robots' speeds (called regularity) is needed. An algorithm producing schedules for regular collections is proposed. A proof of its optimality, using an algebraic approach, is given for regular collections of up to three robots. Optimality of the schedule for larger collections depends on the eigenvalues of matrices whose characteristic polynomials are of higher degree and is left open.

3.2 Optimal Fence Patrolling Centralized Algorithm

In this section we give the general, centralized algorithm generating optimal patrolling schedules for the Fence Patrolling Problem and for any number of robots.

3.2.1 Schedule Obtained by a Centralized Algorithm

Consider the schedule defined by `CentralizedPatrolling`.

Algorithm 3.1: `CentralizedPatrolling`

Input: Set of robots R with associated walking and patrolling speeds

Output: Schedule of R

```

1 Let  $\sigma_j = \sum_{i=1}^j \frac{1}{1/p_i + 1/w_i}$  for  $j = 1, \dots, n$ 
2 for  $j = 1, \dots, n$  do
3   Place robot  $r_j$  at initial position  $x_j = \sigma_j / \sigma_n$ 
4   repeat
5     In patrolling state, move (left) at speed  $p_j$  until reaching point  $x_{j-1}$ 
6     In walking state, move (right) at speed  $w_j$  until reaching point  $x_j$ 
7   until forever
8 end
```

The idea of the algorithm is to divide the unit interval into subsegments, and to make each robot operate only in the subsegment associated with it. Robot r_i is associated with a subsegment of size proportional to $\frac{p_i w_i}{p_i + w_i}$. Each robot zigzags between the endpoints of its subsegment patrolling in one direction and walking in the opposite direction.

In this section we show that `CentralizedPatrolling` produces an optimal patrolling schedule by proving Theorem 3.3. First we need a few observations.

Lemma 3.1. *The idleness $\mathcal{I}_{off}$ of `CentralizedPatrolling` satisfies*

$$\mathcal{I}_{off} = \frac{1}{\sum_{i=1}^n \frac{1}{1/p_i + 1/w_i}}.$$

Proof. As $\sigma_j, j = 0, \dots, n$, is an increasing sequence, each robot r_j operates within the subsegment $[x_{j-1}, x_j]$, for $j = 1, \dots, n$, having interior disjoint with all other subsegments. Consider any index j and any point $x \in [x_{j-1}, x_j]$. Denote by t^* a time when r_j is in the patrolling state and $r_j(t^*) = x$. Before r_j visits x in patrolling state the next time, it has to traverse from x to x_{j-1} patrolling followed by walking the subsegment $[x_{j-1}, x_j]$ and patrolling from x_j to x . The time needed for this equals

$$\frac{x - x_{j-1}}{p_j} + \frac{x_j - x_{j-1}}{w_j} + \frac{x_j - x}{p_j} = (x_j - x_{j-1}) \left(\frac{1}{p_j} + \frac{1}{w_j} \right) = \frac{1}{\sum_{i=1}^n \frac{1}{1/p_i + 1/w_i}}.$$

■

3.2.2 Lower Bound of the Centralized Schedule

We show below that `CentralizedPatrolling` is optimal. We show first

Lemma 3.2. *Let $D > 1$ be the distance patrolled by robot r_i during some time interval. Then in the same time interval r_i must walk distance at least $D - 1$.*

Proof. Suppose, by symmetry, that r_i can patrol only in the right-to-left (i.e. negative) direction of the interval. The difference between the initial and the final position of the robot equals to $R - L$, where L denotes the sum of the lengths of its left-to-right moves and R denotes the sum of the lengths of its right-to-left moves. Observe that $R - L \leq 1$.

As the total patrolling distance is at most R and the total walking distance is at least L we have the claim of the lemma. $\blacksquare$

Theorem 3.3. *CentralizedPatrolling produces an optimal patrolling schedule.*

Proof. Consider any algorithm $\mathcal{A}$ and its idleness $\mathcal{I}_{\mathcal{A}}$. It is sufficient to show, that for any $\epsilon > 0$ there exists a time interval $T = [t^*, t^* + \mathcal{I}_{\text{off}} - \epsilon]$ during which some point of the segment is not patrolled by any robot. Let κ be an integer such that

$$\kappa > \frac{\sum_{i=1}^n \frac{1/w_i}{1/p_i + 1/w_i}}{\epsilon \sum_{i=1}^n \frac{1}{1/p_i + 1/w_i}}.$$

Consider any time interval K of size $= \kappa \mathcal{I}_{\mathcal{A}}$. We prove that K must contain interval T with the property from the claim made above.

Let d_i denote the distance traversed by r_i while patrolling during interval K . As each point of the segment must be patrolled at least κ times during time interval K we have

$$\sum_{i=1}^n d_i \geq \kappa.$$

By Lemma 3.2 we have

$$\frac{d_i}{p_i} + \frac{d_i - 1}{w_i} \leq \kappa \mathcal{I}_{\mathcal{A}},$$

hence

$$d_i \leq \frac{\kappa \mathcal{I}_{\mathcal{A}} + 1/w_i}{1/p_i + 1/w_i}.$$

Therefore

$$\kappa \leq \sum_{i=1}^n d_i \leq \kappa \mathcal{I}_{\mathcal{A}} \sum_{i=1}^n \frac{1}{1/p_i + 1/w_i} + \sum_{i=1}^n \frac{1/w_i}{1/p_i + 1/w_i}$$

and

$$\mathcal{I}_{\mathcal{A}} \geq \frac{1}{\sum_{i=1}^n \frac{1}{1/p_i + 1/w_i}} - \frac{\sum_{i=1}^n \frac{1/w_i}{1/p_i + 1/w_i}}{\kappa \sum_{i=1}^n \frac{1}{1/p_i + 1/w_i}} \geq \frac{1}{\sum_{i=1}^n \frac{1}{1/p_i + 1/w_i}} - \epsilon$$

This proves Theorem 3.3. ■

From the proof of Theorem 3.3 (or more precisely due to Lemma 3.1), it follows that almost all points of the segment have the same idle time (with the exception of the endpoints of subsegments which may be alternately visited by two robots). From `CentralizedPatrolling`, it follows that each robot r_i operates solely inside a subsegment of size $\frac{1}{1/p_i + 1/w_i} / \sum_{i=1}^n \frac{1}{1/p_i + 1/w_i}$, for $i = 1, \dots, n$. By scaling down the consideration to the idle time of 1 we easily obtain:

Corollary 3.4. *The patrolling range of robot r_i having patrolling speed p_i and walking speed w_i equals $\frac{1}{1/p_i + 1/w_i}$. The patrolling range of a set of robots equals the sum of their patrolling ranges.*

3.2.3 Application to Transportation: Scheduling with Regular Delivery

In this section we show a nice application of our previous findings in *Scheduling with Regular Delivery* (see Section 3.1.1 for the definition). The following Proposition shows that the idleness of `CentralizedPatrolling` is the optimal solution for the problem concerning Scheduling with Regular Delivery.

Proposition 3.5. *The solution (i.e., the optimal time interval) of the problem concerning Scheduling with Regular Delivery for the set of robots R equals $\mathcal{I}_{\text{off}}$ - the idleness of `CentralizedPatrolling`.*

Proof. Each robot of `CentralizedPatrolling` revisits the left endpoint of its subinterval (and also its right endpoint) regularly at time intervals of $\mathcal{I}_{off}$. It is then possible to synchronize robot movements so that each robot gets to its subsegment left endpoint (or right endpoint) exactly at the same when arrives there the left (resp. right) neighbouring robot. Suppose that robot r_n picks up an item at each visit to point 1 of the segment, and that, each time a robot reaches the left endpoint of its subinterval, it transfers the carried item to its left neighbour. Then each robot r_i either walks left-to-right unload with speed w_i or transports one item travelling right-to-left. Consequently, r_1 delivers one new item to point 0 at regular time intervals $\mathcal{I}_{off}$. The optimality follows by the argument similar to that used in the proof of Theorem 3.3. ■

3.3 Fence Patrolling in the Distributed Model

In this section we consider the case when the collection of patrolling robots acts in a distributed way (see Section 3.1.1 for the model). We are interested in an algorithm having the same idleness as the one of the optimal centralized algorithm even if our robots are very weak. For notational reference we will assume that there exist two motionless robots r_0 and r_{n+1} (i.e. $w_0 = p_0 = w_{n+1} = p_{n+1} = 0$), positioned at the left and right endpoint, respectively. This way every robot r_i , for $i = 1, \dots, n$, bounces at its left or right neighbour. We have the following lemma.

Lemma 3.6. *For any collection R of robots there exists a centralized algorithm producing an optimal schedule in stable configuration, in which the robots behave according to the bouncing rule.*

Proof. Consider the partition of the unit interval in `CentralizedPatrolling`. By Corollary 3.4 each robot executing `CentralizedPatrolling`, within the same

time interval, independently covers a segment subinterval proportional to its patrolling range. It is then possible to reschedule the robots' starting times so that each robot arrives at the left endpoint of its subinterval exactly at the same time as when its left neighbour arrives at the right endpoint of its interval, resulting in a meeting. ■

We will attempt to design a distributed algorithm producing robots trajectories converging to a schedule of a stable configuration of robots. The task seems of special interest, given that robots are assumed to be oblivious and silent.

FIGURE 3.1: Two robots r_1, r_2 patrolling in opposite directions.

They first collide at a point x and they both change to walking. After bouncing (not necessarily at the same time) at the endpoints of the interval $[0, 1]$ they change to patrolling and meet anew at point x' . The vertical line indicates time.

3.3.1 Distributed Optimal Schedule for Two Robots

The purpose of this section is to demonstrate that two primitive robots can optimally solve the Fence Patrolling Problem.

Let $\mathcal{I}_{opt}$ be the optimal idleness of the offline schedule for two robots. We design an algorithm, for which for any $\epsilon > 0$ there exists a time t^* , such that in every time interval $[t, t + \mathcal{I}_{opt} + \epsilon]$, with $t \geq t^*$, each point of the segment is visited by some robot. Obviously, using such weak robots, it is impossible to design an algorithm which achieves

optimal idle time only after some finite time of their operation. This would need robots capable of recognizing the parameters of the environment (e.g. patrolling and walking speeds of robots, distance travelled, time between collisions, etc.).

Algorithm 3.2: `OnlinePatrolling`

Input: Two robots r_1, r_2 placed at the two segment endpoints

Output: Schedule of R

- 1 Both robots start in patrolling state moving towards each other
 - 2 Each robot switches state and direction when colliding either with the other robot or with an endpoint
-

We show that `OnlinePatrolling` is the optimal one, i.e. its idleness equals the idleness of the optimal offline `CentralizedPatrolling`. The first critical observation is that collision points converge to a stable configuration.

Lemma 3.7. *The sequence of collision points of `OnlinePatrolling` converges to the point*

$$\frac{\frac{1}{p_2} + \frac{1}{w_2}}{\frac{1}{p_1} + \frac{1}{w_1} + \frac{1}{p_2} + \frac{1}{w_2}}.$$

Proof. Suppose that the two robots following `OnlinePatrolling` meet at a point x , $0 < x < 1$. Suppose also that before the meeting both robots were in the patrolling state and moving towards each other. We show first that the next meeting occurs at a point x' such that

$$x' = -\frac{\frac{1}{w_1} + \frac{1}{w_2}}{\frac{1}{p_1} + \frac{1}{p_2}}x + \frac{\frac{1}{p_2} + \frac{1}{w_2}}{\frac{1}{p_1} + \frac{1}{p_2}}. \quad (3.1)$$

Indeed, as $p_1 < w_2$ and $p_2 < w_1$ no robot can be caught from behind by another one, while walking along the segment. Consequently, both robots reach the segment endpoints and they restart patrolling while moving towards each other, eventually colliding at x' . As both robots spend the same time while travelling from x to x' (cf. Figure 3.1) we have $\frac{x}{w_1} + \frac{x'}{p_1} = \frac{1-x}{w_2} + \frac{1-x'}{p_2}$. Therefore $x' \left(\frac{1}{p_1} + \frac{1}{p_2} \right) = -x \left(\frac{1}{w_1} + \frac{1}{w_2} \right) + \frac{1}{p_2} + \frac{1}{w_2}$.

We see from Identity (3.1) that $x' = -Ax + B$ where

$$A := \frac{\frac{1}{w_1} + \frac{1}{w_2}}{\frac{1}{p_1} + \frac{1}{p_2}}, B := \frac{\frac{1}{p_2} + \frac{1}{w_2}}{\frac{1}{p_1} + \frac{1}{p_2}}. \quad (3.2)$$

If x_1 is the initial collision point then we see that the k th collision point satisfies the recurrence $x_k = -Ax_{k-1} + B$. This yields a geometric series from which it follows that $x_k = (-A)^k x_0 + B \frac{1 - (-A)^k}{1 + A}$. Since $A < 1$ as $k \rightarrow \infty$ we get

$$x_k \rightarrow \frac{B}{1 + A} = \frac{\frac{1}{p_2} + \frac{1}{w_2}}{\frac{1}{p_1} + \frac{1}{w_1} + \frac{1}{p_2} + \frac{1}{w_2}}$$

which completes the proof. ■

Our intention is to generalize the lemma above for dynamic systems with arbitrarily many robots. Until then, we show that `OnlinePatrolling` produces the optimal schedule.

Theorem 3.8. *The idleness of the schedule of `OnlinePatrolling` equals $\mathcal{I}_{off}$ - the idleness of the optimal schedule produced by the centralized algorithm.*

Proof. It is sufficient to show that for any $\epsilon > 0$ there exists a point in time t^* such that for any $t > t^*$ and any time interval $T = [t, t + \mathcal{I}_{off} + \epsilon]$ each point of the segment is patrolled by some robot. Observe that from the start to the first meeting point x_0 robot r_1 patrols the interval $[0, x_0]$, while during the same time interval r_2 patrols the interval $[x_0, 1]$. Hence we have $\frac{x_0}{p_1} = \frac{1 - x_0}{p_2}$. Solving for x_0 we get $x_0 = \frac{1}{p_2} / \left(\frac{1}{p_1} + \frac{1}{p_2} \right)$. Observe that the distance $D = |x_k - x_{k-1}|$ between the $(k - 1)$ -th and k -th meeting

points (defined in the proof of Lemma 3.7)

$$\begin{aligned}
D &= \left| \begin{pmatrix} (-A)^k x_0 + B \frac{1-(-A)^k}{1+A} \\ - \left((-A)^{k-1} x_0 + B \frac{1-(-A)^{k-1}}{1+A} \right) \end{pmatrix} \right| \\
&= \left| ((-A)^k - (-A)^{k-1}) \left(x_0 - \frac{B}{1+A} \right) \right| \\
&= |(-A)^k (B - x_0(A+1))| \\
&= A^k (B - x_0(A+1)) \quad (\text{since } A > 0 \text{ and } (B - x_0(A+1)) > 0)
\end{aligned}$$

is converging to 0, since $|A| < 1$. As x_k and x_{k-1} are on the different sides of the convergence point $\frac{B}{1+A}$, within every time interval $\mathcal{I}_{off} + \frac{D}{\min(p_1, p_2)}$ the entire segment is jointly patrolled by both robots. Let K be such that

$$K \geq \log_A \frac{\epsilon \cdot \min(p_1, p_2)}{B - x_0(A+1)}$$

and let

$$t^* > (K+1) \left(\frac{1}{p_1} + \frac{1}{w_1} + \frac{1}{p_2} + \frac{1}{w_2} \right).$$

As $\left(\frac{1}{p_1} + \frac{1}{w_1} + \frac{1}{p_2} + \frac{1}{w_2} \right)$ is the time between two consecutive bounces between robots r_1, r_2 , after time t^* the robots bounced at least k times. Hence (as $0 < A < 1$) for $t > t^*$ the idle time $\mathcal{I}(p, t)$ of every point p is

$$\mathcal{I}(p, t) \leq \mathcal{I}_{off} + \frac{D}{\min(p_1, p_2)} = \mathcal{I}_{off} + \frac{A^k (B - x_0(A+1))}{\min(p_1, p_2)} \leq \mathcal{I}_{off} + \epsilon,$$

which proves the theorem. ■

Observe that `OnlinePatrolling` works even if the robots do not necessarily start their respective schedules at the same time. Indeed, because $p_i < w_j$, $i, j = 1, 2$, when the second robot wakes up and starts patrolling it cannot meet the robot which started

first while this robot is in the walking state. Therefore the robots meet when they are both in the patrolling state and the subsequent meetings converge to the same point as before.

Note that, when robots start walking rather than patrolling, their subsequent meeting points do not converge.¹ On the other hand if one robot starts in the walking state while the other one in the patrolling state the convergence is possible in at most one of the two symmetric cases, depending which among the two values $1/p_1 + 1/w_2$ or $1/w_1 + 1/p_2$ is larger. In particular, when $1/p_1 + 1/w_2 = 1/w_1 + 1/p_2$ (e.g. in the case of identical robots) the meeting points alternate between two symmetric positions on the segment and the idleness is clearly suboptimal (cf. Fig. 3.2).

FIGURE 3.2: Two robots r_1, r_2 perpetually alternate between $x(0)$ and $x(1)$.

r_1 starts by patrolling and r_2 by walking. Here $w_1 = 5, p_1 = 3\frac{1}{7}, w_2 = 10, p_2 = 4$.

3.3.2 Distributed Schedule for n Primitive Robots

In this section we propose a distributed solution for an arbitrary number of primitive robots that induces an interesting dynamic system, which we analyse in subsequent sections.

¹This follows from the proof of Theorem 3.8, as in this case, the critical value of quantity $A = \frac{1/p_1 + 1/p_2}{1/w_1 + 1/w_2} > 1$ (the roles of patrolling and walking speeds are swapped in the equation for x').

As observed for the case of two robots, the convergence was attested if the two robots were colliding always in the patrolling state. For more than two robots this condition can no longer be guaranteed as all but two robots collide with both neighbours. We begin our analysis by proposing `DistributedPatrolling`, an intuitive distributed schedule that assumes the bouncing rule, i.e. that robots can respond to bounces by flipping their moving state (e.g. from walking to patrolling) and their moving direction. Moreover, as in our setting robots will start walking simultaneously at the same segment endpoint we naturally assume that $p_i \neq p_j$ (and that $w_i \neq w_j$), otherwise identical robots would always stay together and only one of them would contribute to the patrolling algorithm.

Algorithm 3.3: `DistributedPatrolling`

Input: A collection R of robots with distinct patrolling and walking speeds

Output: Schedule of R

- 1 All robots start from the rightmost endpoint of the interval, in patrolling state moving right-to-left
 - 2 Each robot switches state and direction while colliding with the other robot or with an endpoint
-

As all robots patrol in the same direction and they change states only when meeting we can conclude with the following:

Observation 3.9. `DistributedPatrolling` produces a dynamic schedule with Regular Delivery, for a given set of robots.

Notice that our algorithm defines a fascinating dynamical system of memoryless robots moving back and forth in an interval. The analysis of the system dynamics is very complicated, given that robot collisions might occur either between robots moving in opposite directions, or in the same directions (i.e. a collision may happen from behind). In what follows we analyse the dynamics under the assumption that we have one type of collisions, which as we shall see in the next subsections naturally arise by restricting the configuration of the robot speeds to what we later call monotone speeds.

We call the dynamical system that arises from `DistributedPatrolling` *regular* if collisions occur only between robots that move in opposing directions, and therefore collisions happen only between a robot that is in patrolling state moving right-to-left and a robot in walking state moving left-to-right. We are interested in answering whether regular dynamical systems converge in a stable configuration, and whether this configuration has optimal idle time. Below we propose a highly efficient algorithm for verifying whether a regular dynamical system has this property (for any number of robots). Then we answer this question in the positive for up to 4 robots, under the assumption that speeds satisfy a natural condition.

3.3.3 Dynamics for Regular Systems of Primitive Robots

Dynamic systems of primitive robots induced by `DistributedPatrolling` are highly complex and difficult to analyse. The purpose of this section is to provide a deep and technical analysis of the dynamics of regular systems. We conclude the section by proposing a highly non-trivial algorithm for deciding convergence of primitive robots in a stable configuration (where robots eventually move in disjoint subintervals).

Notice that in every dynamical system, and at every point in the time horizon, robots will appear on the interval in the same order. We rename the robots so that robot $i + 1$ is to the right of robot i , after they develop according to `DistributedPatrolling`. Below we denote by x_t^i the point in the interval where robots r_i, r_{i+1} bounce for the t -th time. The purpose of the next lemma is to predict points x_t^i . The reader may view it as the analogue of (part of the proof of) Lemma 3.7 that dealt with only two robots.

Lemma 3.10. *In a regular system we have*

$$\left(\frac{1}{p_{i+1}} + \frac{1}{w_i}\right) x_{t+1}^i - \left(\frac{1}{p_i} + \frac{1}{w_i}\right) x_{t+1}^{i-1} = \left(\frac{1}{p_{i+1}} + \frac{1}{w_{i+1}}\right) x_t^{i+1} - \left(\frac{1}{p_i} + \frac{1}{w_{i+1}}\right) x_t^i$$

Proof. First we claim that $x_t^{i-1} < x_t^i$, $i = 1, \dots, n$, and that robots r_{i-1}, r_i bounce at x_t^{i-1} before r_i, r_{i+1} bounce at x_t^i . Indeed, let $\tau(x_t^i)$ denote the time of this bounce. Note that robot r_1 (while patrolling) bounces first at the origin and on its way back (now walking) bounces with robot r_2 (which is patrolling). After robot r_2 bounces with robot r_1 , it begins walking and eventually bounces with robot r_3 which moves in opposite direction and is patrolling, etc. This reasoning shows that $x_1^{i-1} < x_1^i$, $i = 1, \dots, n$ as well as that r_{i-1}, r_i bounce for the first time before r_i, r_{i+1} do. Since the system is regular, each robot alternates its bounces between both neighbours r_{i-1}, r_{i+1} which is sufficient to conclude the claim.

Next observe that between time $\tau(x_t^i)$ and $\tau(x_{t+1}^i)$ robot r_{i-1} first patrols right-to-left the interval $[x_{t+1}^{i-1}, x_t^i]$ then it walks left-to-right the interval $[x_{t+1}^{i-1}, x_{t+1}^i]$. During the same time interval r_i first walks left-to-right the interval $[x_t^i, x_t^{i+1}]$ then it patrols right-to-left the interval $[x_{t+1}^i, x_{t+1}^{i+1}]$ (see Figure 3.3).

Comparing both times we get

$$\frac{x_t^i - x_{t+1}^{i-1}}{p_i} + \frac{x_{t+1}^i - x_{t+1}^{i-1}}{w_i} = \frac{x_t^{i+1} - x_t^i}{w_{i+1}} + \frac{x_{t+1}^{i+1} - x_{t+1}^i}{p_{i+1}}.$$

Regrouping terms implies the lemma. ■

FIGURE 3.3: Two time consecutive bounces between robots r_i, r_{i+1} .

We can rewrite the recurrence in Lemma 3.10 in a more concise matrix form. Define the following matrices.

$$\begin{aligned} A &= \text{diag} \left(\frac{1}{p_{i+1}} + \frac{1}{w_i} \right)_{i=1, \dots, n-1} - \text{L-diag} \left(\frac{1}{p_i} + \frac{1}{w_i} \right)_{i=2, \dots, n-1} \\ B &= \text{diag} \left(\frac{1}{p_i} + \frac{1}{w_{i+1}} \right)_{i=1, \dots, n-1} - \text{U-diag} \left(\frac{1}{p_i} + \frac{1}{w_i} \right)_{i=2, \dots, n-1} \\ c^T &= \left(0, \dots, \frac{1}{p_n} + \frac{1}{w_n} \right), \end{aligned} \quad (3.3)$$

where by L-diag and U-diag we mean the low and upper diagonal matrices of dimension $(n-1) \times (n-1)$ and entries as indicated placed below and above the main diagonal, respectively.

Theorem 3.11. *Consider a regular dynamical system of n robots which is produced by `DistributedPatrolling` and let A, B, c be the matrices defined in Equations (3.3). If the moduli (norms) of all eigenvalues of the matrix $A^{-1}B$ are less than 1, then the schedule of `DistributedPatrolling` converges to a schedule in stable configuration which is also optimal (w.r.t. to centralized algorithms). In particular, for every $\epsilon > 0$, after $\Theta(\log 1/\epsilon)$ bounces (iterations) of any pair of neighbouring robots the idle time $\mathcal{I}_{\text{DistPatrol}}(p, t)$ is such that*

$$\mathcal{I}_{\text{DistPatrol}}(p, t) \leq (1 + \epsilon) \frac{1}{\sum_{i=1}^n \frac{1}{1/p_i + 1/w_i}}.$$

Proof. Let $X_t \in \mathbb{R}^{n-1}$ be the vector $(x_t^1, \dots, x_t^{n-1})^T$. Then the recurrence of Lemma 3.10 can be rewritten in matrix form as $AX_{t+1} + BX_t = c$. From this, we derive that

$$\begin{aligned} X_t &= (-1)^t (A^{-1}B)^t \\ &\quad + (I + A^{-1}B)^{-1} \left(I - (-1)^{-1} (A^{-1}B)^t \right) A^{-1}c. \end{aligned}$$

Next consider the eigenvalue decomposition $A^{-1}B = Q\Lambda Q^{-1}$, where Q is an orthogonal matrix. Then $(A^{-1}B)^t = Q\Lambda^t Q^{-1}$, and since $\lim_{t \rightarrow \infty} \Lambda^t = \mathbf{0}$ as all eigenvalues have norm less than 1, we conclude that $\lim_{t \rightarrow \infty} X_t$ exists, i.e. the sequence converges to $X^* = (I + A^{-1}B)^{-1} A^{-1}c = (A + B)^{-1}c$ and the convergence is linear. From the definition of the recurrence, it follows that the schedule of `DistributedPatrolling` converges to the schedule $\mathcal{K}$ which is in stable configuration X^* .

By Corollary 3.4 which, in view of Lemma 3.6, applies also to stable configurations the patrolling range of the collection of robots equals

$$\sum_{i=1}^n \frac{1}{1/p_i + 1/w_i}$$

and by the rate of convergence, after $\Theta(\log 1/\epsilon)$ bounces of neighbouring robots, the idle time is already no more than

$$(1 + \epsilon) \frac{1}{\sum_{i=1}^n \frac{1}{1/p_i + 1/w_i}}.$$

■

Note that already Theorem 3.11 suggests a numerical method for checking whether the dynamical system arising from `DistributedPatrolling` converges or not; given patrolling and walking speeds p_i, w_i , first compute matrix A^{-1} , and then calculate all eigenvalues of $A^{-1}B$ and verify that their norm is less than 1. Matrix inversion can be done explicitly and efficiently, say by Gauss-Jordan elimination or by LU decomposition. Finding however the eigenvalues of the non-Hermitian matrix $A^{-1}B$ is at least as difficult as finding the roots of high degree polynomials. In light of Abel's impossibility theorem, one has to rely on numerical methods to verify that the moduli of the

eigenvalues of $A^{-1}B$ are indeed less than 1. In fact, a number of sophisticated numerical methods have been proposed to efficiently find eigenvalues of special families of matrices.

We depart from this approach, and in contrast to numerical methods, we propose an explicit, symbolic and efficient algorithm for verifying the precondition of Theorem 3.11 without explicitly computing the eigenvalues of $A^{-1}B$. Our strategy is to first give an explicit expression of A^{-1} , which in turn will allow us to calculate the characteristic polynomial of $A^{-1}B$. Finally, we invoke a powerful theorem that characterizes the range of polynomial roots (without finding them), and that can be exploited algorithmically.

We begin by calculating the characteristic polynomial of $A^{-1}B$. Note that for a group R of n robots, the characteristic polynomial of $A^{-1}B$ is of degree $n - 1$. Also, any $r \times r$ leading principal minor $A^{-1}B$ can be computed from the $r \times r$ leading principal minors of A, B , which only depend on robots $1, \dots, r + 1$. In fact the $r \times r$ leading principal minor $A^{-1}B$ is exactly the critical matrix whose eigenvalues determine the convergence of `DistributedPatrolling` for input robots $1, \dots, r + 1$. So, we are motivated in denoting by $D_r(\lambda)$ the characteristic polynomial of the $r \times r$ leading principal minor $A^{-1}B$ (i.e. $D_{n-1}(\lambda) = |A^{-1}B - \lambda I|$). We choose to abbreviate $D_r(\lambda)$ by D_r . The next lemma provides two alternative recursive relations for D_r (each will be convenient in different arguments) that allow us to compute D_{n-1} , and will be also used later to establish convergence for special cases of robots.

Lemma 3.12. *Consider matrices A, B as defined in (3.3), and introduce the abbreviations $a_i := \frac{1}{p_{i+1}} + \frac{1}{w_i}$, $b_i := \frac{1}{w_{i+1}} + \frac{1}{p_i}$, $c_i := -\frac{1}{p_{i+1}} - \frac{1}{w_{i+1}}$, and $\alpha_i := \frac{b_i}{a_i} - \frac{c_{i-1}^2}{a_{i-1}a_i}$, $\beta_i := \frac{c_{i-1}^2}{a_{i-1}a_i}$. Then for the characteristic polynomials D_r , the following equivalent recurrences*

hold true for all $r \geq 2$

$$D_r = (\alpha_r - \lambda) \cdot D_{r-1} + \sum_{t=1}^{r-2} \left(\alpha_{t+1} \cdot \prod_{j=t+2}^r \beta_j \right) \cdot D_t + \frac{b_1}{a_1} \prod_{j=2}^r \beta_j, \quad (3.4)$$

$$D_r = \left(\frac{b_r}{a_r} - \lambda \right) D_{r-1} + \frac{c_{r-1}^2}{a_{r-1}a_r} \lambda D_{r-2} \quad (3.5)$$

and with initial conditions $D_1 = \frac{b_1}{a_1} - \lambda$, $D_0 = 1$.

Proof. The inverse of A is a lower triangular matrix and can be easily verified to be defined as

$$A_{ij}^{-1} := (-1)^{i+j} \frac{\prod_{t=j}^{i-1} c_t}{\prod_{r=j}^i a_r}, \quad \text{if } j \leq i$$

and is 0 otherwise. Therefore, for the (so-called Hessenberg matrix, as it has zero entries above the first subdiagonal) matrix $A^{-1}B$ we have that

$$(A^{-1}B)_{ij} = (-1)^{i+j} \left(\frac{b_j}{a_i} \prod_{t=j}^{i-1} \frac{c_t}{a_t} - \frac{c_{j-1}}{a_i} \prod_{t=j}^i \frac{c_{t-1}}{a_{t-1}} \right), \quad \text{if } i \geq j - 1$$

and 0 otherwise, with the understanding that $c_0 = 0$. The following interesting relation holds for the entries of $A^{-1}B$, that is useful in finding the characteristic polynomial of the matrix.

$$(A^{-1}B)_{i,j} = -\frac{c_{i-1}}{a_i} (A^{-1}B)_{i-1,j}, \quad \forall i > j. \quad (3.6)$$

Next we introduce D'_r to denote a small variation of D_r . D'_r is the determinant of the same principal minor of $A^{-1}B - \lambda I$ (up to entry (r, r)) with the only difference that the entry (r, r) is replaced by $(A^{-1}B)_{r,r}$, instead of $(A^{-1}B - \lambda I)_{r,r}$.

With this notation, we can evaluate $|A^{-1}B - \lambda I|$ by expanding the determinant with respect to the entries $(n-1, n-1)$ and $(n-2, n-1)$. Using Equation (3.6), we

observe that

$$D_r = \left(\frac{b_r}{a_r} - \frac{c_{r-1}^2}{a_{r-1}a_r} - \lambda \right) D_{r-1} + \frac{c_{r-1}^2}{a_{r-1}a_r} D'_{r-1}, \quad (3.7)$$

$$D'_r = \left(\frac{b_r}{a_r} - \frac{c_{r-1}^2}{a_{r-1}a_r} \right) D_{r-1} + \frac{c_{r-1}^2}{a_{r-1}a_r} D'_{r-1}, \quad (3.8)$$

where the recurrence ends at $D_1 = \frac{b_1}{a_1} - \lambda$ and $D'_1 = \frac{b_1}{a_1}$. Repeated substitution of (3.8) to (3.7) and some direct calculations imply recurrence (3.4). Recurrence (3.5) is obtained from (3.4) by subtracting two consecutive terms of the sequence D_r . ■

Notice that Lemma 3.12, and in particular Equation (3.5), allows us to calculate the characteristic polynomial D_{n-1} of $A^{-1}B$ by performing no more than $\Theta(n^2)$ arithmetic operations (additions, multiplications and divisions) between speeds p_i, w_i . Next we give an efficient algorithm for deciding whether the moduli of the roots of an arbitrary polynomial $f : \mathbb{R} \mapsto \mathbb{R}$ are all less than 1. Our intention is to run `DecideConverge` with input D_{n-1} , i.e. the characteristic polynomial of $A^{-1}B$.

Algorithm 3.4: `DecideConverge`

Input: A polynomial $f : \mathbb{R} \mapsto \mathbb{R}$ of degree t of the form $\sum_{i=0}^t \gamma_i \lambda^i$

Output: YES if and only if $\delta_1 < 0$ and $\delta_j > 0$ for $j = 2, \dots, t$

```

1 Set  $\gamma_i^{(0)} = \gamma_i$ , for  $i = 0, \dots, t$ 
2 for  $j = 0, \dots, t - 1$  do
3   for  $k = 0, \dots, j + 1$  do
4      $\gamma_k^{(j+1)} \leftarrow \gamma_0^{(j)} \gamma_k^{(j)} - \gamma_{n-j}^{(j)} \gamma_{n-j-k}^{(j)}$ 
5   end
6 end
7 for  $j = 0, \dots, t - 1$  do
8    $\delta_{j+1} \leftarrow \gamma_0^{(j+1)} = \left( \gamma_0^{(j)} \right)^2 - \left( \gamma_{n-j}^{(j)} \right)^2$ 
9 end
```

Theorem 3.13. *A set of n robots R for which the output of `DistributedPatrolling` gives a regular dynamical system converges to a stable configuration if and only if*

DecideConverge outputs YES on input D_{n-1} . As a result, convergence can be decided in $\Theta(n^2)$ arithmetic operations.

Proof. By Theorem 3.11, the regular dynamical system converges to a stable configuration if and only if all eigenvalues of $A^{-1}B$ (as defined in (3.3)) have moduli less than 1. The characteristic polynomial of $A^{-1}B$ can be computed in $\Theta(n^2)$ many operations, as a corollary of Lemma 3.12. Clearly, *DecideConverge* requires no more than $\Theta(n^2)$ arithmetic operations. Therefore, we can decide convergence in $\Theta(n^2)$ arithmetic operations as long as we can show that *DecideConverge* correctly decides whether the input polynomial f has all its roots (real or complex) strictly inside the unit circle.

Correctness of *DecideConverge* is an immediate corollary of Theorem 42.1, p. 150 in Marden [1949]: “Set $\Delta_r = \prod_{j=1}^t \delta_j$, $r = 1, \dots, t$ and suppose that k many of the products Δ_r are negative, and the remaining $t - k$ of them are positive. Then f has exactly k roots strictly inside the unit circle, exactly $t - k$ roots strictly outside the unit circle (and hence no roots on the unit circle).” ■

3.3.4 Monotone Robot Collections, and Convergence

In this section we demonstrate some special families of primitive robots that can solve Fence Patrolling optimally. The analysis even of the restricted families of three or four robots remains surprisingly technical and non-trivial.

Our technical results of Section 3.3.3 on regular dynamical systems raise the question whether such systems exist. A natural family of robots is when either the sum or the product of patrolling and walking speeds is constant for all robots or when some constant “power” of a robot may be used for improving its patrolling ability at the expense of its walking ability. In such a collection of robots, all patrolling speeds are dominated

by the walking speeds, and the non-increasing order of patrolling speeds is the inverse order of that of the walking speeds. We make the definition formal.

Definition 3.14. The collection R of n robots is called monotone if for $i, j = 1, \dots, n$:

1. $p_i < w_i$,
2. $p_i \neq p_j$, and
3. $p_i < p_j \implies w_i > w_j$.

A natural example of a monotone collection of robots is one where each robot i independently decides how to distribute its energy e , which is the same for all robots, to walking and patrolling speeds w_i, p_i respectively, such that $w_i + p_i = e$. As it is observed before, a collection of robots that develop according to `DistributedPatrolling` preserve the order they appear on the line. Without loss of generality we may assume that their indices are consecutive along the segment, i.e. that $w_n > w_{n-1} > \dots > w_1 > p_1 > p_2 > \dots > p_n$.

Lemma 3.15. *For a monotone collection of robots R , the dynamical system that arises from `DistributedPatrolling` is regular (i.e. collisions occur while robots approach each other, the left one being in the walking state and the right one in the patrolling state.)*

Proof. Initially all robots patrol right-to-left until the fastest patrolling robot collides with the left endpoint and starts walking left-to-right. Any “head on” collision results in the robot on the right switching to patrolling left-to-right and the robot on the left switching to patrolling right-to-left. So it is sufficient to prove that collisions from behind never take place. Suppose to the contrary, that there exists such a collision between a pair of consecutive robots on the segment, r_i, r_{i+1} , for $i = 1, \dots, n - 1$.

Obviously r_i, r_{i+1} cannot collide when r_i moves left and r_{i+1} moves right. If both robots move right-to-left then, by assumption, they must be walking, and since $w_i < w_{i+1}$, r_i cannot catch r_{i+1} . Similarly, if both robots move left-to-right, by assumption, they are patrolling and since $p_i > p_{i+1}$, r_{i+1} cannot catch r_i . ■

As an immediate observation, we also obtain that

$$\frac{b_i}{a_i} < 1 \quad \forall i = 1, \dots, n-1, \quad \& \quad \frac{(c_{i-1})^2}{a_{i-1}a_i} < 1 \quad \forall i = 2, \dots, n-1 \quad (3.9)$$

for all regular collections of n robots, where a_i, b_i, c_i are as in Lemma 3.12. In fact, the characteristic polynomial D_{n-1} has leading coefficient $(-1)^n$, while it is also immediate from (3.5) that the constant coefficient is $\prod_{i=1}^{n-1} \frac{b_i}{a_i} < 1$. This automatically shows that the condition $\delta_1 < 0$, of `DecideConverge`, holds true for monotone collections of robots. In fact, we conjecture that monotone collections of robots always converge to a stable configuration, i.e. that $\delta_j > 0$ for $j = 2, \dots, n-1$, but a general proof is eluding us. Still the proof of convergence for up to $n \leq 3$ robots is easy to establish. The proof of the next proposition relies on (3.9).

Proposition 3.16. *For a monotone collection of $n \leq 3$ robots the schedule produced by `DistributedPatrolling` has the optimal idleness.*

Proof. We show that the conditions of Theorem 3.11 are satisfied. The case of 1 robot is straightforward. For $n = 2$ robots, the characteristic polynomial is $D_1 = \frac{b_1}{a_1} - x$, which by (3.9) has one real root with absolute value less than 1.

Now we turn our attention to $n = 3$, and by (3.5) the characteristic polynomial $D_2(\lambda)$ has the form

$$P_W(\lambda) := (U - \lambda)(V - \lambda) + W\lambda = \lambda^2 - (U + V - W)\lambda + UV$$

where U, V, W are non negative constants which by (3.9) are strictly less than 1.

We show that the moduli of the roots of $P_W(\lambda)$ are less than 1. First we observe that $P_0(\lambda)$ has this property.

Case 1: If $P_W(\lambda)$ has real roots, then these are

$$\rho_{1,2}^{(W)} = \frac{U + V - W \pm \sqrt{(U + V - W)^2 - 4UV}}{2},$$

with the understanding that $\rho_1^{(W)}, \rho_2^{(W)}$ correspond to the square root having positive and negative sign respectively. Then we observe that $\rho_1^{(W)} < \rho_1^{(0)} < 1$, while also $\rho_1^{(W)} > -W/2 > -1/2$ (by ignoring the positive terms). Hence $-1/2 < \rho_1^{(W)} < 1$. Similarly, we see that $\rho_2^{(W)} < (U + V)/2 < 1$ (by ignoring the negative terms). And finally note that

$$U + V - \sqrt{(U + V - W)^2 - 4UV} > -W,$$

since $(U + V + W)^2 > (U + V - W)^2 - 4UV$. Therefore, $\rho_2^{(W)} > (-2W)/2 = -1$, concluding that $-1 < \rho_2^{(W)} < 1$, as well.

Case 2: If $P_W(\lambda)$ has complex roots, say σ_1, σ_2 , then it must be the case that $\|\sigma_1\|^2 = \|\sigma_2\|^2 = \sigma_1\sigma_2 = UV < 1$. ■

We now prove convergence of monotone collections of $n = 4$ robots by using a refinement of monotonicity.

Definition 3.17. The collection R of n robots is called strongly monotone if it is monotone and for all r we have $\left(\frac{1}{p_r} + \frac{1}{w_{r+1}}\right) \left(\frac{1}{p_r} + \frac{1}{w_{r-1}}\right) > \left(\frac{1}{p_r} + \frac{1}{w_r}\right)^2$.

Due to the definitions of a_i, b_i and that of α_i in Lemma 3.12, asking that a collection of robots is strongly monotone is equivalent to asking that $\alpha_r > 0$ for every r (see (3.4)).

This allows us to show that characteristic polynomials associated with such robots have

no negative real roots. We can now prove that the characteristic polynomial of every strongly monotone collection of robots has real roots less than 1 in absolute value.

Theorem 3.18. *For every monotone (not necessarily strongly) collection of n robots, $D_r(\lambda)$ preserves sign for all $\lambda \geq 1$. If in addition robots are strongly monotone, then $D_r(\lambda)$ preserves sign (and is actually positive) for all $\lambda < 0$. As a result all real roots of D_r lie strictly between 0 and 1.*

Proof. The less technical proof concerns the strongly monotone collections of robots. For this consider the cone $\mathcal{C}$ of polynomials of the form $\sum_{t=0}^r (-1)^t \rho_t \lambda^t$, where $\rho_t > 0$, i.e. polynomials whose odd-degree monomial coefficients are negative, and whose even-degree monomial coefficient are positive. Clearly, any polynomial $p(\lambda) \in \mathcal{C}$ is positive for every $\lambda < 0$ (and is actually decreasing).

We claim that for all $r \geq 0$, $D_r \in \mathcal{C}$. To that end, we first observe that the statement is true for $r = 0, 1$. For any $r \geq 2$, we invoke (3.4). Since all α_i, β_j are positive reals, we can show that $D_r \in \mathcal{C}$ as long as we can verify that $(\alpha_r - \lambda) \cdot D_{r-1} \in \mathcal{C}$ (the rest of summands in (3.4) are conical combinations of polynomials in $\mathcal{C}$). It is straightforward now to check that $-\lambda D_{r-1} \in \mathcal{C}$, hence $(\alpha_r - \lambda) \cdot D_{r-1} = \alpha_r D_{r-1} + (-\lambda D_{r-1}) \in \mathcal{C}$, as wanted.

Now we focus on a monotone (not necessarily strongly) collection of robots. We prove by induction on r that for all $\lambda \geq 1$, D_r is a polynomial which is

$$\begin{cases} \text{positive and increasing,} & \text{if } r \text{ is even} \\ \text{negative and decreasing,} & \text{if } r \text{ is odd} \end{cases}$$

Indeed, the statement is true for $r = 1, 2$. Next we turn our attention to $r \geq 3$. We have in mind to invoke (3.5). Now fix any $\lambda_0 \geq 1$. Note that $\frac{b_r}{a_r} - \lambda_0 < 0$. Next observe that

if r is even, then $D_{r-1}(\lambda_0) < 0$ and $D_{r-2}(\lambda_0) > 0$, so that $D_r(\lambda_0) > 0$. Similarly, if r is odd, then $D_{r-1}(\lambda_0) > 0$ and $D_{r-2}(\lambda_0) < 0$, so that $D_r(\lambda_0) < 0$, exactly as wanted.

Next we show the promised monotonicity. Let's denote by $D'_r(\lambda)$ the first derivative of $D_r(\lambda)$ with respect to λ . Then we see that

$$D'_r(\lambda) = -\lambda D_{r-1}(\lambda) + \left(\frac{b_r}{a_r} - \lambda\right) D'_{r-1}(\lambda) + \frac{c_{r-1}^2}{a_{r-1}a_r} D_{r-2}(\lambda) + \frac{c_{r-1}^2}{a_{r-1}a_r} \lambda D'_{r-2}(\lambda)$$

If r is even then we argue that $D'_r(\lambda) > 0$ for all $\lambda \geq 1$. Indeed, we have that

$$\begin{aligned} -\lambda D_{r-1}(\lambda) &= - \cdot + \cdot - = + \\ \left(\frac{b_r}{a_r} - \lambda\right) D'_{r-1}(\lambda) &= - \cdot - = + \\ \frac{c_{r-1}^2}{a_{r-1}a_r} D_{r-2}(\lambda) &= + \cdot + = + \\ \frac{c_{r-1}^2}{a_{r-1}a_r} \lambda D'_{r-2}(\lambda) &= + \cdot + \cdot + = + \end{aligned}$$

Since all summands of $D'_r(\lambda)$ are positive, $D_r(\lambda)$ is increasing.

Similarly, if r is odd, we show that $D'_r(\lambda) < 0$ for all $\lambda \geq 1$. Indeed, we have that

$$\begin{aligned} -\lambda D_{r-1}(\lambda) &= - \cdot + \cdot + = - \\ \left(\frac{b_r}{a_r} - \lambda\right) D'_{r-1}(\lambda) &= - \cdot + = - \\ \frac{c_{r-1}^2}{a_{r-1}a_r} D_{r-2}(\lambda) &= + \cdot - = - \\ \frac{c_{r-1}^2}{a_{r-1}a_r} \lambda D'_{r-2}(\lambda) &= + \cdot + \cdot - = - \end{aligned}$$

Since all summands of $D'_r(\lambda)$ are negative, $D_r(\lambda)$ is decreasing. ■

In order to show that a group of strongly monotone robots converges to a stable configuration, it remains to prove all complex roots of D_r have norm < 1 ; this is the main idea behind the proof of Proposition 3.19.

Proposition 3.19. *For a strongly monotone collection of $n = 4$ robots the schedule produced by `DistributedPatrolling` has the optimal idleness.*

Proof. Again, we show that the conditions of Theorem 3.11 are satisfied. When $n = 4$, and using (3.5), we can write the characteristic polynomial we need to study, that has the form

$$D_3(\lambda) = (A_3 - \lambda)(A_2 - \lambda)(A_1 - \lambda) + \lambda(B_2(A_3 - \lambda) + B_3(A_1 - \lambda))$$

where by A_i we abbreviate b_i/a_i and by B_i we abbreviate $c_{i-1}^2/a_{i-1}a_i$. Next we argue that all roots of D_3 have norm less than 1. By assuming strong monotonicity, Theorem 3.18 says that all real roots have norm between 0 and 1. Hence, we only need to check any complex roots. All we need to use below is that $0 \leq A_i, B_i < 1$, and this follows by assuming simple (speed) monotonicity.

Since D_3 is of degree 3, it always has a real root, call it r , and at most two complex roots (that are conjugate to each other), say with norm $\|\rho\|$. Since the constant term of D_3 is $A_1A_2A_3$, it follows that $\|\rho\| = \frac{A_1A_2A_3}{r}$. Next we prove that $r \geq \min\{A_1, A_2, A_3\}$, concluding what we need. Indeed, consider the polynomial

$$\begin{aligned} \frac{1}{B_2 + B_3} D_3(\lambda) &= \frac{1}{B_2 + B_3} (A_3 - \lambda)(A_2 - \lambda)(A_1 - \lambda) \\ &\quad + \lambda \left(\frac{B_2}{B_2 + B_3} (A_3 - \lambda) + \frac{B_3}{B_2 + B_3} (A_1 - \lambda) \right) \end{aligned}$$

which clearly has the same roots as $D_3(\lambda)$. Now, the root r above is a value for λ that satisfies the following equality

$$\frac{1}{B_2 + B_3}(A_3 - \lambda)(A_2 - \lambda)(A_1 - \lambda) = -\lambda \left(\frac{B_2}{B_2 + B_3}(A_3 - \lambda) + \frac{B_3}{B_2 + B_3}(A_1 - \lambda) \right)$$

The left-hand-side polynomial, has real roots A_1, A_2, A_3 , and most importantly it is decreasing for all $x \leq \min\{A_1, A_2, A_3\}$ and for all $\lambda \geq \max\{A_1, A_2, A_3\}$. The right-hand-side polynomial is of degree 2, and has two real roots. One of them is 0, and the other, call it $\bar{r}$, is a convex combination of A_1, A_3 , hence we have $\min\{A_1, A_3\} \leq \bar{r} \leq \max\{A_1, A_3\}$. Moreover, the degree 2 polynomial is negative for $0 < \lambda < \bar{r}$ and positive for $\lambda > \bar{r}$, and is increasing for all $\lambda \geq \bar{r}$. Since $\bar{r}$ is in the line segment between $\min\{A_1, A_3\}, \max\{A_1, A_3\}$, it must be the case that the graphs of the two polynomials intersect for some λ between $\min\{A_1, A_2, A_3\}$ and $\max\{A_1, A_2, A_3\}$. Therefore, $r \geq \min\{A_1, A_2, A_3\}$ as wanted. ■

3.4 Conclusion

Patrolling a given domain with a swarm of two-speed robots is a challenging problem with interesting trade-offs. Its difficulty, even for patrolling a segment, is due to the fact that there are many patrolling strategies to be taken into account. We gave an optimal offline algorithm for any robot collection and optimal dynamic schedules for two robots. We also gave an efficient algorithm deciding self-stabilization of a distributed schedule for the case of regular dynamical systems. We proved that the distributed algorithm is self-stabilizing for up to four robots whose speeds satisfy a certain monotonicity property. However, convergence of our distributed algorithm for more than four robots (whose speeds satisfy a certain monotonicity property) remains open.

Chapter 4

Rendezvous

4.1 Introduction

Rendezvous – the *gathering* of exactly two robots – is an fundamental problem in robotics and communications which appears any time mobile agents need to meet to exchange information.

Very few results exist for rendezvous on the ring where robots have differing single speeds. We therefore begin by examining the special case where the robots’ search and walking speeds are identical. We refer to this as *one-speed* rendezvous.

For two-speed rendezvous, rendezvous will only occur if at least one of the two robots is in a searching state while the robots are co-located. This distinction is meaningless for one-speed rendezvous, as we assume the robots are perpetually in a searching state.

¹This chapter contains results which first appeared in E. Kranakis, D. Krizanc, F. MacQuarrie and S. Shende. Randomized rendezvous algorithms for agents on a ring with different speeds. In *Proceedings of the 16th International Conference on Distributed Computing and Networking (ICDCN 2015)*, pages 9:1 – 9:10. ACM, 2015.

4.1.1 Preliminaries and Notation

We give randomized algorithms for rendezvous where the initial direction of movement is chosen at random. The two robots have different speeds u and 1, with $u > 1$. The robots are unaware of their speeds, which remain constant throughout their movement in the ring. The algorithms discussed are *uniform* in that the robots execute identical algorithms regardless of their speed. The ring topology is in the continuous domain and the robots can move in either direction with their respective speeds. The first step in our algorithms is always random in that the robots randomly and independently of each other select their initial direction of movement between clockwise (*cw*) and counter-clockwise (*ccw*).

4.1.2 Rendezvous Problem

We consider the following rendezvous problem in this thesis:

Randomized Rendezvous Problem

Problem 4.1. *Two robots are placed uniformly at random on a closed fence of length L . The robots are equipped with identical chronometers, execute identical algorithms, but have different speeds $u, 1$ (where $u > 1$). The robots have access to a finite number of random bits, and may have some knowledge of their environment (the magnitude of u , the length of the fence L , etc.) The goal of the Randomized Rendezvous Problem is to provide an algorithm which minimizes the expected time until the robots achieve rendezvous.*

4.1.3 Robot Model for Rendezvous

We add the following capabilities to our basic robot model in order to execute the one-speed rendezvous algorithms.

knowledge

Depending on the algorithm, robots may have the following knowledge:

- The length of the ring L ,
- the speed of the robots, but not which speed is its own,
- the number of randomized rounds to execute, and
- the ratio of speeds to the length of the ring.

random bits

The robots have access to random bits, the number specified by the algorithm.

4.1.4 Outline and Results

In the sequel we give several randomized algorithms for rendezvous. Section 4.2 gives a randomized rendezvous algorithm Rendezvous_0 with $\frac{u \cdot L}{2(u^2 - 1)}$ expected time to rendezvous and which requires no particular a priori knowledge other than simply that the topology is a ring. The algorithm is proved to be optimal among all randomized algorithms employing a single random bit during their execution.

Section 4.3 is devoted to the analysis of a randomized algorithm $\text{Rendezvous}_1(k)$ such that during its execution each robot uses at most $k + 1$ random bits. The algorithm is shown to be optimal within a specified class of randomized algorithms. We also demonstrate a *sharp threshold* on the expected rendezvous time depending on the

speed u , in that among all randomized algorithms using at most $k + 1$ random bits, $\text{Rendezvous}_1(k)$ is optimal, for $u \leq 2$, while Rendezvous_0 is optimal, for $u > 2$. Therefore, the ability to use random bits to change direction is only beneficial when the difference in speeds is low ($u < 2$).

Section 4.4 is devoted to the analysis of various tradeoffs concerning knowledge and expected rendezvous time. We provide new improved randomized rendezvous algorithms employing more random bits while robot knowledge may include the length of the ring and/or robot speeds and analyse their expected rendezvous time. Table 4.1 summarizes the main results of this chapter.

TABLE 4.1: Summary of performance of randomized rendezvous algorithms for two robots in a ring.

Algorithm	Bits	Exp. Rendezvous Time	Known
Rendezvous_0	1	$\frac{uL}{2(u^2-1)}$	$\emptyset$
$\text{Rendezvous}_1(k)$	$k + 1$	$\frac{(u+2)L}{2u(u+1)} - \frac{L(u-2)}{2u(u^2-1)(u+1)^k}$	$k, \frac{L}{u+1}$
Rendezvous_2	∞	$\frac{(u+2)L}{2u(u+1)}, \text{ for } u \leq 2$	L, u
Rendezvous_2	∞	$\frac{uL}{2(u^2-1)}, \text{ for } u > 2$	L, u
Rendezvous_3	∞	L	L
Rendezvous_4	∞	$\frac{(u+2)L}{(u+1)^2}, \text{ for } u < 3$	L
Rendezvous_5	∞	$\frac{4L}{u+1}$	u

The first column refers to the algorithm, the second to the number of random bits used per robot, and the third is the expected rendezvous time, and the fourth column what each robot knows. Note that algorithms $\text{Rendezvous}_0, \text{Rendezvous}_2$ are shown to be optimal, while Rendezvous_1 is optimal for $u \leq 2$. L is the length of the ring and u and 1, where $u > 1$, the respective speeds of the two robots.

4.2 Single Random Bit

In this section we consider *single random bit* algorithms in which the robots can each flip only a single random bit during the rendezvous search. Consider the following basic

rendezvous algorithm Rendezvous_0 . We show that Rendezvous_0 is optimal among all single random bit algorithms. Note that u, L are unknown to the robots. Further, cw, ccw denote clockwise and counter-clockwise directions of movement, respectively.

Algorithm 4.1: Rendezvous_0

- 1 Select a direction from $\{cw, ccw\}$ uniformly at random
 - 2 Walk until rendezvous
-

Theorem 4.2. *Consider two robots in a bi-directional ring of length L with speeds $u, 1$, respectively, such that $u > 1$.*

1. *Algorithm Rendezvous_0 accomplishes rendezvous in $\frac{uL}{2(u^2-1)}$ expected time.*
2. *Algorithm Rendezvous_0 is optimal in that $\frac{uL}{2(u^2-1)}$ is a lower bound on the expected rendezvous time of any synchronous algorithm that accomplishes rendezvous using at most one random bit per robot.*

Proof. The proof is based on three Lemmas. First we prove the upper bound.

Lemma 4.3. *Rendezvous_0 accomplishes rendezvous in expected time $\frac{uL}{2(u^2-1)}$.*

Proof. Let the two robots with corresponding speeds $u, 1$ be denoted $r(u), r(1)$, respectively. For a given distance, $a \leq L$, between the robots *in the direction* of movement of robot $r(u)$, consider the initial configurations A_a and B_a where the robots are just beginning to move: in the former case, in the same direction, and in the latter case, in opposite directions around the ring. In configuration A_a , the relative speed of the faster robot is $(u - 1)$ and hence, rendezvous occurs at time $\frac{a}{u-1}$, whereas in configuration B_a , its relative speed is $(u + 1)$ and rendezvous occurs at time $\frac{a}{u+1}$.

FIGURE 4.1: Initial configurations A_a and B_a for distance $a \leq L$ between robots

Since configurations A_a and B_a are equally likely, it follows that T_a , the expected time to rendezvous given an initial distance a between the robots is given by:

$$\begin{aligned} T_a &= \frac{1}{2} \left(\frac{a}{u-1} + \frac{a}{u+1} \right) \\ &= \frac{ua}{u^2 - 1} \end{aligned} \tag{4.1}$$

The initial distance a is uniformly distributed in $[0, L)$, hence T , the expected time to rendezvous is:

$$\begin{aligned} T &= \frac{1}{L} \int_0^L \frac{u}{u^2 - 1} a \cdot da \\ &= \frac{uL}{2(u^2 - 1)} \end{aligned}$$

■

Next, we show that algorithm Rendezvous_0 is optimal in the class of rendezvous algorithms that only use one random bit, by generalizing the previous result to a more permissive class of algorithms which provide the robots with additional (but identical) information.

The algorithm $\text{Rendezvous}_0(t)$ in this class is parametrized by a *time threshold* $t \geq 0$ known to both robots. After setting the initial direction at random to either *cw* or

ccw, each robot executes *at most two rounds*. In the first round, it moves in its chosen direction for time t or until rendezvous, *whichever occurs first*. If rendezvous does not occur within the time threshold, then in the second round, each robot independently *reverses* direction and simply keeps walking until rendezvous.

Note that if t is large enough - specifically, if $t \geq L/(u - 1)$ - then the algorithm $\text{Rendezvous}_0(t)$ is identical to Rendezvous_0 since the robots *will* rendezvous in the first round, and the second round never occurs. Hence, our analysis will assume that $t < L/(u - 1)$.

Algorithm 4.2: $\text{Rendezvous}_0(t)$

Input: A time threshold t

- 1 Select a direction from $\{cw, ccw\}$ uniformly at random
 - 2 Walk for time t or until rendezvous
 - 3 **if** no rendezvous by time t **then**
 - 4 Reverse direction
 - 5 Walk until rendezvous
 - 6 **end**
-

Lemma 4.4. *Algorithm Rendezvous_0 has the least expected rendezvous time among all algorithms $\text{Rendezvous}_0(t)$, for $t < L/(u - 1)$. For $t \geq L/(u - 1)$, both algorithms Rendezvous_0 and $\text{Rendezvous}_0(t)$ take the same expected time to rendezvous.*

Proof. From the observation made earlier, we only need to prove the statement for $t < L/(u - 1)$, i.e. when $t(u - 1) < L$. As before, refer to Figure 4.1 depicting the two instantaneous configurations A_a and B_a where the variable a is the distance between the robots in the initial direction of motion chosen by the faster robot, $r(u)$.

Note that if $a \leq t(u - 1)$, then the robots will rendezvous in round 1 since their relative speed is greater than or equal to $(u - 1)$ in both configurations. From Equation (4.1), the expected time to rendezvous is $T_a = ua/(u^2 - 1)$.

Case 1: When $a \in [0, t(u-1)]$, the contribution, T_1 , to the expected completion time of $\text{Rendezvous}_0(t)$ is given by:

$$\begin{aligned}
 T_1 &= \frac{1}{L} \cdot \int_0^{t(u-1)} T_a \cdot da \\
 &= \frac{u}{L(u^2-1)} \int_0^{t(u-1)} a \cdot da \\
 &= \frac{ut^2(u-1)^2}{2L(u^2-1)}
 \end{aligned} \tag{4.2}$$

For larger initial distances, there are two possibilities leading to three case analyses:

- if $t(u+1) \leq L$, then either
 - $a \in (t(u-1), t(u+1)]$ (case 2 below), or
 - $a \in (t(u+1), L]$ (case 3 below)
- Otherwise, $t(u+1) > L$ and the only remaining range is $a \in (t(u-1), L]$ (case 4 below).

Case 2: When $t(u-1) < a \leq t(u+1) \leq L$, the robots will rendezvous in round 1 from configuration B_a at time $a/(u+1)$. However, if they are in configuration A_a initially, the robots will switch to round 2 at time t , reverse directions and after an additional time of t , will be in an instantaneous configuration A_{L-a} (recall that the subscript is the distance in the direction of the faster robot). Hence, they will rendezvous at time $2t + \frac{L-a}{u-1}$. Since A_a and B_a are equally likely, the expected time to rendezvous in this case is

$$\begin{aligned}
 T_a &= \frac{a}{2(u+1)} + \frac{1}{2} \left(2t + \frac{L-a}{u-1} \right) \\
 &= t + \frac{L}{2(u-1)} - \frac{1}{u^2-1} a.
 \end{aligned}$$

The contribution, T_2 , to the expected time of $\text{Rendezvous}_0(t)$ is given by

$$\begin{aligned}
 T_2 &= \frac{1}{L} \cdot \int_{t(u-1)}^{t(u+1)} \left(t + \frac{L}{2(u-1)} - \frac{1}{u^2-1}a \right) \cdot da \\
 &= \frac{2t^2}{L} + \frac{t}{u-1} - \frac{1}{2L(u^2-1)}(t^2(u+1)^2 - t^2(u-1)^2) \\
 &= \frac{2t^2}{L} + \frac{t}{u-1} - \frac{2ut^2}{L(u^2-1)}
 \end{aligned} \tag{4.3}$$

Case 3: When $t(u+1) < a \leq L$, the robots are already too far apart for rendezvous to occur within time t in round 1, hence the robots will always switch to round 2 regardless of the initial configuration. Hence, at time $2t$, they will be back at their *initial positions*, only moving in the opposite direction. In other words, if the robots were in configuration A_a (respectively, B_a) at first, then after time $2t$, they would be in configuration A_{L-a} (respectively, B_{L-a}). This means that with they will rendezvous either at time $2t + \frac{L-a}{u+1}$ with probability $1/2$ or at time $2t + \frac{L-a}{u-1}$ with the remaining probability $1/2$. It follows that the expected time to rendezvous in this case is

$$T_a = 2t + \frac{u}{u^2-1} (L - a)$$

Thus, T_3 , the contribution to the expected time of $\text{Rendezvous}_0(t)$ is given by

$$\begin{aligned}
 T_3 &= \frac{1}{L} \cdot \int_{t(u+1)}^L \left(2t + \frac{u(L-a)}{u^2-1} \right) \cdot da \\
 &= \frac{2t(L-t(u+1))}{L} + \frac{u(L-t(u+1))}{u^2-1} \\
 &\quad - \frac{u}{2L(u^2-1)}(L^2 - t^2(u+1)^2) \\
 &= \frac{uL}{2(u^2-1)} + \frac{u-2}{u-1}t - \frac{2ut^2}{L} + \frac{ut^2(u+1)^2}{2L(u^2-1)} - \frac{2t^2}{L}
 \end{aligned} \tag{4.4}$$

Case 4: When $t(u-1) \leq a \leq L < t(u+1)$, the situation is very similar to the one in

case 2 above: from configuration A_a , the robots will always proceed to round 2 and will rendezvous at time $2t + \frac{L-a}{u-1}$, while from configuration B_a , they will always rendezvous in round 1 at time $a/(u+1)$. The expected time to rendezvous in this case is the same as in case 2, viz.

$$T_a = t + \frac{L}{2(u-1)} - \frac{1}{u^2-1} a;$$

however, this value is now integrated over the range $a \in [t(u-1), L]$ to yield T_4 , the contribution to the expected rendezvous time for $\text{Rendezvous}_0(t)$:

$$\begin{aligned} T_4 &= \frac{1}{L} \cdot \int_{t(u-1)}^L \left(t + \frac{L}{2(u-1)} - \frac{1}{u^2-1} a \right) \cdot da \\ &= \frac{uL}{2(u^2-1)} + \frac{t}{2} - \frac{t^2(u-1)(2u+1)}{2L(u+1)} \end{aligned} \quad (4.5)$$

Putting everything together, we see that T , the total expected time to rendezvous for algorithm $\text{Rendezvous}_0(t)$, can be computed as follows.

- if $t(u+1) \leq L$, then

$$\begin{aligned} T &= T_1 + T_2 + T_3 \\ &= \frac{uL}{2(u^2-1)} + t \left(1 - \frac{ut}{L} \right) \end{aligned} \quad (4.6)$$

Note that the first term is the expected time for algorithm Rendezvous_0 . The second term is non-negative when $t(u+1) \leq L$ since $L \geq ut$.

- if $t(u+1) > L$, then

$$\begin{aligned} T &= T_1 + T_4 \\ &= \frac{uL}{2(u^2-1)} + \frac{t}{2} \left[1 - \frac{t(u-1)}{L} \right] \end{aligned} \quad (4.7)$$

Again, we note that the first term is the expected completion time for algorithm Rendezvous_0 . The second term is positive since $t < L/(u - 1)$.

In both cases, therefore, the expected time to rendezvous in algorithm Rendezvous_0 is less than or equal to the corresponding time for algorithm $\text{Rendezvous}_0(t)$. This concludes the proof of Lemma 4.4. ■

We next generalize the framework: for $k \geq 0$, consider the class of algorithms where the robots try to achieve rendezvous in $k + 1$ rounds using a vector of **round timers**, $\vec{t} := (t_1, t_2, \dots, t_k)$, as follows:

- the first round always begins with a *random choice* of direction of movement,
- before every subsequent round, the robots reverse directions deterministically,
- the i^{th} round runs for time t_i or until rendezvous, whichever occurs first, and
- the last round always runs until rendezvous is accomplished.

Thus after choosing the initial direction randomly, the two robots walk for time t_1 ; if they fail to rendezvous they reverse direction at time t_1 and walk for time t_2 ; if they fail to rendezvous they reverse direction at time $t_1 + t_2$; and so on. If no rendezvous has occurred after k rounds, then in the last round, the robots move without being timed out until they rendezvous.

Consider the family of algorithms,

$$\mathcal{A}_k = \{\text{Rendezvous}_0(\vec{t}) : \vec{t} = (t_1, t_2, \dots, t_k) \text{ with } t_1, t_2, \dots, t_k \geq 0\}$$

Observe that these algorithms are non-adaptive in that the two robots do not change the traversal pattern specified by the algorithm during its execution.

Algorithm 4.3: $\text{Rendezvous}_0(\vec{t} = (t_1, t_2, \dots, t_{k-1}))$

Input: A vector of round timers $\vec{t} = (t_1, t_2, \dots, t_k)$

- 1 Select a direction from $\{cw, ccw\}$ uniformly at random
 - 2 **for** $i = 1, 2, \dots, k$ **do**
 - 3 Walk for time t_i or until rendezvous whichever occurs first
 - 4 **if** no rendezvous by time t_i in the round **then**
 - 5 Reverse direction
 - 6 **end**
 - 7 **end**
 - 8 In round $k + 1$, walk until rendezvous.
-

Lemma 4.5. *The expected rendezvous time of any algorithm in $\mathcal{A}_k$ is not better than algorithm Rendezvous_0 .*

Proof. We will prove this statement by induction. It is easy to see that Lemmas 4.3 and 4.4 respectively establish the basis cases for $k = 0$ and $k = 1$.

Assuming the statement true for the family $\mathcal{A}_k$ for some $k \geq 1$, consider any algorithm $\text{Rendezvous}_0(\vec{t}) \in \mathcal{A}_{k+1}$ where $\vec{t} = (t_1, t_2, \dots, t_{k+1})$. For simplicity, denote algorithm $\text{Rendezvous}_0(\vec{t})$ by A_{k+1} , and define its *refinements*, the algorithms

$$A_k \equiv \text{Rendezvous}_0(t_1, t_2, \dots, t_k) \text{ and } A_{k-1} \equiv \text{Rendezvous}_0(t_1, t_2, \dots, t_{k-1}).$$

Now, for any initial placement and direction of the robots, we have three cases to consider:

If rendezvous occurs before time $t_1 + \dots + t_k$ then all three algorithms A_{k+1}, A_k, A_{k-1} behave the same. Therefore the induction hypothesis applies.

If rendezvous occurs after time $t_1 + \dots + t_k$ but before time $t_1 + \dots + t_k + t_{k+1}$ then algorithms A_{k+1} and A_k behave the same and therefore the induction hypothesis applies again.

If rendezvous occurs after time $t_1 + \dots + t_{k+1}$ then the argument of Lemma 4.4 applies since regardless of the starting position of the robots the expected rendezvous time of algorithm Rendezvous_0 is $\frac{uL}{2(u^2-1)}$. Therefore the expected rendezvous time of algorithm $A_{k+1} \equiv \text{Rendezvous}_0(t_1, t_2, \dots, t_{k+1})$ is at least the expected rendezvous time of algorithm $\text{Rendezvous}_0(t)$, for some $t \geq t_1 + t_2 + \dots + t_{k+1}$. This completes the proof of Lemma 4.5. ■

If we now combine the upper bound and the results of Lemma 4.4 and 4.5 we see that the proof of Theorem 4.2 is complete. ■

4.3 $k + 1$ Random Bits

We saw in Theorem 4.2 that we could not improve the expected rendezvous time of algorithm Rendezvous_0 even if the robots used a vector $\vec{t} := (t_1, \dots, t_k)$ to pre-empt each of the first k rounds if no rendezvous occurred. In this section we show how to improve on algorithm Rendezvous_0 by using additional random bits. We demonstrate that if the choice of directions in the schedule $\vec{t} := (t_1, t_2, \dots, t_k)$ are *randomized*, in the sense that the robots can flip a coin to *re-decide* on the direction at the start of each round, then we can improve on the expected rendezvous time. We define the following class of algorithms.

Observe that $\text{Rendezvous}_1(k, t)$ is a recursive algorithm and during its execution will require that each robot flips a total of $k + 1$ random bits. In the base case, algorithm $\text{Rendezvous}_1(0, t)$ executes our basic random algorithm Rendezvous_0 . Note that the algorithm requires only knowledge of t . A special case of the algorithms $\text{Rendezvous}_1(k, t)$ is when $t = (\frac{L}{u+1}, \dots, \frac{L}{u+1})$. For convenience, we abbreviate

Algorithm 4.4: $\text{Rendezvous}_1(k, t)$ **Input:** Number of rounds k , schedule **vector** $t = (t_1, t_2, \dots, t_k)$

```

1 Select a direction from  $\{cw, ccw\}$  uniformly at random
2 if  $k = 0$  then
3   | Walk until rendezvous
4 end
5 Walk for time  $t_k$  or until rendezvous whichever occurs first
6 if no rendezvous by time  $t_k$  in the round then
7   | Execute  $\text{Rendezvous}_1(k - 1, t)$ 
8 end

```

$\text{Rendezvous}_1\left(k, \left(\frac{L}{u+1}, \dots, \frac{L}{u+1}\right)\right)$ with $\text{Rendezvous}_1(k)$. In the sequel we prove the following theorem.

Theorem 4.6. *Consider two robots in a bi-directional ring of length L with speeds $u, 1$, respectively, such that $u > 1$.*

1. *Algorithm $\text{Rendezvous}_1(k)$ accomplishes rendezvous in*

$$\frac{(u+2)L}{2u(u+1)} - \frac{L(u-2)}{2u(u^2-1)(u+1)^k} \quad (4.8)$$

expected time. In particular, algorithm $\text{Rendezvous}_1(k)$ has expected rendezvous time which is lower than the expected rendezvous time of algorithm Rendezvous_0 if and only if $u < 2$.

2. *Algorithm $\text{Rendezvous}_1(k)$ has optimal expected rendezvous time within the class of rendezvous algorithms $\text{Rendezvous}_1(k, t)$, where $t \geq 0$ and $u \leq 2$.*

Proof. The proof is based on two Lemmas. First we prove the upper bound.

Lemma 4.7. *Algorithm $\text{Rendezvous}_1(k)$ accomplishes rendezvous in*

$$\frac{(u+2)L}{2u(u+1)} \cdot \left(1 - \frac{1}{(u+1)^k}\right) + \frac{1}{(u+1)^k} \cdot \frac{uL}{2(u^2-1)}$$

expected time.

Proof. Let $a \leq L$ be the initial distance of the two robots. Observe that on the one hand, given that the robots are moving in the opposite direction rendezvous occurs in $\frac{a}{u+1}$ time with probability $1/2$. On the other hand, if the robots are moving in the same direction then they will rendezvous in time $\frac{a}{u-1}$, provided that $\frac{a}{u-1} \leq \frac{L}{u+1}$; however, if $\frac{a}{u-1} > \frac{L}{u+1}$ then the robots will walk at most $\frac{L}{u+1}$ time and flip a *new random coin* so as to execute algorithm $\text{Rendezvous}_1(k-1)$.

Therefore the expected time E_k of algorithm $\text{Rendezvous}_1(k)$ until rendezvous will be equal to

$$\begin{aligned} & \frac{1}{2L} \int_0^L \frac{a \cdot da}{u+1} + \frac{1}{2L} \int_0^{\frac{(u-1)L}{u+1}} \frac{a \cdot da}{u-1} \\ & + \frac{1}{2L} \int_{\frac{(u-1)L}{u+1}}^L \left(\frac{L}{u+1} + E_{k-1} \right) \cdot da, \end{aligned} \quad (4.9)$$

where E_{k-1} is the expected rendezvous time of algorithm $\text{Rendezvous}_1(k-1)$.

It remains to calculate the integrals in Formula (4.9); these can be derived easily as follows

$$\begin{aligned} \frac{1}{2L} \int_0^L \frac{a \cdot da}{u+1} &= \frac{L}{4(u+1)} \\ \frac{1}{2L} \int_0^{\frac{(u-1)L}{u+1}} \frac{a \cdot da}{u-1} &= \frac{(u-1)L}{4(u+1)^2} \\ \frac{1}{2L} \int_{\frac{(u-1)L}{u+1}}^L \frac{L}{u+1} \cdot da &= \frac{L}{(u+1)^2}. \end{aligned}$$

Substituting these into Formula (4.9) we can obtain easily the following expression for the expected value E_k .

$$\begin{aligned}
E_k &= \frac{1}{2L} \int_0^L \frac{a \cdot da}{u+1} + \frac{1}{2L} \int_0^{\frac{(u-1)L}{u+1}} \frac{a \cdot da}{u-1} \\
&\quad + \frac{1}{2L} \int_{\frac{(u-1)L}{u+1}}^L \left(\frac{L}{u+1} + E_{k-1} \right) \cdot da \\
&= \frac{L}{4(u+1)} + \frac{(u-1)L}{4(u+1)^2} + \frac{L}{(u+1)^2} \\
&\quad + \frac{1}{2L} \int_{\frac{(u-1)L}{u+1}}^L E_{k-1} \cdot da \\
&= \frac{(u+2)L}{2(u+1)^2} + \frac{1}{2L} \int_{\frac{(u-1)L}{u+1}}^L E_{k-1} \cdot da \\
&= \frac{(u+2)L}{2(u+1)^2} + \frac{1}{u+1} E_{k-1}, \tag{4.10}
\end{aligned}$$

where we derived the last identity using the fact that $L - \frac{(u-1)L}{u+1} = \frac{2L}{u+1}$.

Lets use the abbreviation $C := \frac{(u+2)L}{2(u+1)^2}$. As a consequence of Formula (4.10) we see that

$$\begin{aligned}
E_k &= C \left(1 + \frac{1}{u+1} + \cdots + \frac{1}{(u+1)^{k-1}} \right) \\
&\quad + \frac{1}{(u+1)^k} \cdot \frac{uL}{2(u^2-1)} \\
&= \frac{(u+2)L}{2(u+1)^2} \cdot \left(\frac{1 - \frac{1}{(u+1)^k}}{1 - \frac{1}{u+1}} \right) + \frac{1}{(u+1)^k} \cdot \frac{uL}{2(u^2-1)} \\
&= \frac{(u+2)L}{2u(u+1)} \cdot \left(1 - \frac{1}{(u+1)^k} \right) + \frac{1}{(u+1)^k} \cdot \frac{uL}{2(u^2-1)}
\end{aligned} \tag{4.11}$$

Rearranging the terms proves Equation (4.8).

Now we compare the last term to $\frac{uL}{2(u^2-1)}$. Observe that

$$\begin{aligned}
 E_k \leq E_0 &\Leftrightarrow \frac{(u+2)L}{2(u+1)^2} \cdot \left(1 - \frac{1}{(u+1)^k}\right) \\
 &\quad + \frac{uL}{2(u^2-1)(u+1)^k} \leq \frac{uL}{2(u^2-1)} \\
 &\Leftrightarrow \left(1 - \frac{1}{(u+1)^k}\right) \cdot \frac{(u+2)L}{2(u+1)^2} \\
 &\quad \leq \left(1 - \frac{1}{(u+1)^k}\right) \cdot \frac{uL}{2(u^2-1)} \\
 &\Leftrightarrow \frac{(u+2)L}{2u(u+1)} \leq \frac{uL}{2(u^2-1)} \\
 &\Leftrightarrow u \leq 2.
 \end{aligned}$$

■

As a corollary of the proof above we also obtain the following result concerning monotonicity of the expected rendezvous times for the algorithm presented in Theorem 4.6.

Corollary 4.8 (Monotonicity of Rendezvous Time). *Let E_k denote the expected rendezvous time of algorithm $\text{Rendezvous}_1(k)$, for $k \geq 0$.*

1. *For $u < 2$, the sequence $\{E_i : 0 \leq i \leq k\}$ is monotone decreasing.*
2. *For $u = 2$, the sequence $\{E_i : 0 \leq i \leq k\}$ is constant.*
3. *For $u > 2$, the sequence $\{E_i : 0 \leq i \leq k\}$ is monotone increasing.*

Proof. This is a consequence of the proof of Theorem 4.6 (see Identity (4.11)). Observe that for $k \geq 1$ we have that

$$\begin{aligned} E_k - E_{k-1} &= \left(\frac{(u+2)L}{2u(u+1)} - \frac{uL}{2(u^2-1)} \right) \cdot \left(\frac{1}{(u+1)^{k-1}} - \frac{1}{(u+1)^k} \right) \\ &= \frac{(u-2)L}{2u(u^2-1)} \cdot \left(\frac{1}{(u+1)^{k-1}} - \frac{1}{(u+1)^k} \right). \end{aligned}$$

The statement of the corollary follows immediately. ■

Next we prove the optimality of $\text{Rendezvous}_1(k)$.

Lemma 4.9. *$\text{Rendezvous}_1(k)$ has optimal expected rendezvous time within the class of algorithms $\text{Rendezvous}_1(k, t)$, where $t \geq 0$ and $u \leq 2$.*

Proof. Our proof will be inductive on the number of random bits, k . Let $a \leq L$ be the initial distance of the two robots. We assume that $u \in (1, 2]$, since this is the interval on which the Lemma is defined. Let t be the time threshold for algorithm $\text{Rendezvous}_1(k, (\frac{L}{u+1}, \frac{L}{u+1}, \dots, t))$ and let $E_{t,k}$ denote its expected rendezvous time. We can estimate the expected rendezvous time easily as follows

$$\begin{aligned} E_{t,k} &= \frac{1}{2L} \int_0^{\min\{L, t(u+1)\}} \frac{a \cdot da}{u+1} \\ &\quad + \frac{1}{2L} \int_{\min\{L, t(u+1)\}}^L (t + E_{k-1}) \cdot da \end{aligned} \tag{4.12}$$

$$+ \frac{1}{2L} \int_0^{t(u-1)} \frac{a \cdot da}{u-1} + \frac{1}{2L} \int_{t(u-1)}^L (t + E_{k-1}) \cdot da, \tag{4.13}$$

where above we assume that $t \leq \frac{L}{u-1}$ and $E_{t,0}$ is the expected rendezvous time of algorithm Rendezvous_0 . We know that this is true for $k \leq 1$ (by Theorem 4.2),

and so we only need to prove the inductive step. There are now two cases to consider depending on the value of t .

Case 1: When $t \leq \frac{L}{u+1}$, the Formulas (4.12) and (4.13) are transformed into the following recurrence

$$\begin{aligned}
 E_{t,k+1} &= \frac{t^2(u+1)}{4L} + \frac{(t+E_k)(L-t(u+1))}{2L} \\
 &\quad + \frac{t^2(u-1)}{4L} + \frac{(t+E_k)(L-t(u-1))}{2L} \\
 &= \frac{t^2u}{2L} + \frac{L-tu}{L} (t+E_k) \\
 &= -\frac{u}{2L}t^2 + t + \left(1 - \frac{tu}{L}\right) E_k.
 \end{aligned} \tag{4.14}$$

We want to show, for all $t \in [0, \frac{L}{u+1}]$, that

$$\begin{aligned}
 E_{k+1} &\leq -\frac{u}{2L}t^2 + t + \left(1 - \frac{tu}{L}\right) E_k \\
 \Leftrightarrow \frac{(u+2)L}{2(u+1)^2} + \frac{1}{u+1}E_k &\leq -\frac{u}{2L}t^2 + t + \left(1 - \frac{tu}{L}\right) E_k \\
 \Leftrightarrow E_k \left(\frac{1}{u+1} + \frac{tu}{L} - 1 \right) &\leq -\frac{u}{2L}t^2 + t - \frac{(u+2)L}{2(u+1)^2}
 \end{aligned}$$

We will divide both sides by $(\frac{1}{u+1} + \frac{tu}{L} - 1)$, but we note that since $t \leq \frac{L}{u+1}$, this expression is always less than or equal to zero, and so dividing and simplifying we obtain

$$E_k \geq \frac{-tu^2 + Lu - tu + 2L}{2u(u+1)} \tag{4.15}$$

By Corollary 4.8, since $u \leq 2$ the sequence of E_i is monotonically non-increasing, therefore we want to show that

$$\begin{aligned}
 E_k &\geq \lim_{i \rightarrow +\infty} E_i \geq \frac{-tu^2 + Lu - tu + 2L}{2u(u+1)} \\
 &\Leftrightarrow \frac{(u+2)L}{2u(u+1)} \geq \frac{-tu^2 + Lu - tu + 2L}{2u(u+1)} \\
 &\Leftrightarrow (u+2)n \geq -tu^2 + Lu - tu + 2L \\
 &\Leftrightarrow 0 \geq -tu(u+1)
 \end{aligned}$$

Therefore, (4.15) holds for any positive values of t and u .

Case 2: When $t \in [\frac{L}{u+1}, \frac{L}{u-1}]$, Formulas (4.12) and (4.13) are transformed into the following recurrence

$$\begin{aligned}
 E_{t,k+1} &= \frac{1}{2L} \int_0^L \frac{a \cdot da}{u+1} + \frac{1}{2L} \int_0^{t(u-1)} \frac{a \cdot da}{u-1} \\
 &\quad + \frac{1}{2L} \int_{t(u-1)}^L (t + E_{t,k}) \cdot da \\
 &= \frac{L}{4(u+1)} + \frac{t^2(u-1)}{4L} + \frac{(t + E_{t,k})(L - t(u-1))}{2L} \\
 &= \frac{t}{2} - \frac{u-1}{4L} t^2 + \frac{L}{4(u+1)} + \left(\frac{1}{2} - \frac{t(u-1)}{2L} \right) E_{t,k}. \tag{4.16}
 \end{aligned}$$

We want to show, for all $t \in (\frac{L}{u+1}, \frac{L}{u-1}]$, that

$$\begin{aligned}
E_{k+1} &\leq \frac{t}{2} - \frac{u-1}{4L}t^2 + \frac{L}{4(u+1)} + \left(\frac{1}{2} - \frac{t(u-1)}{2L}\right) E_{t,k} \\
&\Leftrightarrow \frac{(u+2)L}{2(u+1)^2} + \frac{1}{u+1} E_k \\
&\leq \frac{t}{2} - \frac{u-1}{4L}t^2 + \frac{L}{4(u+1)} + \left(\frac{1}{2} - \frac{t(u-1)}{2L}\right) E_k \\
&\Leftrightarrow E_k \left(\frac{1}{u+1} + \frac{t(u-1)}{2L} - \frac{1}{2} \right) \\
&\leq \frac{t}{2} - \frac{u-1}{4L}t^2 + \frac{L}{4(u+1)} - \frac{(u+2)L}{2(u+1)^2}
\end{aligned}$$

We will divide both sides by $\left(\frac{1}{u+1} + \frac{t(u-1)}{2L} - \frac{1}{2}\right)$, and we note that since $t \geq \frac{L}{u+1}$, this expression is always greater than or equal to zero, and so dividing we obtain

$$E_k \leq \frac{\frac{t}{2} - \frac{u-1}{4L}t^2 + \frac{L}{4(u+1)} - \frac{(u+2)L}{2(u+1)^2}}{\frac{1}{u+1} + \frac{t(u-1)}{2L} - \frac{1}{2}} \quad (4.17)$$

By Corollary 4.8, since $u \leq 2$ the sequence of E_i is monotonically non-increasing, therefore we want to show that

$$\begin{aligned}
E_{k+1} &\leq E_0 \leq \frac{\frac{t}{2} - \frac{u-1}{4L}t^2 + \frac{L}{4(u+1)} - \frac{(u+2)L}{2(u+1)^2}}{\frac{1}{u+1} + \frac{t(u-1)}{2L} - \frac{1}{2}} \\
\Leftrightarrow \frac{uL}{2(u^2-1)} &\leq \frac{\frac{t}{2} - \frac{u-1}{4L}t^2 + \frac{L}{4(u+1)} - \frac{(u+2)L}{2(u+1)^2}}{\frac{1}{u+1} + \frac{t(u-1)}{2L} - \frac{1}{2}} \\
\Leftrightarrow 0 &\leq \frac{-tu^2 + 3L + t}{2(u^2-1)} \\
\Rightarrow 0 &\leq -tu^2 + 3L + t \text{ (since } u > 1) \\
\Rightarrow u &\leq 2 \text{ (since } t \leq \frac{L}{u-1})
\end{aligned}$$

Therefore, (4.17) holds so long as $u \leq 2$. ■

Combining the results of Lemmas 4.7 and 4.9 completes the proof of Theorem 4.6. ■

4.4 Rendezvous and Knowledge

There are two interesting questions worth investigating that concern knowledge of the robots and relative sizes of the speeds. The basic algorithm Rendezvous_0 requires no knowledge either of the ring size L or speeds of the robots. $\text{Rendezvous}_1(k)$ requires only knowledge of the ratio $\frac{L}{u+1}$. One may wonder, how about if this is not the case and additional knowledge is provided? Another interesting point here is the relative sizes of the speeds u and 1. The expected rendezvous time of algorithms Rendezvous_0 and $\text{Rendezvous}_2(k)$ is inversely proportional to $u - 1$. In particular, if u is arbitrarily close to 1 it is natural to ask whether better algorithms are possible which (of course) may depend on additional knowledge that the robots may have. We investigate these and other possibilities in this section.

4.4.1 Unknown u and L

This is precisely accomplished by algorithm Rendezvous_0 . The expected rendezvous time is $\frac{uL}{2(u^2-1)}$ and this has been shown to be optimal among all single random bit rendezvous algorithms even if the robots have knowledge of the speeds $u, 1$. Theorem 4.2 also explains the conditions under which this randomized algorithm has optimal expected time.

4.4.2 Known u and L

Consider the following class of algorithms which we will use in our proof of Theorem 4.10.

Algorithm 4.5: $\text{Rendezvous}_2(t)$

Input: Time threshold t

```

1 repeat
2   | Select a direction from  $\{cw, ccw\}$  uniformly at random
3   | Walk for time  $t$  or until rendezvous whichever occurs first
4 until rendezvous

```

Theorem 4.10. *Consider two robots in a bi-directional ring of length L with speeds $u, 1$, respectively, such that $u > 1$. In the class of rendezvous algorithms $\text{Rendezvous}_2(t)$, where $t \geq 0$,*

1. *$\text{Rendezvous}_2\left(\frac{L}{u+1}\right)$ accomplishes rendezvous in optimal expected time, provided that $u \leq 2$, and*
2. *$\text{Rendezvous}_2\left(\frac{L}{u-1}\right)$ accomplishes rendezvous in optimal expected time, provided that $u \geq 2$.*

Moreover, the expected rendezvous time in the former case is equal to $\frac{(u+2)L}{2u(u+1)}$ and in the latter case $\frac{uL}{2(u^2-1)}$.

Proof. The first part of the Theorem follows immediately from taking the limit of Theorem 4.6.

$$\lim_{k \rightarrow +\infty} \frac{(u+2)L}{2u(u+1)} - \frac{L(u-2)}{2u(u^2-1)(u+1)^k} = \frac{(u+2)L}{2u(u+1)}$$

The second part of the Theorem will be proven by contradiction. We begin by noting that any $t \geq \frac{L}{u-1}$ will equivalently result in the execution of Rendezvous_0 . Therefore, if the Theorem is incorrect, then there must be a $t \in [0, \frac{L}{u-1}]$ which results in expected rendezvous time strictly better than $\frac{uL}{2(u^2-1)}$.

Let $a \leq L$ be the initial distance of the two robots. Let t be the time threshold for algorithm $\text{Rendezvous}_3(t)$ and let $E_{t,k}$ denote its expected rendezvous time. We calculate the expected running time as

$$\begin{aligned} E_t = & \frac{1}{2L} \int_0^{\min\{L, t(u+1)\}} \frac{a \cdot da}{u+1} \\ & + \frac{1}{2L} \int_{\min\{L, t(u+1)\}}^L (t + E_t) \cdot da \end{aligned} \quad (4.18)$$

$$+ \frac{1}{2L} \int_0^{t(u-1)} \frac{a \cdot da}{u-1} + \frac{1}{2L} \int_{t(u-1)}^L (t + E_t) \cdot da. \quad (4.19)$$

There are two cases to consider.

Case 1: When $t \leq \frac{L}{u+1}$, the Formulas (4.18) and (4.19) are transformed into the following recurrence

$$\begin{aligned}
 E_t &= \frac{t^2(u+1)}{4L} + \frac{(t+E_t)(L-t(u+1))}{2L} \\
 &\quad + \frac{t^2(u-1)}{4L} + \frac{(t+E_{t,k})(L-t(u-1))}{2L} \\
 &= \frac{t^2 u}{2L} + \frac{L-tu}{L} (t+E_t) \\
 &= -\frac{u}{2L} t^2 + t + \left(1 - \frac{tu}{L}\right) E_t.
 \end{aligned}$$

Solving the recurrence and we obtain $E_t = \frac{L}{u} - \frac{t}{2}$. We then need to show that there exists a t such that

$$\begin{aligned}
 E_t &< \frac{uL}{2(u^2-1)} \\
 \Leftrightarrow \frac{L}{u} - \frac{t}{2} &< \frac{uL}{2(u^2-1)} \\
 \Rightarrow \frac{L}{u} - \frac{L}{2(u-1)} &< \frac{uL}{2(u^2-1)}, \text{ since } t \Leftrightarrow \leq \frac{L}{u-1} \\
 \Leftrightarrow \frac{L(u-2)}{2u(u-1)} &< \frac{uL}{2(u^2-1)} \\
 \Leftrightarrow u &< 2
 \end{aligned} \tag{4.20}$$

However $u \geq 2$, therefore this is a contradiction and in this case no such t exists.

Case 2 When $t \in [\frac{L}{u+1}, \frac{L}{u-1}]$, Formulas (4.18) and (4.19) are transformed into the following recurrence

$$\begin{aligned}
 E_t &= \frac{1}{2L} \int_0^L \frac{a \cdot da}{u+1} + \frac{1}{2L} \int_0^{t(u-1)} \frac{a \cdot da}{u-1} \\
 &\quad + \frac{1}{2L} \int_{t(u-1)}^L (t + E_t) \cdot da \\
 &= \frac{L}{4(u+1)} + \frac{t^2(u-1)}{4L} + \frac{(t + E_t)(L - t(u-1))}{2L} \\
 &= \frac{t}{2} - \frac{u-1}{4L} t^2 + \frac{L}{4(u+1)} + \left(\frac{1}{2} - \frac{t(u-1)}{2L} \right) E_t.
 \end{aligned}$$

Solving the recurrence and we obtain

$$E_t = \frac{2Lt(u+1) - t^2(u^2-1) + L^2}{2(u+1)(L + t(u-1))}$$

We then need to show that there exists a t such that

$$\begin{aligned}
 &\frac{uL}{2(u^2-1)} > E_t \\
 \Leftrightarrow &\frac{uL}{2(u^2-1)} > \frac{2Lt(u+1) - t^2(u^2-1) + L^2}{2(u+1)(L + t(u-1))} \\
 &\text{since } t \leq \frac{L}{u-1}: \\
 \Rightarrow &\frac{uL}{2(u^2-1)} > \frac{2L \left(\frac{L}{u-1} \right) (u+1) - \left(\frac{L}{u-1} \right)^2 (u^2-1) + L^2}{2(u+1)(L + \left(\frac{L}{u-1} \right) (u-1))} \\
 \Leftrightarrow &\frac{uL}{2(u^2-1)} > \frac{L(u+2)}{2u(u+1)} \\
 \Leftrightarrow &2 > u
 \end{aligned}$$

(4.21)

However $u \geq 2$, therefore this is a contradiction and in this case no such t exists. ■

4.4.3 Unknown u and Known L

In this setting, the robots do not know their own speed (which is either u or 1), nor what is the magnitude of u .

Algorithm 4.6: *Rendezvous₃*

Input: Size of ring L

```

1 repeat
2   | Select a direction from  $\{cw, ccw\}$  uniformly at random
3   | Walk for time  $\frac{L}{2}$  or until rendezvous whichever occurs first
4 until rendezvous

```

Theorem 4.11. *Rendezvous₃ accomplishes rendezvous in expected time L .*

Proof. A simple analysis of this algorithm is based on the geometric distribution with probability of success $1/2$. Therefore the expected rendezvous time will be at most L .

■

However, a better upper bound can be proved based on the fact that one of the robots has speed $u > 1$. Consider the following algorithm, where as a notational convenience we use *Rendezvous₄*(0) to be identical to algorithm *Rendezvous₀*.

Algorithm 4.7: *Rendezvous₄*(k)

Input: Ring size L , number of rounds k

```

1 if  $k = 0$  then
2   | Execute Rendezvous0
3 end
4 Select a direction from  $\{cw, ccw\}$  uniformly at random
5 Walk for time  $\frac{L}{2}$  or until rendezvous whichever occurs first
6 if no rendezvous then
7   | Execute Rendezvous4( $k - 1$ )
8 end

```

Observe that $\text{Rendezvous}_4(k)$ is a recursive algorithm and during its execution will require that each robot flips a total of $k + 1$ random bits. In the last “random” step algorithm $\text{Rendezvous}_4(k)$ executes our basic random algorithm Rendezvous_0 . We can prove the following theorem.

Theorem 4.12. *Algorithm $\text{Rendezvous}_4(k)$ accomplishes rendezvous in expected time*

$$\begin{cases} \frac{uL}{2(u^2-1)} & \text{if } u \geq 3 \\ \frac{(u+2)L}{(u+1)^2} & \text{if } u < 3 \end{cases}$$

asymptotically in k .

Proof. This is pretty much the same as the recurrence in Theorem 4.6 (see Equation (4.10)) but with $\frac{L}{u+1}$ being replaced with $\frac{L}{2}$. Let $a \leq L$ be the initial distance of the two robots. Observe that on the one hand, given that the robots are moving in the opposite direction rendezvous occurs in $\frac{a}{u+1}$ time with probability $1/2$. On the other hand, if the robots are moving in the same direction then they will rendezvous in time $\frac{a}{u-1}$, provided that $\frac{a}{u-1} \leq \frac{L}{2}$; however, if $\frac{a}{u-1} > \frac{L}{2}$ then the robots will walk at most $\frac{L}{2}$ time and flip a *new random coin*. The resulting recurrence is now

$$\begin{aligned} E_k = & \frac{1}{2L} \int_0^L \frac{a \cdot da}{u+1} + \frac{1}{2L} \int_0^{\frac{(u-1)L}{2}} \frac{a \cdot da}{u-1} \\ & + \frac{1}{2L} \int_{\frac{(u-1)L}{2}}^L \left(\frac{L}{2} + E_{k-1} \right) \cdot da \end{aligned}$$

There are two cases to consider depending on the speed u .

Case 1: When $u \geq 3$, observe that the last term in the sum above is 0 when $u \geq 3$, in which case we have that

$$\begin{aligned} E_k &= \frac{1}{2L} \int_0^L \frac{a \cdot da}{u+1} + \frac{1}{2L} \int_0^L \frac{a \cdot da}{u-1} \\ &= \frac{uL}{2(u^2-1)}. \end{aligned}$$

Case 2: When $u < 3$, observe that

$$\begin{aligned} E_k &= \frac{L}{4(u+1)} + \frac{(u-1)L}{8} + \frac{(3-u)L}{8} \\ &\quad + \frac{1}{2L} \int_{\frac{(u-1)L}{2}}^L E_{k-1} \cdot da \\ &= \frac{(u+2)L}{4(u+1)} + \frac{1}{2L} \int_{\frac{(u-1)L}{2}}^L E_{k-1} \cdot da \\ &= \frac{(u+2)L}{4(u+1)} + \frac{3-u}{4} E_{k-1} \end{aligned}$$

This gives the value

$$\begin{aligned} E_k &= \frac{(u+2)L}{4(u+1)} \cdot \sum_{i=0}^{k-1} \left(\frac{3-u}{4} \right)^i + \left(\frac{3-u}{4} \right)^k \cdot \frac{uL}{2(u^2-1)} \\ &\rightarrow \frac{(u+2)L}{4(u+1)} \cdot \frac{4}{u+1} \text{ (as } k \rightarrow \infty) \\ &= \frac{(u+2)L}{(u+1)^2}, \end{aligned}$$

thus proving the theorem. ■

4.4.4 Known u and Unknown L

In this setting, the robots do not know their own speed (which is either u or 1), but they do know one robot has speed u and the other 1.

Algorithm 4.8: *Rendezvous₅***Input:** Speed u

```

1  $r \leftarrow 1$ 
2 repeat
3   repeat twice
4     Select a direction from  $\{cw, ccw\}$  uniformly at random
5     Walk for time  $\frac{r}{u+1}$  or until rendezvous whichever occurs first
6   end
7    $r \leftarrow 2r$ 
8 until rendezvous

```

The following theorem analyses this algorithm.

Theorem 4.13. *Rendezvous₅ accomplishes rendezvous in expected time $\frac{4L}{u+1}$.*

Proof. The algorithm runs for expected $\frac{1}{u+1} + \frac{2}{u+1} + \dots + \frac{2^m}{u+1}$ steps for $2^m < L$. After the first time m such that $2^m \geq L$ the algorithm will require 2 expected iterations for the robots to move in the opposite direction. Each such iteration takes $\frac{L}{u+1}$ expected time. It follows that the expected rendezvous time is at most

$$\frac{1}{u+1} + \frac{2}{u+1} + \dots + \frac{2^m}{u+1} \frac{2L}{u+1} \leq \frac{4L}{u+1},$$

which proves the theorem. ■

4.5 Conclusion

In this chapter, we provided several randomized rendezvous algorithms for two robots having different speeds $u, 1$, where $u > 1$, in a uniform anonymous setting (robots execute identical algorithms and are unaware of their own speed) and equipped with identical chronometers, in a synchronous, bi-directional ring of length L .

We presented Rendezvous_0 , and showed that it was optimal among all randomized algorithms employing a single random bit during their execution, and in fact optimal among all randomized algorithms employing $k + 1$ random bits, if $u > 2$.

We also proposed $\text{Rendezvous}_1(k)$ and proved that it optimal among all randomized algorithms employing $k + 1$ random bits, if $u \leq 2$.

Finally, we explored various tradeoffs concerning knowledge and expected rendezvous time.

Chapter 5

Evacuation

5.1 Introduction

The search problem we study in this chapter involves the evacuation of robots from a domain. Whereas the goal of traditional search problems is to minimize the time required for searchers to find one or more targets hidden in a domain, the goal of the problem studied in this chapter is to minimize the time required for every searcher to reach a target (referred to as an *exit*). We can think of many instances where this problem is natural, beginning with the problem of evacuating people from a smoke-filled room. Other potential problem instances include: cows trying to escape from a pasture, robots needing to find a charging station, and boats trying to find safe harbour during a storm. In all these cases, it is not sufficient for the group of searchers to simply find a target; they must all reach a target in order to achieve the goal of the problem.

While *searching* for exits is a core part of the problem, the cost of a solution is measured not by the time required for an exit to be found, but by the time required for both robots

¹This chapter contains results from the submitted paper: J. Czyzowicz, S. Dobrev, K. Georgiou, E. Kranakis, and F. MacQuarrie. *Evacuating two robots from multiple unknown exits in a circle*.

FIGURE 5.1: Example of an evacuation problem with 4 exits.

Robot is represented by $\bullet$, an exit by $\square$, and a starting position by $\times$.

to reach an exit. Therefore, we want to take into consideration both the time it takes to find an exit, and the additional time required for evacuation after an exit has been found. This means we may wish to sacrifice some efficiency in terms of finding an exit so that the robots are always relatively close to some exit. Because of the interplay between these two goals, this chapter yields interesting results even though the domain is rather simple. We believe these results provide a solid foundation for the investigation of this problem on more complex domains.

5.1.1 Preliminaries

Consider a circle of unit length, i.e. of radius $1/2\pi$. Distributed on the circle are k exits. Two autonomous mobile robots are placed on the circle. The goal of the *evacuation problem* is to give an algorithm for the robots which minimizes the time required for both robots to reach any exit, in the worst case. Note that the robots do not need to reach the same exit.

We will use the following notation throughout this chapter (see Figure 5.1). The shortest arc between the starting positions of the two robots will be denoted by L . For a placement of $k \geq 2$ exits, we let D denote the length of the *longest arc* defined by any

two consecutive exits. In other words, when $k = 2$, D is not the distance of the two exits, rather its complement with respect to the disc perimeter (which is 1).

5.1.2 Evacuation Problems

We consider the following evacuation problems in this thesis:

Evacuation Problem with Fixed Initial Distance

Two robots are placed at a given arc distance L on a circle of perimeter length 1 containing k exits, where the maximum arc length between any two consecutive exits is D . Find an algorithm which takes D and L as input and minimizes the worst case time for the all robots to evacuate the circle.

Evacuation Problem with Chosen Initial Distance

Two robots are placed at a chosen arc distance L on a circle of perimeter length 1 containing k exits, where the maximum arc length between any two consecutive exits is D . Find an algorithm which takes D as input, but may choose L , which minimizes the worst case time for the all robots to evacuate the circle.

5.1.3 Robot Model for Evacuation

We add the following capabilities to our basic robot model in order to execute the evacuation algorithms.

TABLE 5.1: Costs of $\text{Evacuate}_L(D)$ for the Evacuation Problem with Fixed Initial Distance.

$D \backslash L$	0	$(0, \frac{1}{2}]$
1	$\frac{3}{4}$	$\frac{3}{4} + \frac{L}{2}$
$(1, \frac{1}{k})$	$\frac{3}{4} \cdot D$	$\frac{3}{4} \cdot D - \frac{L}{2} \leq C_{opt}(D, L) \leq \frac{3}{4} \cdot D + \frac{L}{2}$
$\frac{1}{k}$	$\frac{3}{4} \cdot \frac{1}{k}$	$\frac{3}{4} \cdot \frac{1}{k} + \frac{\delta}{2}$

All results are optimal (i.e. optimal cost, $C_{opt} = \dots$) except for where bounds are given. The upper and lower bounds are parametrized, and should be interpreted as bounding only the specific instance of the problem for a given input of D and L . Note that when $D = 1$, all exits are colocated.

TABLE 5.2: Costs of $\text{Evacuate}(D)$ for the Evacuation Problem with Chosen Initial Distance.

D	L	$Cost$
$(\frac{6}{7}, 1]$	$D - 1$	$\frac{5}{4} \cdot D - \frac{1}{2}$
$(\frac{4}{5}, \frac{6}{7}]$	$\frac{5}{2} \cdot D - 2$	$1 - \frac{D}{2}$
$[\frac{1}{k}, \frac{4}{5}]$	0	$\frac{3}{4} \cdot D$

identical speeds

The robots both move (and search) with the same speed. For notational convenience, we assume that this is a unit speed.

wireless communication

The robots are able to instantaneously communicate any number of bits.

sense of direction

The robots have a shared sense of direction.

knowledge

We assume that the robots have a map of the environment – knowledge of the

length of the circle as well as the relative distance between any two consecutive exits which we assume are indistinguishable. However, we assume that robots do not know the exact exit positions on the circle.

5.1.4 Outline and Results

In Section 5.2, we consider the problem for a single exit, and give an algorithm which achieves an optimal worst case evacuation cost of $\frac{3}{4} + \frac{L}{2}$, where L is the initial distance between the robots.

In Section 5.3, we explore the problem with k exits. We begin by showing an optimal algorithm when the robots begin colocated which achieves a worst case evacuation cost of $\frac{3}{4} \cdot D$, where D is the longest distance between any two exits. We then give, for the case when the exits are evenly spaced, an optimal algorithm which achieves a worst case evacuation cost of $\frac{3}{4} \cdot \frac{1}{k} + \frac{\delta}{2}$, where

$$\delta = \begin{cases} L \bmod \frac{1}{k} & : L \bmod \frac{1}{k} \leq \frac{1}{2k} \\ \frac{1}{k} - L \bmod \frac{1}{k} & : L \bmod \frac{1}{k} > \frac{1}{2k} \end{cases}$$

Finally, in Section 5.4 we further explore the problem with k exits, but consider the case with arbitrary D and L . We begin by giving upper and lower bounds of $\frac{3}{4} \cdot D + \frac{L}{2}$ and $\frac{3}{4} \cdot D - \frac{L}{2}$, respectively, for the Evacuation Problem with Fixed Initial Distance, when D and L are inputs. We then finish the section by giving an algorithm for the Evacuation Problem with Chosen Initial Distance which takes D as input and chooses L . This algorithm is able to match the lower bound of the Evacuation Problem with Fixed Initial Distance for some values of D and L . We conjecture that our choice of L

is optimal for these values of D , which would imply a lower bound for these values of D .

The results of this chapter for the Evacuation Problem with Fixed Initial Distance are summarized in Table 5.1. The results of this chapter for the Evacuation Problem with Fixed Chosen Distance are summarized in Table 5.2. We note that as was the case for the previous problems, the results obtained can easily be scaled for a non-unit length circle length.

5.2 Evacuation with a Single Exit

We begin by considering the problem with only one exit. To further simplify things, we begin this section by considering the simplest of cases: when there is only one exit, and where the robots begin colocated.

Lemma 5.1. *The evacuation problem with a single exit and where the two robots begin colocated ($L = 0$), can be solved with an optimal worst case cost of $\frac{3}{4}$.*

Proof. We first show that the problem can be solved with worst case cost $\frac{3}{4}$. Consider the following simple algorithm (more formally described as Evacuate_0): The robots begin searching in opposite directions until one of them finds the exit, at which point it will broadcast the location of the exit to the other robot. The other robot then calculates and moves along the shortest route to the exit.

We analyse this algorithm as follows: suppose the robots each move distance x before one of them finds the exit (see Figure 5.2). The cost of the algorithm is determined by the time taken by the second robot to exit, and therefore $C = x + \min(2x, 1 - 2x)$, or $C = \min(3x, 1 - x)$. We note that $3x$ is monotone increasing and $1 - x$ is

Algorithm 5.1: Evacuate_0

```

1 Robots orient so that they are “facing” away from each other
2 repeat
3   | SEARCH
4 until exit is found
5 if this robot found the exit then
6   | EXIT
7 else
8   | Let  $x$  be the distance searched before the exit was found
9   | if  $3x \leq 1 - x$  then
10    | Reverse direction
11  end
12  repeat
13    | WALK
14  until exit is reached
15  EXIT
16 end

```

FIGURE 5.2: Single exit evacuation of colocated robots.

monotone decreasing with respect to x , and therefore the choice of x which maximizes $\min(3x, 1 - x)$, and therefore the total cost, will be when $3x = 1 - x$, or when $x = 1/4$. The worst case cost for our algorithm will therefore be $3/4$.

We now show that the algorithm described above is optimal. We define the term *unsearched antipodal pair* to mean a pair of antipodal points on the circle, neither of which has been searched. For any algorithm which solves the problem described in the lemma, during the execution of its worst case there is a point in time t where one of the

points of the last remaining unsearched antipodal pair becomes searched by one of the robots, which we will call r_s . The exit is then “placed” at the point opposite the one just searched. Clearly the time required for r_s to exit is at least $t + 1/2$. At time t , the points searched by both robots are of length at most $2t$. Since no other unsearched antipodal pair exists at time t , it must be the case that $2t \geq 1/2$. Therefore $t \geq 1/4$, which means that the worst case cost of any algorithm is at least $3/4$. ■

We now consider the general problem with one exit: where the robots are forced to start at an arbitrary distance L .

Theorem 5.2. *The evacuation problem with a single exit and where the robots begin at distinct starting points at distance $L \in (0, \frac{1}{2}]$, can be solved with an optimal worst case cost of $\frac{3}{4} + \frac{L}{2}$.*

Proof. It is easy to show that a worst case cost of $\frac{3}{4} + \frac{L}{2}$ is feasible. The algorithm achieving this is simply to have the robots walk towards each other until they meet – which takes time $L/2$ – and then to execute the algorithm from Lemma 5.1. Therefore the worst case cost is $\frac{3}{4} + \frac{L}{2}$.

Note that if there is an unsearched antipodal pair at time $t = \frac{1}{4} + \frac{L}{2}$, the theorem follows using the same arguments as in the proof of Lemma 5.1. Hence, assume that there is no unsearched antipodal pair at time t . Observe also that since $L \leq \frac{1}{2}$, $t \leq \frac{1}{2}$. Since by time t each robot can search only an area of size at most t , then if at time t a robot is located in an area searched by the other robot, its antipodal point is unsearched and so the adversary can place the exit there to force the evacuation time of $\frac{3}{4} + \frac{L}{2}$. Therefore, it is sufficient to consider the situation depicted in Figure 5.3, and due to symmetry, we may assume $x \leq y$ w.l.o.g.

Algorithm 5.2: *Evacuate_L***Input:** Initial starting distance L

```

1 Robots orient so that they are “facing” the shortest arc (of length  $L$ ) between them
2 repeat
3   | WALK
4 until robots meet
5 Robots orient so that they are “facing” away from each other
6 repeat
7   | SEARCH
8 until exit is found
9 if this robot found the exit then
10  | EXIT
11 else
12   | Let  $x$  be the distance searched before the exit was found
13   | if  $3x \leq 1 - x$  then
14     | Reverse direction
15   | end
16   | repeat
17     | WALK
18   | until exit is reached
19   | EXIT
20 end

```

There are four important points to consider in the segment searched by each robot after time $t = \frac{1}{4} + \frac{L}{2}$: The starting location of the robot, the position of the robot at time t , and the two endpoints of the search segment. The line segments (and, to simplify notation, also their lengths) between these relevant points of the searched interval are labelled with the following convention: a_2 (respectively b_2) is the segment defined by the starting and current position of the robot, a_1 (resp. b_1) is the line segment defined by one of the endpoints and the boundary of a_2 , where additionally a_1 belongs to L , and finally, a_3 (resp. b_3) is the remaining segment between the other end point and the other boundary of a_2 , where a_3 does not belong to L . Note that none of a_1, a_2, a_3 (resp. b_1, b_2, b_3) intersect with the possible exception of their endpoints.

The values of x and y are defined dependant on whether the segments a_2 and b_2 , respectively, belong to the segment of length L connecting the starting locations of the robots.

FIGURE 5.3: Two robots starting at distance L explore the circle up to time $t = \frac{1}{4} + \frac{L}{2}$.

The dashed line represents the initial starting distance, L . The crosses represent the starting locations of the robots; the solid points represent the location of each robot at time t ; and the hollow points represent the endpoints of the segment searched by each robot at time t .

They have value 1 if the corresponding segment belongs to L , and value 0 otherwise. In this case, $x = 0$ (since a_2 does not belong to L) and $y = 1$ (since b_2 does). Note that a_1 and b_1 may overlap, however, as explained above, we may assume that at time t no robot is within the area explored by the other robot. Observe also that all the values are non-negative, but many of them can be zero.

From the fact that the robots travel at unit speed we get

$$2a_1 + a_2 + 2a_3 \leq \frac{1}{4} + \frac{L}{2} \quad (5.1)$$

$$2b_1 + b_2 + 2b_3 \leq \frac{1}{4} + \frac{L}{2} \quad (5.2)$$

The definition of x and y yields

$$yb_2 + b_1 + a_1 + xa_2 \geq L \quad (5.3)$$

Summing (5.1) with (5.2) and subtracting (5.3) produces

$$2b_3 + (1 - y)b_2 + b_1 + a_1 + (1 - x)a_2 + 2a_3 \leq \frac{1}{2} \quad (5.4)$$

At time t , if the points opposite to the robots are unsearched or are being searched (at time t), the adversary can place the exit at such an opposite point, forcing evacuation time of $\frac{3}{4} + \frac{L}{2}$.

Hence, it is sufficient to consider only situations in which

$$(1 - x)a_2 + a_1 + b_1 + b_2 + b_3 \geq \frac{1}{2} \quad (5.5)$$

and

$$(1 - y)b_2 + b_1 + a_1 + a_2 + a_3 \geq \frac{1}{2} \quad (5.6)$$

hold. Now, it is sufficient to consider two cases

- Case $x = 0$: Since all the values are non-negative, (5.4) contradicts (5.6)
- Case $x = y = 1$:

$$\begin{aligned} L + \frac{1}{2} &\geq (5.1) + (5.2) \\ &= 2b_3 + b_2 + 2b_1 + 2a_1 + a_2 + 2a_3 \\ &\geq b_3 + b_2 + 2b_1 + 2a_1 + a_2 + a_3 \\ &= (5.6) + (5.5) \\ &\geq 1 \end{aligned}$$

which is a contradiction since $L \leq \frac{1}{2}$.

■

FIGURE 5.4: k -exit evacuation of colocated robots.

5.3 Evacuation with k Exits

In this section, we study the more general problem with k exits. We begin by considering the problem when the k exits are all colocated, and make the following observation:

Observation 5.3. The evacuation problem with k exits and where the exits are colocated ($D = 1$), is equivalent to the evacuation problem with a single exit.

Therefore, all the results we have shown for a single exit are also valid for k colocated exits. We now consider the problem for k exits with an arbitrary D , but where the robots begin colocated.

Theorem 5.4. *The evacuation problem with k exits, where the robots begin colocated ($L = 0$), can be solved with an optimal worst case cost of $\frac{3}{4} \cdot D$, where D is the longest distance between exits.*

Proof. The solution for this more general problem is very similar to that for Lemma 5.1. The algorithm is almost identical: The robots begin searching in opposite directions until one of them finds the exit, at which point it will broadcast the location of the exit to the other robot. The other robot then calculates the direction which minimizes the distance which guarantees finding an exit and moves in that direction until it exits.

Algorithm 5.3: $\text{Evacuate}_0(D)$ **Input:** Longest distance between exits D

```

1 Robots orient so that they are “facing” away from each other
2 repeat
3   | SEARCH
4 until exit is found
5 if this robot found the exit then
6   | EXIT
7 else
8   | Let  $x$  be the distance searched before the exit was found
9   | if  $3x \leq D - x$  then
10  |   | Reverse direction
11  | end
12  | repeat
13  |   | WALK
14  | until exit is reached
15  | EXIT
16 end

```

We analyse this algorithm in a similar manner: suppose the robots each move distance x before one of them finds an exit (see Figure 5.4). The second robot can move $2x$ to evacuate through the same exit as the first robot, or it can continue moving in the same direction, where it will encounter an exit at time no more than $D - 2x$ (since D is the maximum distance between exits). The cost of the algorithm is determined by the time taken by the second robot to exit, and therefore $C = x + \min(2x, D - 2x)$, or $C = \min(3x, D - x)$. We note that $3x$ is monotone increasing and $D - x$ is monotone decreasing with respect to x , and therefore the choice of x which maximizes $\min(3x, D - x)$, and therefore the total cost, will be when $3x = D - x$, or when $x = D/4$. The worst case cost for our algorithm will therefore be $\frac{3}{4} \cdot D$.

We note that the input to the above algorithm is only the longest distance between exits, and at no time are the robots on any other arc than the one on which they start. Now suppose that the above algorithm is not optimal. Then there exists an algorithm $\mathcal{A}$ which has cost $C_{\mathcal{A}} < \frac{3}{4} \cdot D$ for some D . Now consider a circle of length D with a single exit.

By Lemma 5.1, the optimal worst case cost for this circle is $\frac{3}{4} \cdot D$. However, if we were to execute algorithm $\mathcal{A}$, we can achieve a better worst case cost for the circle. This is not possible, and so algorithm $\mathcal{A}$ cannot exist. ■

Finally, we conclude this section by considering the case when the exits are all evenly spaced.

Theorem 5.5. *The evacuation problem with k equally spaced exits ($D = \frac{1}{k}$), and where the robots begin at starting points at distance L , can be solved with an optimal worst case cost of $\frac{3}{4} \cdot \frac{1}{k} + \frac{\delta}{2}$, where*

$$\delta = \begin{cases} L \bmod \frac{1}{k} & : L \bmod \frac{1}{k} \leq \frac{1}{2k} \\ \frac{1}{k} - L \bmod \frac{1}{k} & : L \bmod \frac{1}{k} > \frac{1}{2k} \end{cases}$$

Proof. For the sake of analysis, we assume that the coordinates of the circle are assigned so that the point 0 corresponds with one of the exits. We first observe that since the exits are evenly spaced, when a robot searches a point $p \in [0, 1]$, it is simultaneously searching all points $(p + i/k) \bmod 1$, for $i = 1 \dots k$. We can visualize this as there being “virtual” robots at each of these points which behave exactly as the robot at point p . Consider now what happens if we take every point of the circle modulus $(1/k)$. We begin by noting that every exit is now located at $0 \bmod (1/k)$. A robot at point p and its corresponding virtual counterparts are all located at $p \bmod (1/k)$. Each arc of the circle is therefore an exact mirror of the others and we can collapse the problem to the problem of finding a single exit on a circle of length $1/k$. An example of this, where $k = 4$, is visualized in Figure 5.5. The only issue remaining to address is the initial distance of the robots, L . Since $L \bmod (1/k)$ can be larger than $1/2k$, we must adjust

FIGURE 5.5: Transforming a 4-exit evacuation with equally spaced exits to a single exit problem.

the initial distance in the single exit problem to be:

$$\delta = \begin{cases} L \bmod \frac{1}{k} & : L \bmod \frac{1}{k} \leq \frac{1}{2k} \\ \frac{1}{k} - L \bmod \frac{1}{k} & : L \bmod \frac{1}{k} > \frac{1}{2k} \end{cases}$$

The theorem follows immediately from Theorem 5.2. ■

5.4 k -exit Evacuation with Arbitrary D and L

In the previous section, we looked at cases where either D or L (or both) were fixed at a special value. We now examine the cases for arbitrary values of D and L . We begin by bounding the worst case cost above and below for the Evacuation Problem with Fixed Initial Distance. We then propose an algorithm which gives a solution for the Evacuation Problem with Chosen Initial Distance, for an arbitrary D when the algorithm has a free choice of L .

Lemma 5.6. *The evacuation problem with k exits, where the robots begin at distance L , can be solved with a worst case cost of $\frac{3}{4} \cdot D + \frac{L}{2}$ and cannot be solved with a worst case cost better than $\frac{3}{4} \cdot D - \frac{L}{2}$, where D is the longest distance between exits.*

Proof. Consider the algorithm where the robots walk towards each other until meeting, and then execute the algorithm from Theorem 5.4. It takes time $\frac{L}{2}$ until the robots are colocated, and at most $\frac{3}{4} \cdot D$ to execute the algorithm.

To show the lower bound, assume that there is an algorithm $\mathcal{A}$ which achieves a worst case cost better than $\frac{3}{4} \cdot D - \frac{L}{2}$. We can then construct an algorithm $\mathcal{A}'$ where two colocated robots first move away from each other until they are distance L apart. This will take time $L/2$. The robots then execute algorithm $\mathcal{A}$. Clearly algorithm $\mathcal{A}'$ has worst case time strictly better than $\frac{3}{4} \cdot D$, which is impossible, since $\frac{3}{4} \cdot D$ is the optimal worst case time for two colocated robots. ■

The upper bound is not surprising, since we know there are values of D and L for which this is optimal. What about the lower bound? To this point, every situation we have considered where the robots begin at distinct starting points has lead to higher costs than when the robots begin colocated. Is it possible that there are values of D and L for which we can find an algorithm which has a lower cost than the algorithm for the same D but when the robots are colocated? In fact, it is. We conclude this section by developing an algorithm $\text{Evacuate}_L(D)$ (stated in Theorem 5.8) which, for small values of L and large values of D , gives a worst case cost which is better than the optimal cost for the same D when the robots begin colocated.

Lemma 5.7 (Performance of $\text{Evacuate}_L(D)$). *Let $D \in [1/2, 1]$, which represents the longest distance between exits, be an input to the k -exit evacuation problem. Then, for*

FIGURE 5.6: k -exit evacuation for $\text{Evacuate}_L(D)$.

every $L \in (0, 1 - D]$, the performance of $\text{Evacuate}_L(D)$ is exactly

$$\max_{x \in [0, 1/2 - L/2]} g(x),$$

where

$$g(x) := \begin{cases} \min\{3x + L, D - L - x\} & , \text{ if } x \geq 1 - D \\ \min\{3x + L, 1 - L - x\} & , \text{ if } x < 1 - D \end{cases}$$

Proof. Since $L \in (0, 1 - D]$, we know that not all the exits can lie in the interval between the starting positions of the robots, and so the robots will necessarily encounter at least one exit before they meet. Denote the time when the first exit is found as x . Moreover, after time $1/2 - L/2$, and given that no robot has located an exit until then, the two robots meet on the ring. Hence, if x denotes the time it takes until one robot locates the exit, then $x \in [0, 1/2 - L/2]$.

In order to evaluate the performance of $\text{Evacuate}_L(D)$ we need to examine two cases (shown in Figure 5.6).

Algorithm 5.4: $\text{Evacuate}_L(D)$ **Input:** Initial starting distance L , longest distance between exits D

```

1 if  $L \geq 1 - D$  then
2   Robots orient so that they are “facing” the shortest arc (of length  $L$ ) between them
3   repeat
4     WALK
5   until distance between robots is  $1 - D$ 
6 end
7 Robots orient so that they are “facing” the longest arc between them
8 repeat
9   SEARCH
10 until exit is found
11 if this robot found the exit then
12   EXIT
13 else
14   Let  $x$  be the distance searched before the exit was found
15   if  $x \geq 1 - D$  then
16     if  $2x + L \leq D - L - 2x$  then
17       Reverse direction
18     end
19   else
20     if  $2x + L \leq 1 - L - 2x$  then
21       Reverse direction
22     end
23   end
24   repeat
25     WALK
26   until exit is reached
27   EXIT
28 end

```

In the first case (Figure 5.6a), the exit is located after time $x \geq 1 - D$, say by robot A . We know all exits are contained in an interval of size $1 - D$. Since both robots have already travelled distance $1 - D$, it is impossible for there to be any exits in the interval between their starting positions. Therefore, the second robot knows immediately where the two closest exits are: The exit through which the first robot evacuated, at distance $2x + L$ in the opposite direction; and an exit at distance $D - 2x - L$ in the same direction. Since it will choose the best route, the cost in this case is $x + \min\{2x + L, D - L - 2x\}$.

In the second case (Figure 5.6a), we assume that the exit is located after time $x < 1 - D$, say again by robot A . If robot B continues moving in the same direction, then since in the worst case the undiscovered exits could be located in the arc between the initial positions of the two robots, robot B will find the discovered exit in time $1 - 2x - L$. If robot B chooses to reverse back, then the undiscovered exits might not be in the arc between the initial positions of the robots, in which case robot B will find again the discovered exit in additional time $2x + L$. To conclude, if $x < D$, the cost of the algorithm in the worst case is $x + \min\{2x + L, 1 - L - 2x\}$. ■

Given that D will be given as input to the new evacuation problem, an algorithm may adapt its evacuation strategy accordingly. This is what we propose and explore next.

Theorem 5.8. *Let $D \in [1/k, 1]$, the longest distance between exits, be an input to the k -exit evacuation problem where L is a free choice. Consider $\text{Evacuate}(D)$ that simulates $\text{Evacuate}_L(D)$, where $L = L(D)$ is chosen according to the following rule*

$$L = \begin{cases} 1 - D & , \text{ if } 6/7 < D \leq 1 \\ 5D/2 - 2 & , \text{ if } 4/5 \leq D \leq 6/7 \\ 0 & , \text{ if } 1/k \leq D < 4/5 \end{cases}$$

Then the cost $C(D)$ of $\text{Evacuate}(D)$ is

$$C(D) = \begin{cases} 5D/4 - 1/2 & , \text{ if } 6/7 < D \leq 1 \\ 1 - D/2 & , \text{ if } 4/5 \leq D \leq 6/7 \\ 3D/4 & , \text{ if } 1/k \leq D < 4/5 \end{cases}$$

Algorithm 5.5: $\text{Evacuate}(D)$

Input: longest distance between exits D

```

1 if  $D > 6/7$  then
2   |  $L \leftarrow 1 - D$ 
3 else if  $D \geq 4/5$  then
4   |  $L \leftarrow 5D/2 - 2$ 
5 else
6   |  $L \leftarrow 0$ 
7 end
8 Robots are positioned on the circle at distance  $L$  apart
9 Robots orient so that they are “facing” the longest arc (of length  $1 - L$ ) between them
10 repeat
11   | SEARCH
12 until exit is found
13 if this robot found the exit then
14   | EXIT
15 else
16   | Let  $x$  be the distance searched before the exit was found
17   | if  $x \geq 1 - D$  then
18     | if  $2x + L \leq D - L - 2x$  then
19       | Reverse direction
20     | end
21   | else
22     | if  $2x + L \leq 1 - L - 2x$  then
23       | Reverse direction
24     | end
25   | end
26   | repeat
27     | WALK
28   | until exit is reached
29   | EXIT
30 end

```

FIGURE 5.7: Performance of $\text{Evacuate}(D)$ for $k = 2$ exits.

The value of D is along the x-axis, and the algorithm cost is along the y-axis.

Proof. Note that $L = L(D)$ always satisfies the condition that $L < 1 - D$, hence Lemma 5.7 is applicable for all values of $D \in [1/2, 1]$. Then, the cost $C(D)$ of $\text{Evacuate}(D)$ can be evaluated by an exhaustive case analysis that depends on the relative value of D with respect to the critical values $1, 6/7, 4/5, 1/k$, and it is depicted in Figure 5.7 for $k = 2$ exits.

We begin by observing that when $L = 0$, the cost of $\text{Evacuate}(D)$ is $3D/4$ by Theorem 5.4. We now evaluate the remaining cases by considering the point in time x after which the first exit has been found.

Case: $x \geq 1 - D$

From Lemma 5.7, we know that the time to evacuate for robot B is $\min(3x + L, D - L - x)$. We note that $3x + L$ is monotone increasing and $D - L - x$ is monotone decreasing with respect to x , and therefore the choice of x which maximizes $\min(3x + L, D - L - x)$,

and therefore the total cost, will be when $3x + L = D - L - x$, or when $x = D/4 - L/2$.

In this case, the cost will therefore be $\frac{3}{4} \cdot D - \frac{L}{2}$.

Case: $x < 1 - D$

From Lemma 5.7, we know that the time to evacuate for robot B is $\min(3x + L, 1 - L - x)$. We will focus solely on the case where the cost is $3x + L$ (even if this is more expensive than $1 - L - x$). We want to choose L to minimize

$$\max \left(3x + L, \frac{3}{4} \cdot D - \frac{L}{2} \right).$$

Since $3x + L$ is increasing, and $3D/4 - L/2$ is decreasing w.r.t. L , this is minimized when

$$3x + L = \frac{3}{4} \cdot D - \frac{L}{2}.$$

Since, $x \leq 1 - D$, this holds if

$$3 - 3D + L \leq \frac{3}{4} \cdot D - \frac{L}{2}.$$

Rearranging, we see that

$$L \leq \frac{5}{2} \cdot D - 2,$$

and since $L \geq 0$, this only holds if

$$\frac{4}{5} \leq D.$$

Furthermore, we choose $L = 5D/2 - 2$ to minimize the cost of the algorithm. Finally we note that $L < 1 - D$, and so when

$$1 - D \leq \frac{5}{2} \cdot D - 2,$$

we must set L to a maximum value of $L = 1 - D$. Rearranging the above, we see that this happens when $D > 6/7$. Putting these L values into $3D/4 - L/2$ give us the costs listed in the Theorem. ■

We conclude this section with a conjecture, which if true would imply that $\text{Evacuate}(D)$ is optimal for $D \geq 6/7$.

Conjecture 5.9. *For $D \geq 1/2$, any choice of L such that $L > 1 - D$ is suboptimal for the Evacuation Problem with Chosen Initial Distance.*

Our reasoning for the conjecture is as follows: We begin by noting that *all* exits are located on an arc of length $1 - D$. When $L > 1 - D$ and $D \geq 1/2$, both the long and the short arcs connecting the robots could contain all the exits. All points of these arcs begin unexplored. The robots then execute some algorithm $\mathcal{A}$ which searches the points of the arcs, and reduces the length of the corresponding unexplored arcs. In the worst case, there will be a point in time t at which one of the unexplored arcs is of length $1 - D$. At this point in time, the robots know that at least one exit must be located somewhere on the other arc, which will likely take an additional cost of $\text{Evacuate}_{(1-D)}$ to find. Furthermore, there must be points in time $t_A, t_B > 0$ at which the robots are located at the end points of this unexplored arc. Assume w.l.o.g. that $t_A < t_B$, then the cost of $\mathcal{A}$ should be at least $t_A + \text{Evacuate}_{(1-D)}$. The conjecture comes from being able to obtain a cost of $\text{Evacuate}_{(1-D)}$ by starting the robots at distance $1 - D$.

5.5 Conclusion

In this chapter, we studied the evacuation problem on a circle for 2 robots and k exits, and showed many optimal results. It is not immediately intuitive that the worst case solution costs are independent of the number of exits, but are dependent on the longest distance between exits.

Chapter 6

Conclusion and Future Work

We conclude this thesis by discussing the results obtained, summarizing the open problems stated in the thesis, and suggesting future research that could build upon these results. For a full summary of our results, we refer the reader to the Introduction or to the concluding section of each chapter.

6.1 Discussion

Some patterns emerged while studying the various problems, and some of the results were unexpected and worth highlighting and discussing. We will address each in turn in this section.

Asymmetrical Speeds

The core of the thesis was investigating how the two-speed model of robots might affect search problems. We investigated the two-speed model in the context of search and

patrolling and obtained some interesting results. We also investigated the rendezvous problem with a one-speed model of robots, but where each robot has different speeds.

It was no surprise that giving the robots the ability to move on the domain at a higher speed than searching allowed for more efficient algorithms, since more robots are able to contribute. What was perhaps a bit unexpected was the result for the Beachcombers' Problem which showed that no matter how the speeds of the robots were distributed, every single robot is able to make some contribution to the search, even if that contribution is minimal. This is markedly different from the situation with one-speed robots, where only the fastest robot is involved in the searching process, and adding more robots will not improve the solution (unless one of the added robots happens to be a faster searcher). Upon investigation of this problem, this does make intuitive sense; Even the slowest robot can search at the beginning of an interval which allows all the other robots to walk over this segment faster than if one of them had to search.

The online case is somewhat different. Now, not all robots are necessarily able to contribute in the two-speed case, but it is still possible for multiple robots to be able to increase the overall search speed of the solution – so long as they have a fast enough walking speed. This again compares well with the one-speed case where, again, only the fastest searcher will affect the solution.

For the patrolling problem, the two-speed robot model has less impact. For both the offline and online search problems the two-speed model fundamentally changes the nature of the solution by allowing more than one robot to participate in the searching. This type of impact is not present with the patrolling problem. The optimal solution we show involves a partition strategy, where the interval to be patrolled is divided into sub-intervals which each of the robots patrols independently. This strategy is also completely viable when considering a one-speed model of robots. However, having a higher walking speed does decrease the time it takes for a robot to patrol a given segment. This

is because, in the one-speed model, the points with the greatest idle time are the end-points of the patrolled interval, while all the interior points are visited more frequently. This is inefficient, but unavoidable in the one-speed model. In contrast, when applying the two-speed model, every point has an equal idle time.

Finally, the different speeds are important to the rendezvous problem. While the difference itself is not used to break symmetry, it is important in obtaining our results. If both robots move with the same speed, then it is entirely possible for rendezvous to never occur: if the robots end up moving in the same direction. With a single random bit, this is expected to occur half the time. However, with different speeds, the robots will always rendezvous – although this could take a long time if their speeds are extremely close. It is particularly interesting to note how the difference in speeds can affect the optimal strategy. When the speeds are close (the faster robot is no more than twice as fast as the slow robot), the best strategy is to walk until either rendezvous happens or it is determined that the robots must be moving in the same direction and then randomly choose a new direction. The more rounds like this (which depends on the number of random bits available), the better the expected result. Conversely, if the speeds are different enough (the faster robot is more than twice as fast as the slow robot), then the best strategy is to simply pick an initial direction at random and move until rendezvous.

Difficulty of partitioning

Perhaps the most interesting asymmetry of the two-speed robots is in the impact it has on the difficulty of partitioning groups for the Multi-Source Beachcombers' Problem. It is not at all surprising that partitioning robots into two or more groups with the goal of maximizing the combined search power of those groups can be NP-Hard. However, whereas some (and perhaps all) instances with a W-uniform (all walking speeds equal)

swarm are NP-hard, all instances with S-uniform (all searching speeds equal) swarms are solvable in polynomial time. There seems to be some underlying relationship between searching and walking speeds which determines the difficulty of the problem. It would be interesting to explore this further, to see if this relationship can be quantified, and ideally identify the threshold for which the problem tips from easy to hard.

Knowledge

We should not be at all surprised when more information available to our algorithms leads to more efficient results. This is perhaps most evident in the difference between the offline (Beachcombers' Problem) and Online Beachcombers' Problem, where there are some instances where knowledge of the length of the fence to be searched can cut the finishing time in half. We do also examine how knowledge affects performance with regards to the rendezvous problem, and most of the time, additional knowledge does lead to better performance.

What is surprising, however, is the situations in this thesis where knowledge has little or no impact on the resulting solution. Nowhere is this more evident than when comparing the centralized patrolling solution – where the algorithm has full knowledge of all the robots' speeds, as well as the length of the interval to be patrolled – to the decentralized one. It is very interesting to note that the decentralized algorithm actually converges to the same solution as the centralized algorithm. What's more, the robots needed to execute the decentralized algorithm need fewer capabilities, since they only need to recognize when a collision has occurred, and switch states accordingly.

Another example where knowledge has no impact on the efficiency of the solution is the rendezvous algorithm where the robots simply pick a direction at random and move

until rendezvous. This requires absolutely no knowledge and, as already mentioned, this is the optimal algorithm whenever the robots' speeds are sufficiently different.

Finally, we note that `ObliviousRandomized`, which randomly partitions the robots into t groups gives a good approximation for the Multi-source Beachcombers' Problem, and therefore any additional knowledge is likely not to provide a significant improvement in efficiency.

Simple Algorithms

A final observation we noticed across the various problems is the simplicity of the optimal algorithms. This is not to say that the algorithms are trivial – the analysis required to prove their optimality was technically involved – rather, the algorithms are elegant in their simplicity.

The patrolling algorithm involves partitioning the interval to be patrolled into subintervals, and having each robot independently patrol one sub-interval. The optimal rendezvous algorithms are either just picking a random direction and moving until rendezvous, or repeatedly moving for a fixed distance and picking a new direction at random until rendezvous is achieved. The evacuation problem is always solved by having the robots move away from each other until an exit is found; the remaining robot then takes the shortest route to a known exit. The most complicated algorithms are the searching algorithms `Comb` and `LeapFrog`, and both of these are still relatively simple.

6.2 Open Problems

Specific open problems stated in the thesis are listed here:

- Prove or disprove Conjecture 2.16 that there is no online two-speed algorithm with the competitive ratio of $(2 - \epsilon)$ for any $\epsilon > 0$.
- Prove or disprove Conjecture 2.22 that the two-speed search problem on a closed fence is equivalent to the two-speed search problem on an open fence.
- Find an optimal distributed algorithm for more than three robots under the same assumptions of starting positions and speeds already studied in this thesis.
- Prove or disprove Conjecture 5.9 that a starting distance greater than the longest distance between exits is suboptimal for the evacuation problem.

6.3 Future Work

We will now suggest future directions for research based from the results in this thesis. Since many of these directions are common to multiple search problems, we will organize this section around the proposed directions.

Apply Two-Speed Robot Model

As the rendezvous problem had not previously been studied extensively with a single speed per robot, we limited ourselves to this case. Similarly, we limited ourselves to the assumption of identical robots for the evacuation problem. Now that we have the

results in this thesis, a natural extension would be to apply the two-speed robot model to the study of these problems.

For rendezvous, it would need to be decided under what conditions a rendezvous occurs. Do both robots need to be in a search mode, or just one? We suspect that the only interesting case is when only one robot needs to be in a search mode, but this requires investigation to confirm.

Before evacuation is studied with two-speed robots, it should likely be studied under the conditions where the robots have different speeds. It may be the case that simply having different speeds does not change the problem significantly. In fact, since the evacuation problem seems to consist of a search phase to find an exit followed by a movement phase to evacuate, it may be that the evacuation algorithm proposed is in fact optimal with arbitrary two-speed robots.

Generalized Problems

In order to gain insight and facilitate analysis of some of the problems, we sometimes studied the problems under specialized conditions. An important extension to these problems would then be to study the corresponding problems under more general assumptions.

For the patrolling problem, we limited the robots to search only in a single direction. An important question to determine would be whether or not the partitioning strategy is optimal when the robots can patrol in both directions. We suspect that it may not be, since the partition strategy is not optimal with a group of robots with single, but potentially different speeds. However, it would be just as interesting to see if these more efficient strategies can make use of the two-speed model as well.

Keeping the patrolling assumptions we make in this thesis in place, it would also be nice to relax the restrictions we place on the decentralized algorithms and determine under which conditions the problem for two robots with arbitrary starting positions, movement modes and movement directions stabilizes to an equilibrium. This will require determining what the appropriate behaviour the robots should adopt during collisions from behind.

We also restricted the study of the rendezvous and evacuation problems to the case with two robots. A natural extension would be to expand the one-speed results to a more general gathering problem on a closed fence with multiple robots. Some results are known for the evacuation problem of multiple robots with one exit on a disc (instead of a circle) given by [Czyzowicz et al. \[2014a\]](#), and even in this case optimal results do not yet exist.

Topologies

Building off the results in this thesis to study these problems in more complex domains is a particularly important direction of research. Specifically, investigating the problems on trees, meshes, and other graph topologies seems like a natural direction to expand upon these results. This would likely involve determining the best way to divide the robots into two or more groups, which we have partially addressed in for searching, but seems like it would be more difficult for the other problems.

Examining the search and rendezvous problems on an infinite line with two-speed robots would also be an interesting direction of study, as search and rendezvous on infinite lines are well studied problems.

Finally, it would be interesting to examine all these problems in higher dimension metric spaces (i.e. $\mathbb{R}^2$, $\mathbb{R}^3$).

Distributed Search Algorithms

The search algorithms presented in this thesis are centralized, where full knowledge of the robots' capabilities are known. An important problem would be to apply the two-speed model of mobile robots to the study of distributed search algorithms.

Other Communication Models

Adding or removing communication capabilities is another direction of research. For example, would adding communication capabilities allow for more efficient online search algorithms? Or what about allowing robots to leave one or more pebbles on the circle during rendezvous? The wireless communication capabilities of the robots is a fairly strong assumption in the evacuation problem. What would the problem look like if only face-to-face communication were allowed instead?

Fault Tolerance

Finally, it would also be interesting to study a variant of the problems where some of the robots could be controlled by an adversary and behave maliciously. This could be through lying about finding or not finding a search target, an exit, an intruder, etc. To overcome this, the algorithms would need to be made to be fault tolerant. For example in the searching and patrolling algorithms, each point of the segment might need to be searched by 2, 3 or k different robots, instead of only one.

Abbreviations

CAP	Combinatorial Allocation P roblem
FPTAS	Fully P olynomial- T ime Approximation S cheme
LP	Linear P rogram(min)
1-SBP	1 Source B eachcombers' P roblem
2-SBP	2 Source B eachcombers' P roblem
<i>t</i>-SBP	<i>t</i> Source B eachcombers' P roblem
SPAO	Set P artitioning Problem with A dditive O bjective

Nomenclature

$\mathcal{I}_{\mathcal{K}}$	Idleness of schedule $\mathcal{K}$, page 83
I_L	Interval $[0, L]$ for any positive number L , page 26
n	Number of robots, page 26
p_i	Patrolling speed of robot r_i , page 82
R	Swarm of n robots, page 26
s_i	Searching speed of robot i , page 26
$S_{\mathcal{K}}(L)$	Speed with which $\mathcal{K}$ searches I_L , page 27
w_i	Walking speed of robot i , page 26
$\mathcal{K}$	A mobility schedule, page 27

Bibliography

- N. Agmon, S. Kraus, and G. A. Kaminka. Multi-robot perimeter patrol in adversarial settings. In *ICRA*, pages 2339–2345, 2008.
- S. Albers. Online algorithms: a survey. *Math. Program.*, 97(1-2):3–26, 2003.
- S. Albers and M. R. Henzinger. Exploring unknown environments. *SIAM J. Comput.*, 29(4):1164–1188, 2000.
- S. Albers and S. Schmelzer. Online algorithms - what is it worth to know the future? In *Algorithms Unplugged*, pages 361–366. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-15327-3. doi: 10.1007/978-3-642-15328-0_37. URL http://dx.doi.org/10.1007/978-3-642-15328-0_37.
- S. Albers, K. Kursawe, and S. Schuierer. Exploring unknown environments with obstacles. *Algorithmica*, 32(1):123–143, 2002.
- A. Allahverdi, C. T. Ng, T. C. E. Cheng, and M. Kovalyov. A survey of scheduling problems with setup times or costs. *Eur J Oper Res*, 187(3):985–1032, 2008.
- A. Almeida, G. Ramalho, H. Santana, P. A. Tedesco, T. Menezes, V. Corruble, and Y. Chevaleyre. Recent advances on multi-agent patrolling. In *SBIA*, pages 474–483, 2004.
- S. Alpern. The rendezvous search problem. *SIAM Journal on Control and Optimization*, 33(3):673–683, 1995.
- S. Alpern. Asymmetric rendezvous search on the circle. *Dynamics and Control*, 10(1):33–45, 2000. ISSN 0925-4668. doi: 10.1023/A:1008339811950. URL <http://dx.doi.org/10.1023/A%3A1008339811950>.

- S. Alpern and S. Gal. *The theory of search games and rendezvous*, volume 55. Kluwer Academic Publishers, 2002.
- S. Alpern, A. Morton, and K. Papadaki. Optimizing randomized patrols. *Operational Research Group, London School of Economics and Political Science*, 2009.
- S. Alpern, A. Morton, and K. Papadaki. Patrolling games. *Operations research*, 59(5): 1246–1257, 2011.
- F. Amigoni, N. Basilico, N. Gatti, A. Saporiti, and S. Troiani. Moving game theoretical patrolling strategies from theory to practice: An usarsim simulation. In *ICRA*, pages 426–431, 2010.
- S. Angelopoulos, A. López-Ortiz, and K. Panagiotou. Multi-target ray searching problems. *Algorithms and Data Structures*, pages 37–48, 2011.
- D. Angluin, J. Aspnes, Z. Diamadi, M. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18(4):235–253, 2006.
- D. Angluin, J. Aspnes, D. Eisenstat, and E. Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, 2007.
- S. Anily and A. Federgruen. Structured partitioning problems. *Operations Research*, 39(1):130–149, 1991.
- R. Baeza-Yates and R. Schott. Parallel searching in the plane. *Computational Geometry*, 5(3):143–154, 1995.
- R. Baeza Yates, J. Culberson, and G. Rawlins. Searching in the plane. *Information and Computation*, 106(2):234–252, 1993.
- R. A. Baeza-Yates, J. C. Culberson, and G. J. E. Rawlins. Searching in the plane. *Information and Computation*, 106:234–234, 1993.
- E. Bampas, J. Czyzowicz, L. Gąsieniec, D. Ilcinkas, and A. Labourel. Almost optimal asynchronous rendezvous in infinite multidimensional grids. In N. Lynch and A. Shvartsman, editors, *Distributed Computing*, volume 6343 of *Lecture Notes in*

- Computer Science*, pages 297–311. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-15762-2. doi: 10.1007/978-3-642-15763-9_28. URL http://dx.doi.org/10.1007/978-3-642-15763-9_28.
- N. Baumann and M. Skutella. Earliest arrival flows with multiple sources. *Mathematics of Operations Research*, 34(2):499–512, 2009.
- J. Beauquier, J. Burman, J. Clement, and S. Kutten. On utilizing speed in networks of mobile agents. In *Proceeding of the 29th ACM SIGACT-SIGOPS Symposium on Principles of distributed computing*, pages 305–314. ACM, 2010.
- A. Beck. on the linear search problem. *Israel Journal of Mathematics*, 2(4):221–228, 1964.
- R. Bellman. An optimal search problem. *Bull. Am. Math. Soc.*, page 270, 1963a.
- R. Bellman. An optimal search. *Siam Review*, 5(3):274–274, 1963b.
- M. A. Bender, A. Fernández, D. Ron, A. Sahai, and S. Vadhan. The power of a pebble: Exploring and mapping directed graphs. *Inf. Comput.*, 176(1):1–21, July 2002. ISSN 0890-5401. doi: 10.1006/inco.2001.3081. URL <http://dx.doi.org/10.1006/inco.2001.3081>.
- P. Berman. On-line searching and navigation. In A. Fiat and G. Woeginger, editors, *Online Algorithms The State of the Art*, pages 232–241. Springer, 1998.
- M. Blum and D. Kozen. On the power of the compass (or, why mazes are easier to search than graphs). In *Foundations of Computer Science, 1978., 19th Annual Symposium on*, pages 132–142, 1978. doi: 10.1109/SFCS.1978.30.
- A. Bonato and R. Nowakowski. *The game of cops and robbers on graphs*. American Mathematical Soc., 2011.
- P. Brucker, A. Gladky, J. A. Hoogeveen, M. Kovalyov, C. Potts, T. Tautenhahn, and S. Velde. Scheduling a batching machine. *Journal of Scheduling*, 1(1):31–54, 1998.
- A. K. Chakravarty, J. B. Orlin, and U. G. Rothblum. Consecutive optimizers for a partitioning problem with applications to optimal inventory groupings for joint replenishment. *Operations Research*, 33(4):820–834, 1985a.

- A. K. Chakravarty, J. B. Orlin, and V. G. Rothblum. A partitioning problem with additive objective with an application to optimal inventory groupings for joint replenishment. *Operations. Research*, 30:1018–1020, 1985b.
- J. Chalopin, S. Das, and N. Santoro. Rendezvous of mobile agents in unknown graphs with faulty links. In *Distributed Computing*, pages 108–122. Springer, 2007.
- J. Chalopin, P. Flocchini, B. Mans, and N. Santoro. Network exploration by silent and oblivious robots. In *WG*, pages 208–219, 2010.
- Y. Chevaleyre. Theoretical analysis of the multi-agent patrolling problem. In *IAT*, pages 302–308, 2004.
- M. Chrobak, L. Gąsieniec, T. Gorry, and R. Martin. Group search on the line. In *SOFSEM 2015: Theory and Practice of Computer Science*, pages 164–176. Springer, 2015.
- T. H. Chung, G. A. Hollinger, and V. Isler. Search and pursuit-evasion in mobile robotics. *Autonomous Robots*, 31(4):299–316, 2011.
- M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro. Distributed computing by mobile robots: Gathering. *SIAM J. Comput.*, 41(4):829–879, 2012.
- A. Collins, J. Czyzowicz, L. Gasieniec, A. Kosowski, E. Kranakis, D. Krizanc, R. Martin, and O. Morales Ponce. Optimal patrolling of fragmented boundaries. In *SPAA*, 2013.
- J. Czyzowicz, S. Dobrev, E. Kranakis, and D. Krizanc. The power of tokens: rendezvous and symmetry detection for two mobile agents in a ring. In *SOFSEM 2008: Theory and Practice of Computer Science*, pages 234–246. Springer, 2008.
- J. Czyzowicz, L. Gasieniec, A. Kosowski, and E. Kranakis. Boundary patrolling by mobile agents with distinct maximal speeds. In *ESA*, pages 701–712, 2011.
- J. Czyzowicz, D. Ilcinkas, A. Labourel, and A. Pelc. Worst-case optimal exploration of terrains with obstacles. *Inf. Comput.*, 225:16–28, 2013a.
- J. Czyzowicz, A. Kosowski, and A. Pelc. Deterministic rendezvous of asynchronous bounded-memory agents in polygonal terrains. *Theory of Computing Systems*, 52

- (2):179–199, 2013b. ISSN 1432-4350. doi: 10.1007/s00224-011-9379-7. URL <http://dx.doi.org/10.1007/s00224-011-9379-7>.
- J. Czyzowicz, E. Kranakis, and E. Pacheco. Localization for a system of colliding robots. In *ICALP* (2), pages 508–519, 2013c.
- J. Czyzowicz, L. Gąsieniec, T. Gorry, E. Kranakis, R. Martin, and D. Pajak. Evacuating robots via unknown exit in a disk. In *Distributed Computing*, pages 122–136. Springer, 2014a.
- J. Czyzowicz, L. Gąsieniec, K. Georgiou, E. Kranakis, and F. MacQuarrie. The beachcombers’ problem: Walking and searching with mobile robots. In *Structural Information and Communication Complexity*, volume 8576 of *Lecture Notes in Computer Science*, pages 23–36. Springer International Publishing, 2014b. ISBN 978-3-319-09619-3. doi: 10.1007/978-3-319-09620-9_4. URL http://dx.doi.org/10.1007/978-3-319-09620-9_4.
- J. Czyzowicz, L. Gąsieniec, K. Georgiou, E. Kranakis, and F. MacQuarrie. The multi-source beachcombers’ problem. In *Proceedings of the 10th International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics*, ALGOSENSORS 2014, pages 3–21. Springer Berlin Heidelberg, 2015a.
- J. Czyzowicz, K. Georgiou, E. Kranakis, L. Narayanan, J. Opatrny, and B. Vogtenhuber. Evacuating robots from a disk using face-to-face communication. In *CIAC*, volume to appear, 2015b.
- S. Das, P. Flocchini, S. Kutten, A. Nayak, and N. Santoro. Map construction of unknown graphs by multiple agents. *Theor. Comput. Sci.*, 385(1-3):34–48, 2007.
- G. De Marco, L. Gargano, E. Kranakis, D. Krizanc, A. Pelc, and U. Vaccaro. Asynchronous deterministic rendezvous in graphs. *Theoretical Computer Science*, 355(3):315–326, 2006.
- E. D. Demaine, S. P. Fekete, and S. Gal. Online searching with turn cost. *Theoretical Computer Science*, 361(2):342–355, 2006.

- X. Deng and C. H. Papadimitriou. Exploring an unknown graph. In *Foundations of Computer Science, 1990. Proceedings., 31st Annual Symposium on*, pages 355–361. IEEE, 1990.
- X. Deng, T. Kameda, and C. H. Papadimitriou. How to learn an unknown environment (extended abstract). In *FOCS*, pages 298–303, 1991.
- D. Dereniowski, Y. Disser, A. Kosowski, D. Pajak, and P. Uznanski. Fast collaborative graph exploration. In *ICALP*, volume 7966, pages 520–532, 2013.
- A. Dessmark, P. Fraigniaud, and A. Pelc. Deterministic rendezvous in graphs. In *Algorithms-ESA 2003*, pages 184–195. Springer, 2003.
- E. W. Dijkstra. *Selected writings on computing: a personal perspective*. Springer-Verlag New York, Inc., 1982.
- S. Dobzinski and M. Schapira. An improved approximation algorithm for combinatorial auctions with submodular bidders. In *SODA*, pages 1064–1073. ACM Press, 2006. ISBN 0-89871-605-5.
- A. Dumitrescu, A. Ghosh, and D. T. Csaba. On fence patrolling by mobile agents. *CoRR*, abs/1401.6070, 2014.
- Y. Elmaliach, A. Shiloni, and G. A. Kaminka. A realistic model of frequency-based multi-robot polyline patrolling. In *AAMAS (I)*, pages 63–70, 2008.
- Y. Elmaliach, N. Agmon, and G. A. Kaminka. Multi-robot area patrol under frequency constraints. *Ann. Math. Artif. Intell.*, 57(3-4):293–320, 2009.
- Y. Elor and A. M. Bruckstein. Autonomous multi-agent cycle based patrolling. In *ANTS*, pages 119–130, 2010.
- U. Feige and J. Vondrák. The submodular welfare problem with demand queries. *Theory of Computing*, 6(1):247–290, 2010. URL <http://dx.doi.org/10.4086/toc.2010.v006a011>.
- O. Feinerman, A. Korman, Z. Lotker, and J. S. Sereni. Collaborative search on the plane without communication. In *Proceedings of the 2012 ACM symposium on Principles of distributed computing*, pages 77–86. ACM, 2012.

- O. Feinerman, A. Korman, S. Kutten, and Y. Rodeh. Fast rendezvous on a cycle by agents with different speeds. *Proceedings of the 15th International Conference on Distributed Computing and Networking (ICDCN 2014)*, 2014.
- S. Fekete, C. Gray, and A. Kröller. Evacuation of rectilinear polygons. In *Combinatorial Optimization and Applications*, pages 21–30. Springer, 2010.
- A. Fiat, D. P. Foster, H. Karloff, Y. Rabani, Y. Ravid, and S. Viswanathan. Competitive algorithms for layered graph traversal. In *Foundations of Computer Science, 1991. Proceedings., 32nd Annual Symposium on*, pages 288–297. IEEE, 1991.
- R. Fleischer, T. Kamphans, R. Klein, E. Langetepe, and G. Trippen. Competitive online approximation of the optimal search ratio. *SIAM J. Comput.*, 38(3):881–898, 2008.
- P. Flocchini, E. Kranakis, D. Krizanc, N. Santoro, and C. Sawchuk. Multiple mobile agent rendezvous in a ring. In *LATIN 2004: Theoretical Informatics*, pages 599–608. Springer, 2004.
- P. Flocchini, G. Prencipe, and N. Santoro. Distributed computing by oblivious mobile robots. *Synthesis Lectures on Distributed Computing Theory*, 3(2):1–185, 2012. doi: 10.2200/S00440ED1V01Y201208DCT010. URL <http://www.morganclaypool.com/doi/abs/10.2200/S00440ED1V01Y201208DCT010>.
- F. V. Fomin and D. M. Thilikos. An annotated bibliography on guaranteed graph searching. *Theor. Comput. Sci.*, 399(3):236–245, 2008.
- P. Fraigniaud, L. Gasieniec, D. R. Kowalski, and A. Pelc. Collective tree exploration. *Networks*, 48(3):166–177, 2006.
- B. Gluss. An alternative solution to the “lost at sea” problem. *Naval Research Logistics Quarterly*, 8(1):117–122, 1961.
- N. Hazon and G. A. Kaminka. On redundancy, efficiency, and robustness in coverage for multiple robots. *Robotics and Autonomous Systems*, 56(12):1102–1114, 2008.
- Y. Higashikawa, N. Katoh, S. Langerman, and S. Tanigawa. Online graph exploration algorithms for cycles and trees by multiple searchers. *J. Comb. Optim.*, 2012.

- F. Hoffmann, C. Icking, R. Klein, and K. Kriegel. The polygon exploration problem. *SIAM Journal on Computing*, 31(2):577–600, 2001.
- J. V. Howard. Rendezvous search on the interval and the circle. *Operations Research*, 47(4):550–558, 1999. doi: 10.1287/opre.47.4.550. URL <http://pubsonline.informs.org/doi/abs/10.1287/opre.47.4.550>.
- M. Y. Kao and M. L. Littman. Algorithms for informed cows. In *AAAI-97 Workshop on On-Line Search*, 1997.
- M. Y. Kao, J. H. Reif, and S. R. Tate. Searching in an unknown environment: An optimal randomized algorithm for the cow-path problem. In *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, pages 441–447. Society for Industrial and Applied Mathematics, 1993.
- A. Kawamura and Y. Kobayashi. Fence patrolling by mobile agents with distinct speeds. In *ISAAC*, pages 598–608, 2012.
- J. M. Kleinberg. On-line search in a simple polygon. In *Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 8–15. Society for Industrial and Applied Mathematics, 1994.
- B. O. Koopman. *Search and screening*. Operations Evaluation Group, Office of the Chief of Naval Operations, Navy Department, 1946.
- E. Kranakis, D. Krizanc, N. Santoro, and C. Sawchuk. Mobile agent rendezvous in a ring. In *Distributed Computing Systems, 2003. Proceedings. 23rd International Conference on*, pages 592–599. IEEE, 2003.
- E. Kranakis, D. Krizanc, and S. Rajsbaum. Mobile agent rendezvous: A survey. In *Structural Information and Communication Complexity*, pages 1–9. Springer, 2006.
- E. Kranakis, D. Krizanc, and P. Morin. Randomized Rendez-Vous with Limited Memory. In *Proceedings of 8th Latin American Theoretical Informatics Symposium*, LNCS 4957, pages 605–614, 2008.
- E. Kranakis, D. Krizanc, and E. Markou. The mobile agent rendezvous problem in the ring. *Synthesis Lectures on Distributed Computing Theory*, 1(1):1–122, 2010.

- A. Machado, G. Ramalho, J.-D. Zucker, and A. Drogoul. Multi-agent patrolling: An empirical analysis of alternative architectures. In *MABS*, pages 155–170, 2002.
- M. Marden. *The Geometry of the Zeros of a Polynomial in a Complex Variable*, volume 3 of *Math. Surv.* AMS, 1949.
- A. Marino, L. E. Parker, G. Antonelli, and F. Caccavale. Behavioral control for multi-robot perimeter patrol: A finite state automata approach. In *ICRA*, pages 831–836, 2009.
- Y. Métivier, N. Saheb, and A. Zemhari. Analysis of a randomized rendezvous algorithm. *Information and Computation*, 184(1):109–128, 2003.
- P. Nahin. *Chases and Escapes: The Mathematics of Pursuit and Evasion*. Princeton University Press, 2012.
- C. T. Ng, M. S. Barketau, T. C. E. Cheng, and M. Y. Kovalyov. "product partition" and related problems of scheduling and systems reliability: Computational complexity and approximation. *European Journal of Operational Research*, 207(2):601–604, 2010.
- C. H. Papadimitriou and M. Yannakakis. Shortest paths without a map. *Theoretical Computer Science*, 84(1):127–150, 1991.
- F. Pasqualetti, A. Franchi, and F. Bullo. On optimal cooperative patrolling. In *CDC*, pages 7153–7158, 2010.
- C. Potts and M. Kovalyov. Scheduling with batching: a review. *European Journal of Operational Research*, 120(2):228–249, 2000.
- C. E. Shannon. Presentation of a maze-solving machine. In *8th Conf. of the Josiah Macy Jr. Found.(Cybernetics)*, pages 173–180, 1951.
- L. D. Stone. *Theory of optimal search*. Academic Press New York, 1975.
- R. Uzsoy. Scheduling a single batch processing machine with non-identical job sizes. *The International Journal of Production Research*, 32(7):1615–1635, 1994.
- G. Wang, M. J. Irwin, H. Fu, P. Berman, W. Zhang, and T. L. Porta. Optimizing sensor movement planning for energy efficiency. *ACM Transactions on Sensor Networks*, 7(4):33, 2011.

-
- V. Yanovski, I. A. Wagner, and A. M. Bruckstein. A distributed ant algorithm for efficiently patrolling a network. *Algorithmica*, 37(3):165–186, 2003.
- X. Yu and M. Yung. Agent rendezvous: A dynamic symmetry-breaking problem. In *Automata, Languages and Programming*, pages 610–621. Springer, 1996.

Index

Beachcombers' Problem

- multi-source, [29](#)
- offline, [29](#)
- online, [29](#)

Evacuation Problem, [145](#)

exit, [143](#)

fence

- closed, [5](#)
- open, [5](#)

Fence Patrolling Problem, [82](#)

idleness

- of a point, [82](#)
- of a schedule, [83](#)

patrolling, [82](#)

- direction, [82](#)
- range, [83](#)
- speed, [82](#)

Randomized Rendezvous Problem, [113](#)

regular system, [97](#)

robot, [2](#)

- contribution, [48](#)
- speed
 - patrolling, [82](#)
 - searching, [25](#)
 - walking, [25](#), [82](#)
- two-speed, [4](#)

schedule

- correct, [26](#)
- feasible, [27](#)
- idleness, [83](#)
- mobility, [27](#)
- searching, [26](#)
- speed, [29](#)

Scheduling with Regular Delivery, [84](#)

search power, [43](#)

searching, [25](#)

speed, [25](#)

Set Partition Problem with Additive Objective, [61](#)

source, [26](#)

swarm

- monotone, [105](#)
 - strongly monotone, [107](#)
- robot, [46](#)
- speed, [45](#)
- uniform
 - S-uniform, [26](#)
 - totally uniform, [52](#)
 - W-uniform, [26](#), [52](#)

time

finishing, [27](#)

walking, [25](#), [82](#)

speed, [25](#), [82](#)