Exploration Problems using Autonomous Robots: PhD Thesis

Université du Québec en Outaouais
Département d'Informatique et d'Ingénierie

Maxime Godon

April 2018

Abstract

We introduce and study search problems for collections of mobile robots. Throughout our work, we consider cases where robots may experience various type of faults, which may hinder the progress of the algorithm and require novel ways to solve the problem. We first discuss the search and exploration problem in the Euclidian plane, where a group of k robots with various maximal speeds, among which up to f may be faulty, have to discover a target and gather at its location, using various communication models. We then discuss the problem of search for a non-adversarial, uncooperative robot on the cycle, based on various premisses regarding the information available to the robots. We also consider the evacuation problem on the disc in the presence of faulty robots. In the evacuation problem, the robot finding a target point (exit) on the boundary of the environment must communicate it to other robots, which then all need to move to the exit (evacuate). Finally, we consider the problem of exploring various types of graphs in the presence of time constraints: each node has to be visited before a certain moment in time for the algorithm to succeed. We discuss the complexity of the exploration, based on the number of robots, the presence or absence of faults, and the topology of the graph.

For all problems, we introduce algorithms to solve the problem, and discuss their efficiency.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The area of search problems using mobile robots[1] is well-studied in the realm of theoretical computer science, from classical algorithms such as Dijkstra's algorithm to find the shortest path to the famous Traveling Salesperson Problem. Those have direct real-world applications, be it to find the fastest route for a car trip, as is implemented by software such as *Google Map*, or the best way for a postman to complete their itinerary. The case of exploration of geometrical environments is itself well-studied and has a plethora of real-world applications, ranging from patrolling of an area using unmanned aerial vehicles (UAV) to exploration of new environments, such as the exploration by the Mars Rover. It is but a natural extension to this setting to consider faulty agents: robots in charge of any given exploration may stop working, or provide unreliable data at any time. Under more adversarial conditions, a robot may be hacked by an adversary and start acting erratically or contrary to the intention of its original programming. As such, building for resilience is a necessary form of

[1] In this thesis, the words *robot* and *agent* are used indiscriminately.

precaution in many real-world areas. This is reflected in our theoretical world by discussing problems where agents may experience crash-faults and Byzantine faults.

1.2 Objective and Methodology

The purpose of this thesis will be to study various exploration problems using mobile agents. For these problems, we intend to find algorithms that solve the problem using optimal resources (time, number of steps, number of required agents, percentage of the agents that have to be reliable, etc.), and to establish their efficiency using lower and upper bound proofs.

An algorithm solves an exploration problem by producing a schedule that will be followed by the agents. This schedule has to ensure that the objective of the problem is met (be it complete exploration, rendezvous, evacuation, etc.), with respect to the constraints of the problem. Our goal is to achieve efficiency, which can be defined as minimal usage of time, of the agent's energy, or in certain cases, a competitive ratio, which is defined as the ratio between the total cost required by an on-line algorithm and the best-case cost to solve the problem offline. We focus on the correctness of produced schedules, and on the efficiency (and sometimes optimality) of said schedules. We also discuss the correctness and efficiency of the algorithms producing those schedules.

The methods used to offer these solutions include oracle and adversarial arguments, proofs of NP-Completeness by Karp reduction, proofs by induction and contradiction and proof by exhaustive consideration of all possible cases.

1.3 Models Studied

The work presented in this thesis revolves around a few main concepts: mobile agents, searching, tolerance to faults and communication.

All our work is related to the use of mobile agents. In this thesis, we use the terms "mobile agent" and "robot" indiscriminately. It refers to an entity modelled as a single point in the geometric environment. A mobile agent has four main capabilities: (1) movement: an agent may traverse the environment in which it is located; (2) perception: a mobile agent is equipped with sensors that allow it to register information located in its environment; (3) computation: an agent can make independent calculations. In this thesis, we assume that an agent has access to unlimited memory; (4) communication: depending on the model, an agent is either equipped with a device similar to a broadcasting station, that allows it to communicate wirelessly with every other agent, or a simple device that allows it to exchange information when at some time it is located at the same point of the environment as another agent.

The main objective of the algorithms presented here is to search the environment for a given target. In a part of our work, this target is mobile, and in other parts, it is motionless. The search may be conducted by multiple robots, and it may be required that every robot completes the search for the algorithm to accomplish its goal. Search differs from exploration in the sense that it does not always require for a single robot or a group of robots to visit every point of the environment (which may, in some cases, be infinite). A part of our work focuses on evacuation, which is sometimes referred to as group search.

Every part of this work includes robots that may be faulty, to a various degree. Three main type of faults are discussed in this paper: uncooperative, crash faults and byzantine faults. An uncooperative robot is completely outside the control of

the algorithm, and though it will not try to hamper the progress of the algorithm, it will not help it in any way either. Specifically, we use the concept of bus: a non-adversarial, uncooperative mobile agent that moves at its maximum speed along a predefined route. It does not deviate from its path, nor slow down for any reason. A robot experiencing crash fault may stop working at any moment, as chosen by an adversary. The robot may stop moving, communicating, searching or listening to other robots' communication. It does not, however, gain additional capabilities, such as the capability to transmit false information. A robot experiencing byzantine faults is similar to a robot experiencing crash faults, but also has the capability to transmit false information, deviate from its assigned path, stop exploring or stop communicating. When communicating, however, we consider that all information given by the robot is signed to its identity. It is therefore impossible for the robot to pretend to be another agent. A byzantine robot will act in an adversarial way, trying to hamper the progress of the algorithm to the best of its capacities. The objective of the algorithms presented in this thesis is to build a resilience to those faults. We use the term "reliable robot" to refer to a robot that does not experience faults.

Our algorithms consider two main communication models: the face-to-face (F2F) and the wireless communication model. In the face-to-face communication setting, agents may exchange any amount of information at no cost (instantaneous communication), but only when at the same moment in time they are located at the same point in the environment, whereas in the wireless communication setting, agents may exchange any amount of information at no cost (instantaneous communication), regardless of the respective location of the agents in the environment.

1.4 Thesis Results

The purpose of this thesis is to study various exploration problems using a group of k mobile robots. We are particularly interested in the scenario where some robots are unreliable, i.e. $f < k$ robots may experience faults.

In the first part of our work, a collection of k robots, initially placed at the origin of the plane, are searching for a stationary target. Each robot r_i has a unit visibility range and can move no faster than its maximal speed v_i, for $i = 1, 2, ..., k$. We consider two communication models: wireless, in which a message sent by a robot can reach all other robots immediately, regardless of their positions, and face-to-face, in which robots can only exchange information when they are meeting. We assume that up to $f < k$ robots may be unreliable. We consider two models of unreliability: (1) crash faults, in which we deal with an absence of some of the robots' capacities (communication, perception, motion, etc.), and (2) Byzantine faults, in which the robots may be malicious in that they may lie (e.g., transmitting wrong information).

The goal is to minimize the group search (also known as evacuation) time, which is equal to the time of arrival to the target of the last reliable robot. This is expressed as a function of d, the distance from the origin to the target.

Our proposed algorithms for crash faults are asymptotically optimal in d in both communication models. For byzantine faults, we propose an algorithm which is asymptotically optimal for the wireless model. In the F2F model, we propose two algorithms: the first one has a competitive ratio of 2, while the second algorithm works for $k \geq 2f + 2$ and is optimal when the robots' speeds are all equal.

Our results also extend to the traditional search model which measures the time of arrival to the target of the first reliable robot.

The second part of our work assumes that k robots are placed on the perimeter of

a unit (radius) disk at a position of our choosing and can move with maximum speed 1. A non-adversarial, uncooperative agent, called *bus* is moving with constant speed s along the perimeter of the cycle. The robots are searching for the moving bus but do not know its exact location; during the search the robots can move anywhere on the perimeter of the cycle and can communicate wirelessly. We give algorithms which minimize the worst-case search time required for at least one of the robots to find the bus.

We obtain the following results for one robot. 1) If the robot knows the speed s of the bus but does not know its direction of movement then the optimal search time is shown to be exactly 1a) $2\pi/s$ if $s \geq 1$, 1b) $4\pi/(s+1)$ if $1/3 \leq s \leq 1$, and 1c) $2\pi/(1-s)$ if $s \leq 1/3$. 2) If the robot does not know neither the speed nor the direction of movement of the bus then there is an algorithm with search time $2\pi(1+\frac{1}{s+1})$, and for any $\epsilon > 0$, it is possible to assign a speed s to the bus such that no algorithm can achieve rendezvous with the bus in time less than $4\pi - \frac{\epsilon}{2}$.

We also generalize these results to $k \geq 2$ robots and prove analogous tight upper and lower bounds depending on the knowledge the robots have about the speed and direction of movement of the bus, provided that the robots can communicate wirelessly.

The third part of our work considers the evacuation problem on a circle for three robots, at most one of which is faulty. The three robots search for an exit placed at an unknown location on the perimeter of the circle. During the search, robots can communicate wirelessly at any distance. The goal is to minimize the time it takes the last non-faulty robot to reach the exit.

Our main contributions are two highly efficient and non intuitive evacuation protocols for the non-faulty robots to reach the exit in two well-studied fault-models, Crash and Byzantine. Moreover, we complement our positive results by lower bounds

in both models. A summary of our results reads as follows:

- *Case of Crash Faults:* Lower Bound ≈ 5.082; Upper Bound ≈ 6.309,
- *Case of Byzantine Faults:* Lower Bound ≈ 5.948; Upper Bound ≈ 6.921,

For comparison, it is known (see [56]) that in the case of three *non-faulty* robots with wireless communication we have a lower bound of 4.159, and an upper bound of 4.219 for evacuation, while for two non-faulty robots $1+2\pi/3+\sqrt{3} \approx 4.779$ is a tight upper and lower bound for evacuation.

In the fourth part of this thesis, a graph environment must be explored by a collection of mobile robots. Some of the robots, a priori unknown, may turn out to be unreliable. The graph is weighted and each node is assigned a deadline. The exploration is successful if each node of the graph is visited before its deadline by a reliable robot. The edge weight corresponds to the time needed by a robot to traverse the edge. Given the number of robots which may crash, is it possible to design an algorithm, which will always guarantee the exploration, independently of the choice of the subset of unreliable robots by the adversary? We find the optimal time, during which the graph may be explored. Our approach permits to find the maximal number of robots, which may turn out to be unreliable, and the graph is still guaranteed to be explored.

We concentrate on line graphs and rings, for which we give positive results. We start with the case of collections involving only reliable robots. We give algorithms which find the optimal times needed for exploration when the robots are assigned to fixed initial positions as well as when such starting positions may be determined by the algorithm. We extend our consideration to the case when some number of robots may be unreliable. Our most surprising result is that solving the line exploration problem with robots at given positions, which may involve crash-faulty ones, is NP-

hard, while the same problem has polynomial solutions for a ring and for the case when the initial positions of the robots on the line are arbitrary.

The exploration problem is shown to be NP-hard for star graphs, even when the collection consists of only two reliable robots.

1.5 Thesis Outline

In chapter 2, we introduce a survey of the literature concerning mobile agents. We further divided the chapter according to the main focus of the papers surveyed, starting with general search and exploration problems. We then discuss search and exploration using multiple robots. Next, we survey rendezvous and gathering games, then introduce papers where the environment may change over time. We then present a section attempting to introduce the most important variations of problems involving search by multiple robots, and discuss the state of the art for those problems. We finally discuss some practical applications of this research.

In chapter 3, we introduce the exploration of the plane with multiple robots. We first discuss the case of non-faulty robots, then extend our model to include crash-faults, and finally, byzantine faults. For all cases, we discuss both the face-to-face and the wireless communication model. Algorithms as well as upper and lower bounds are presented for all cases.

In chapter 4, we discuss the problem of searching a ring for a non-adversarial, uncooperative mobile agent. We first discuss the case of search using a single robot, then generalise this situation to multiple robots, providing algorithms and bounds based on various conditions (prior knowledge of the speed of the bus relative to the speed of the robot; prior knowledge of the direction; number of robots at our disposal).

In chapter 5, we discuss the problem of evacuating a disc in the presence of faulty

robots. We consider the case of 3 robots, one of which can be either crash faulty or byzantine faulty, under the wireless communication model. For both fault models, we provide proofs of lower bounds, and non-trivial efficient algorithms.

In chapter 6, we discuss the problem of exploring graphs with time constraints by unreliable robots. We start by providing an algorithm based on dynamic programming to offer a solution to the problem of a single robot on the line, then apply the same logic to multiple robots on the line, for both given and arbitrary starting positions. We then discuss the case where some robots may experience crash faults. We discuss the ring environment and prove that such an exploration on graphs as simple as the star graph in NP-hard.

In chapter 7, we conclude and discuss open problems and interesting variations to the situations we presented in this thesis.

1.6 Publications

Publications upon which this thesis is based

Referred Conference Papers

1. J. Czyzowicz, M. Godon, E. Kranakis, A. Labourel, E. Markou. Exploring Graphs with Time Constraints by Unreliable Collections of Mobile Robots. In Proceedings of SOFSEM 2018, 44th International Conference on Current Trends in Theory and Practice of Computer Science, January 29 - February 2, 2018, Krems an der Donau, Austria, Springer LNCS.

2. J. Czyzowicz, S. Dobrev, M. Godon, E. Kranakis, T. Sakai, J. Urrutia. Searching for a Non-adversarial, Uncooperative Agent on a Cycle. In proceedings of Algosensors 2017, 13th International Symposium on Algorithms and Experi-

ments for Wireless Networks, September 7-8, Vienna, Austria, Springer LNCS.

3. J. Czyzowicz, K. Georgiou, M. Godon, E. Kranakis, D. Krizanc, W. Rytter, M. Wlodarczyk. Evacuation from a Disc in the Presence of a Faulty Robot. In proceedings of SIROCCO 2017, 19-22 June 2017, Porquerolles, France, Springer LNCS.

Submitted papers

1. J. Czyzowicz, M. Godon, E. Kranakis, A. Labourel, Searching the Plane with Faulty Robots, to appear.

Chapter 2

Survey of Literature

2.1 Introduction

The exploration problem has been central in the world of theoretical computer science for the past fifty years. This problem has known plenty of variations. In its most simple form, a single robot is exploring either a geometric environment or a graph in search of a treasure hidden somewhere within the environment. A very similar problem is often to explore the environment in its totality. In this survey, we will consider the exploration problem and some of its most popular variations.

2.2 Searching and Exploration by a Single Robot

The simplest variation of the exploration problem considers a single robot in an environment that can either be a graph (e.g. ring, tree, general graph) or a geometric environment (two-dimensional space, sometimes represented as polygons or a grid) is tasked with the exploration of the totality of this environment, sometimes with the purpose of drawing a map of this environment.

In some variations of this problem, the entire map of the environment can't be drawn, and the knowledge of the environment is then given by a so-called *quotient graph* (a quotient graph regroups all nodes in sets of equivalence classes, which in certain variations of the exploration problem is the most information a robot exploring an unknown graph environment can gather). In some instances of the problem, this exploration includes the traversal of all edges of a graph; in others, the visitation of all nodes is sufficient. In a geometric environment, a robot has an additional attribute: visibility. This visibility can either be unitary (or otherwise limited) or unlimited.

The efficiency of algorithms proposed use mostly two metrics: the number of edges traversed, and the amount of memory required by a robot.

2.2.1 Graph Environment

In the graph environment, the exploration problem is usually resolved when the robot has explored all edges of the graph. The common measure of efficiency of a given algorithm is its competitive ratio: the ratio between the amount of edge traversals required by an agent following the algorithm and the minimal amount of edge traversals required by an robot with perfect knowledge of the environment.

In [83], a single robot has for goal the exploration of a directed, strongly connected graph. The robot is able to differentiate a new point from one it has already visited, and can correctly identify the degree of each node it visits. The efficiency of the algorithm is measured by the ratio of edge traversed without prior knowledge of the graph compared to the minimal amount of edge traversals required with knowledge of the graph. The authors of [83] provide a tight bound of 2 for Eulerian graphs, and prove that this ratio is unbounded when the deficiency (the smallest number of edges that have to be added to the graph in order to make it Eulerian) of the graph is

unbounded. The authors also provide an algorithm with a number of edge traversals that is exponential in the deficiency of the graph.

A different measure of the efficiency of the algorithm is used in [112], where the efficiency of a robot exploring all edges of a directed, strongly connected graph is measured using a competitive ratio between the number of edges traversed by the robot and the minimal number of edge traversals required by a robot with full knowledge of the graph. The authors claim to introduce the first deterministic online algorithm with a competitive ratio polynomial in d the deficiency of the graph.

[88] considers the task of exploring all edges of an undirected connected graph by a single robot traversing as few edges as possible. The quality of a given algorithm is measured by the ratio between the number of edges traversed and the minimal number of edge traversals given knowledge of the graph. Such a ratio is referred to as the *overhead* of the algorithm for a given class of graphs when maximised over all possible starting nodes of the graph for this given class of graphs. The paper considers three possible scenarios: in the first one, the robot knows nothing about the graph; in the second, the robot has an isometric map of the graph without specification about the starting point; and in the third one, the robot is also aware of its starting point. For various classes of graphs, an algorithm is provided, often with an optimal overhead.

The first sub-exponential algorithm to explore an unknown environment represented as a directed, strongly connected graph is provided in [4], where an algorithm with a bound of $d^{O(\log d)}m$ is provided, with d the deficiency of the graph and m the number of edges. The authors also show that their algorithm is optimal by proving a matching lower bound for this variation of the exploration problem.

Simple Depth-First search (DFS) algorithms have been used to explore graphs in [166], with the number of edge traversals used as a measure for the efficiency of the

algorithm. They establish an upper bound of $\min(mn, dn^2+m)$ edge traversals, with d being the deficiency of the graph, m edges and n nodes.

DFS algorithms are also used in [128], where a robot has to solve the exploration problem (traverse all edges) in an unlabelled graph with no prior knowledge of the graph. In particular, there is no knowledge about the size of the graph. It is shown that a robot with constant memory K is unable to explore certain graphs with a maximum degree $d|d \geq 3$ and $K+1$ nodes. Moreover, a robot needs at least $\Omega(D \log d)$ bits of memory to explore a graph of diameter D and maximum degree d, and this is a tight bound.

In [93], the task of exploring a tree of maximum degree Δ is given to a robot that has a limited memory. It is observed that $O(\log \Delta)$ bits of memory are required to explore such tree if stopping is not required (that is, the robot does not have to stop at the completion of the exploration). If stopping is required, it is shown that bounded memory is not sufficient. Moreover, the authors show that $\Omega(\log \log \log n)$ bits of memory are required for some trees with n vertices. Finally, the paper considers a variation of the problem where the robot has to return to its starting node after exploration, and show that at least $\Omega(\log n)$ bits of memory are required to do so. An algorithm that matches this bound is provided in [138], thus solving the exploration of trees with $O(\log n)$ bits of memory.

The exploration problem is presented in a different light in [195], under the name of *Chinese Postman Problem* (CPP), and a variation for directed graph, the *Directed Chinese Postman Problem* (DPP). The standard definition of this problem requires for the agent to return to its original position after exploring all of the edges; the paper also introduces the *Open Chinese Postman Problem*, where the requirement of returning to the original position is dropped. The paper offers algorithms that resolves all those problems, but without optimising the amount of memory used.

Memory concerns are considered in [127], where it is shown, for a directed graph with n nodes of maximum out-degree d, that a minimum of $\Omega(n \log d)$ bits of memory are required by an agent to complete the exploration of the graph. The paper provides an algorithm that solves the problem with $O(nd \log n)$ bits.

In [39], a robot in an undirected graph has to visit all edges, then return to its starting point. The efficiency of the algorithm is measured by the number of edges traversed. Alternatively, the efficiency is measured by the minimal amount of memory required by the robot. In this paper, the robot knows a bound n on the size of the graph, and a bound d on the maximal degree of a node. The paper offers two algorithms, both aiming to improve the upper bound established in [65]. The first algorithm aims at a minimal number of edge traversals while the second tries to minimise the required amount of memory for the robot. Both algorithms represent an improvement compared to the upper bound established in [65].

[86] explores the information required by a robot in order to draw a complete or partial map of a graph. A complete map of a graph is understood to be an isomorphic copy of the map including its port numbers, and a partial map is understood to be a spanning tree with port numbers. The robot is forced to use a deterministic algorithm, and is unable to mark nodes in any way. It is proven that this map drawing is possible without further information if the graph is a tree. Otherwise, some bits of information, called *advice*, are required for the robot to construct either the complete or partial map. The paper establishes that the minimal size of the advice is linked to the number of nodes n of the graph, the number of edges m of the graph, and the multiplicity μ of the graph, that is, the number of nodes that have an identical view of the graph. Bounds on the minimal size of the advice for both the construction of a partial and complete map are provided. Tight bounds are provided for $\mu = 1$.

In [177], a robot in a connected weighted unlabelled undirected unknown graph

has to visit all nodes of the graph, without necessarily having to visit all edges. The robot only learns the weight of all edges when it is located in a node adjacent to those edges. The paper builds upon the work of [148], that presented an improved DFS algorithm that was 16-competitive on planar graphs. [177] prove that the aforementioned algorithm does not have constant competitive ratio on general graphs. Furthermore, the paper provides a constant competitive algorithm for general graphs with a bounded number of distinct weights.

2.2.2 Geometric Environment

In an interesting amalgam between the exploration problem and the illumination problem, [153] considers the problem of a single robot starting at an arbitrary vertex of a polygon and trying to find the shortest path inside the polygon that reaches a different arbitrary vertex, and provide an algorithm that is linear in the shortest distance between those two vertices, while also providing a 1.41 competitive ratio as a lower bound (the quotient between the distance travelled by the algorithm and the shortest path for their problem given knowledge of the environment).

A very similar problem was studied in [82], where an agent is tasked with the creation of the map of a room filled with obstacles. The proposed algorithm is measured by the competitive ratios between the distance required by the algorithm and the optimal distance, given knowledge of the environment. They show that there are no algorithms guaranteeing a competitive ratio for general rooms with obstacles, yet provide an algorithm for a room with a bounded number of obstacles. This bound is further improved in [154], which also provides an algorithm that is within a constant factor of optimal in the worse-case for arbitrary rooms with unbounded number of polygons.

[64] explores a similar problem: a robot has to explore a terrain with obstacles, both represented as polygons. Two scenarios are considered: in the first one, the robot has unlimited vision (though blocked by obstacles) whereas in the second one, the robot has a unitary vision range. The robot is tasked with the full exploration of the environment, and the efficiency of the algorithm used is measured based on the length of the trajectory of the robot. For both cases, an upper bound and tight lower bound are provided.

An interesting twist to this problem is considered in [155], where a robot attempts to localise itself in a geometric environment priorly known. They refer to this problem as the *location problem*, and builds upon the algorithms provided in [11]. The article also raises the question of landmark placement in an environment in order to simplify further attempts at localisation. A landmark is a point in the environment that can be used by a robot to localise itself. The algorithm provided in the paper is asymptotically better than the one discussed in [11].

The concept of unknown environment with landmarks is referred to as the *boundary place graph* in [194]. Using an algorithm that focuses on incremental construction of the environment, the authors manage to identify all landmarks in the environment. The algorithm described in the paper has been implemented in a mobile robots platform, thus proving the possibility to correctly explore an environment without the need for GPS-like systems or metric representation of the environment.

The search for a target located at an unknown location in a one-dimensional environment (the line) is known as the cow-path problem, or linear search problem, as introduced in [19], which proves that a minimum distance of $\Omega(9d)$, with d the original distance between the robot and a treasure, is required for a robot on a line to find the treasure.

The problem of exploration on the line by a single robot is considered in [81]

with an additional assumption: whereas most algorithms allow the robot to change its direction without cost, this paper adds a cost of d to each change of direction of the robot. An algorithm solving the problem in cost $9 \cdot OPT + 2d$, with OPT the optimal time with knowledge of the location of the target. The notion of turn cost is also applied to exploration on the star graph, and an algorithm solving this problem in optimal time is provided, with matching lower bound. Randomised strategies are discussed.

[149] considers randomised solutions to this problem, and conclude that the random algorithm they provide is optimal, using the total distance covered by the robot as measurement for the performance of their algorithm. The authors furthermore observe that their algorithm is almost twice as efficient as any deterministic algorithm.

A lower bound of $\Omega(\sqrt{n})$-competitive is proven for both deterministic and randomised algorithms in polygonal environments in [5], and is generalised to tri-dimensional rectilinear polyhedra without obstacles.

Simple greedy algorithms are used to explore unknown terrains in [156], and even though they do not yield optimal results (as is to be expected), they offer acceptable results. The authors discuss applications and advantages of the usage of such an algorithm, and observe that it can take advantage of prior knowledge of the environment, and offer the possibility to be applied to multiple robots acting in cooperation.

An A* approximation algorithm is used in [172] to determine the path followed by a robot with limited visibility in an unknown environment discretised as a two-dimensional grid. Experimental results are provided to discuss the efficiency of the algorithm.

2.3 Collective Search and Exploration

The search and exploration problems were studied under many variations, and some of those involve multiple robots. The classical problem of exploration can be modified by the addition of one or more robots. Sometimes, a graph that could not be entirely mapped with the use of a single robot can now be completely recognised by a team of collaborating robots. The presence of multiple robots allows for various strategies, such as the introduction of immovable robots that serve as markers to help another robots orientate in a given environment. Those immovable robots are often referred to as pebbles, or sometimes, when they can store and share information, as whiteboards. The presence of multiple robots introduces a new variable for the studied problem, that is, the communication between autonomous robots. The four most studied variations of a communication model are the face-to-face communication, where robots can only share information when they are located at the exact same location; the wireless communication, where robots can freely exchange communication with no regard to their respective location; the pebble communication, where robots are unable to communicate, but may leave markers on the ground that contain no information; and finally, the whiteboard communication, where robots may leave markers on the ground that do contain a certain amount of information that can be read by other robots.

The ring environment is discussed in [114], where a group of k identical robots are tasked with the exploration of all nodes of a ring of size n. In this model, robots are equipped with sensors, but are unable to communicate. Each node must be visited by at least one robot, and all robots must decide to remain idle for the algorithm to be completed. It is shown that this problem is impossible if k and n are co-prime (the only common divisor between k and n is 1). The authors also show that the

minimum number of robots required to explore a ring of size n is $O(\log n)$.

[22] considers the exploration of a graph environment with two robots, and provide an algorithm, called the homing-sequence algorithm, to accomplish this in a time that is polynomial in n the number of nodes. They compare the performances of this algorithm to a random walk algorithm, and conclude that the random algorithm has better performances than the homing-sequence algorithm.

In [99], a group of robots equipped with wireless communication are tasked with the exploration of all nodes in a graph, and must then return to their initial position. The robots have no prior knowledge of the graph. The efficiency of the algorithm is given by the competitive ratio between the total time required by the algorithm and the smallest amount of time required, given knowledge of the graph. A lower bound of $\Omega(\log k/\log\log k)$ with k the number of robots is introduced for the competitive ratio of any given deterministic algorithms. Using the number of edges traversed instead of the elapsed time, the authors present an algorithm with a competitive ratio of $(4-2/k)$ on trees.

A linear algorithm for the exploration of all nodes of a graph by a group of robots is provided in [84]. More precisely, the paper considers a scenario in which a group of robots can explore a n-node graph located at a distance at most D of the starting point of the robots in $O(D)$, if the number of agents k is polynomial in D and n ($k = Dn^{1+\epsilon} < n^{2+\epsilon}$ for any $\epsilon > 0$). This algorithm works even with a local communication model in which robots can only exchange information when they are located at the same node.

[132] classifies decision problems for multiple robots in a graph environment using deterministic algorithms. It is proven that the class of all problems that can be verified with a certificate is much wider than the class of all decidable problems in this context, and the paper shows the existence of a *Mobile-Agent Verifiable*-complete

problem.

An alternative communication model is discussed in [189], where robots with wireless communication have a limited range of communication. An algorithm based on a distributed bidding model is introduced, using two measures: (1) the distance between robots, and (2) a map synchronisation mechanism; both of which are meant to reduce the amount of information exchanged between robots. Simulations are used to measure the efficiency of the algorithm.

A similar bidding model is used in [165], to the difference that in this paper, communication is impossible between robots. More so, [165] is concerned with collision between robots, and the bidding value function used to determine the path of all robots also aims at avoiding collisions.

2.3.1 Exploration Using Pebbles and Whiteboards

A popular variation of the exploration problem sets either a single robot or a group of robots in a graph environment that they must fully explore. In addition to their own capabilities, robots can be equipped with one or more pebbles: movable devices that are used to uniquely identify a node or an edge. Alternatively, an environment in which every node is equipped with a whiteboard is considered. Pebbles were a single unary bit of information; whiteboards serve as bookkeeping devices, and as such, are able to conserve a certain number of bits of information.

The problem of graph exploration using a single robot equipped with pebbles is discussed in [21]. The paper considers an unlabelled strongly connected directed general graph. It is shown that robot with knowledge of an upper bound on the number of nodes of the graph can explore it using only one pebble. If this assumption is dropped (that is, the robot has no prior knowledge of the graph), then at least

$\Omega(\log \log n)$ pebbles are required and are sufficient for the complete exploration of the graph.

[129] first considers the problem of exploration of all edges of a graph with a group q of non-collaborative K-state robots, and show that there is a graph of size $O(qK)$ that cannot be explored by any robot from the group. By applying this principle, the authors consider the problem of exploration with stop (the robot has to stop its exploration once it is done), and provide the robot with a single pebble. For graphs of size at most n, a robot with $\Omega(\log n)$ bits of memory can solve the exploration problem with stop. The authors also prove that there is an algorithm to solve the exploration problem with stop requiring a robot with $O(D \log \Delta)$ bits of memory, with D the diameter of the graph and Δ the maximal node-degree in the graph.

[96] considers the exploration of a graph by a single robot equipped with one or more pebbles in the absence of any further information about the graph, and provide experimental results to show the correctness of the exploration algorithm provided.

A variation on the standard pebble model is presented in [24], where a single robot represented as a Finite State Automaton is tasked with the exploration of an unknown directed graph. Rather than placing a pebble, however, the robot can, at every node, place a pebble on one of the exit ports of the nodes, thus serving as traffic signals. Two algorithms are studied: the first one solving the exploration in $O(|V||E|)$ edge traversals; and the second one traversing every edge three times.

In [127], it is established that a robot with no memory is unable to achieve the exploration of a directed graph. In order to alleviate this problem, the authors provide an algorithm that adds to every node of out-degree d a whiteboard with a memory of size $O(\log d)$, and prove that this bound is tight for an agent with constant-size memory.

The exploration of a graph by a group of robots with the help of whiteboards which

serve as communication devices is considered in [33]. Tree and graph exploration are examined and results are corroborated with simulation results. In this paper, whiteboards are not immovable devices fixed at a given node, but are rather devices that can be moved freely by the robots and can be dropped at any node.

A slight variation of the exploration with whiteboard problem is presented in [75]. In this instance of the problem, a group of k robots have to build identically labelled maps of a n-nodes graph. They are unable to communicate directly, but can freely read and write on whiteboards located at every node. In some cases, this problem is shown to be unsolvable. An algorithm that solves the problem is presented under the assumption that n and k are co-prime.

[76] is also interested in the problem of common map building by a group of robots in an unknown graph. The paper provides the robots with either the number of nodes n in the graph, or the number of robots k. The efficiency of an algorithm is measured by the total number of edge traversals by all the robots. Respecting the condition established in [75] that n and k have to be co-prime, the algorithm provided solves the problem in $O(km)$ with m the number of edges of the graph.

The exploration of the tree by a group of robots is considered in [126]. It is stated that scheduling an optimal collective exploration is NP-Hard, even when the tree is known. Algorithms are provided for the exploration by a group of robots under three conditions: in the first one, robots benefit from perfect wireless communication; in the second one, robots can use whiteboards located at every node to communicate; and in the third one, robots are completely unable to exchange information.

2.3.2 ANTS

The problem of *Ants Nearby Treasure Search* (ANTS), first introduced in [110], is a generalisation of the cow-path problem. The authors of [110] describe the problem as follows: a group of k identical agents are initially placed at the origin of a two-dimensional infinite grid. Somewhere on this grid, at distance D of the starting position is a treasure hidden by an adversary. The objective of any given algorithm is to find this treasure as fast as possible. Communication settings vary, but are generally limited; sometimes communication is altogether impossible. The robots do not necessarily start the execution of their algorithm at the same time. In this first paper, the authors show that the time required to find the treasure is $\Omega(D + D^2/k)$, and they provide an algorithm that matches this bound under the condition that the robots are aware of the value of k. They also present a tight bound for the competitive penality that must be paid to compensate for the lack of knowledge of k by the robots.

A variation of the ANTS problem is provided in [106] and [105], where the robots are represented as asynchronous randomised Finite State Automaton: they possess constant-size memory, and can communicate locally (when two robots are at the same point of the grid) through constant-size messages. They show that those restrictions do not diminish the performance of the robots, and provide an algorithm that matches the bound provided in [110]. This variation is further explored in [103], where the minimum number of robots required to accomplish various exploration scenarios are discussed. Those scenarios consider different computational capabilities for the robots, as well as different timing parameters. For all scenarios, upper and lower bounds are provided.

Communication between robots is further restricted in [179], where the only form of communication allowed is *loneliness detection*, where a robot located at a node is

able to tell whether it is alone or shares the node with one or more other robot. In this context, robots are still represented as Finite State Automaton, thus possessing constant-size memory. An algorithm solving the ANTS problem under such circumstances is provided, finding the treasure in time $O(D \log k + D^2/k)$, thus beating the lower bound for agents that are unable to communicate.

[108] explores the relationship between the memory of the robots and the running time performances of the algorithm. More specifically, for various time performances, the paper establishes a lower bound on the memory size of the robot.

A new metric to evaluate the performance of an algorithm is introduced in [171]. This metric, called the *selection complexity*, represents *how likely a given algorithmic strategy is to arise in nature due to selective pressures* [171]. For this metric, an algorithm working in asymptotically optimal time, taking the fineness of available probabilities into account.

A variation of the problem called *ant colony house-hunting* is introduced in [139]. The objective of the algorithm is to explore a terrain nearby the origin in order to find a location that is regarded as "best". In order to do this, all robots must share the task of exploring the environment, then come to a consensus and move to the chosen location. The paper shows that a time of at least $\Omega(\log n)$ with n the number of robots is required for the formation of a consensus. Two algorithms are provided to solve this problem: the first one runs in optimal time $O(\log n)$, but is described as non characteristic of natural ant behaviour. The second one runs in time $O(k \log n)$ and is simpler and closer to the natural behaviour of ants.

Animal metaphors similar to the ANTS problem are used in [202], where a group of robots equipped with constant-size memory is tasked with the cleaning of a dirty floor represented as a non-convex region of Z^2 that may include obstacles. Interestingly, in this scenario, the only mean of communication of the robots is the cleanness of the

floor itself. Experimental results are provided.

At the intersection between ANTS problems and problems using whiteboards is [1]: in this paper, k robots modelled as either Finite State Automata or Turing Machines located at the origin of an infinite two-dimensional grid are tasked with the identification of a treasure hidden at a distance D from their starting location. Each robot is able to place information, referred to as *pheromone*, at its current location. This pheromone can then be read by other robots, thus allowing communication. For robots modelled as FSM, $\Omega(D)$ pheromones are needed for the identification of the treasure; this bound lowers to $\Omega(k)$ when the robots are modelled as TM. Algorithms matching those bounds are provided, and solve the problem in $O(D + D^2/k)$, which is proven to be optimal. The pheromones are also used as fault-tolerance mechanism, which will be discussed later on.

Faults are introduced in the ANTS problem in [192] and [104]. In this variation of the problem, up to f robots may be faulty and stop working at an arbitrary time. The authors present an algorithm that is able to detect and recover from failures, and that performs in time $O(D + D^2/n + Df)$ with D the distance to the treasure and n the number of robots.

[10] adapt the exploration problem to require one more step before the completion of any algorithm. Once robots have identified their target on the grid, they must communicate this information to other robots, then transport the treasure back to the nest. Five algorithms are provided for the exploration phase, and their competitive ratio is discussed. Simulation data is provided to support findings.

2.4 Search, Rendezvous and Gathering Games

One of the most common variation of the group search problem is the rendezvous problem [7]. In the rendezvous problem, two agents located at distinct starting position are tasked with meeting one another. This is significantly different from the exploration problem, where the target was immovable, or from the cops and robbers problem (discussed later), where the target is actively trying to avoid being caught. In this variation of the problem, both agents collaborate in order to ensure a meeting. This problem has been studied under many variations: graph and geometric environment, synchronous and asynchronous movements of the robots, variable starting time and moment of apparition in the environment, memory restrictions for the robots, presence or absence of a common compass, randomised or deterministic algorithms, and limited visibility in the geometric environment are some of the most common variations. The problem of ensuring a meeting between n robots is known as the *gathering problem.*

2.4.1 Rendezvous and Gathering

[131] studies the amount of memory that is required for two robots to meet in a tree. It is first established that rendezvous is only possible if the initial position of the agents is not symmetrical. For a tree of size n, it is shown that $\Theta(\log n)$ bits are required and sufficient to ensure the rendezvous. The paper further shows that it is impossible for two robots represented as Finite State Automaton to rendezvous in all bounded degree trees.

In [16], two asynchronous robots are located in an infinite grid of dimension $\delta > 0$ embedded in the Euclidean space. Both robots share a common compass and are both equipped with a GPS-like device - in other words, they are both aware of their

respective position compared to the origin, and share the orientation of the spaces on all dimensions. An algorithm ensuring the rendezvous in time $O(d^{\delta}polylogd)$, which is close to the lower bound $\Omega(d^{\delta})$. Furthermore, this problem is expanded by providing the robots with a visibility radius. This visibility radius may be different for both robots, and the condition for the rendezvous is that both robots are within the visibility radius of the other. An algorithm allowing asynchronous rendezvous in $O((\frac{d}{r})^{\delta}polylog(\frac{d}{r}))$ with $r = \min(r_1, r_2)$ is provided.

[63] models an unknown bounded terrain as a polygon in which two robots have to meet. An adversary determines the walk of the robot (that is, the adversary does not choose the path of the robot, but chooses the speed at which the robot travels it, with the possibility of using negative speed). Algorithms that ensure rendezvous are provided for the following conditions: the presence or absence of obstacles in the environment; robots having agreeing or disagreeing compasses; and robots having a prior map of the environment with information about their exact starting position or not. The efficiency of the algorithm is determined by the length of the longest path. Lower bounds are provided for all conditions, and all bounds are shown to be tight.

In [78], labelled asynchronous robots have to meet in an unknown anonymous graph. As for almost every asynchronous rendezvous problem, rendezvous is allowed inside the edges. The efficiency of an algorithm is measured by the number of edge traversals required before the rendezvous happens. The first case studied in the paper is the infinite line: in this case, the rendezvous can be ensured after $O(D|L_{min}|^2)$ with D the original distance between the robots and L_{min} the smallest label of a robot, provided that D is known. If D is unknown, $O((D + |L_{max}|)^3)$ is required. The second case studied is the ring, on which those bounds are still valid. An optimal algorithm requiring $O(n|L_{min}|)$ is also provided when n is known and $O(n|L_{max}|)$ for n unknown. Finally, for arbitrary graphs, it is shown that the rendezvous is possible

with knowledge of an upper bound on the size of the graph, and an optimal algorithm requiring $O(D|L_{min}|)$ is provided, requiring prior knowledge of the topology of the graph and the initial positions of both robots.

[92] considers the rendezvous problem in an unknown graph by asynchronous labelled agents. The performance of an algorithm is measured by the number of edge traversals required before the rendezvous happens. [92] is the first paper to provide an algorithm polynomial in the size of the graph and the size of the smaller label.

The use of *Universal Traversal Sequences* and *Universal Exploration Sequences* to solve rendezvous problems with asynchronous labeled robots are first introduced in [193], and are used to improve previously existing upper bounds for this problem. This paper is based on the work of [185].

The problem of rendezvous by synchronous anonymous agents in a graph is studied in [65] in relationship with the minimum amount of memory required by the robots for the rendezvous to be possible, using the results discussed in [193]. It is shown that $\Theta(\log n)$ bits of memory are required and sufficient to ensure rendezvous on a graph of size n. This holds even if the agents do not appear on the graph at the same moment.

A survey of deterministic rendezvous algorithms in graphs is presented in [181].

In [71], two labelled asynchronous robots have to meet in a possibly infinite connected graph or in an unknown terrain in the plane. A deterministic algorithm ensures the rendezvous under the following *sine qua non* conditions: in the graph, only connectedness is required. In the geometric scenario, it is required that the terrain is closed, the robots start at interior points of the terrain, and the starting point of the agents have rational coordinates.

A correlation between required time and required memory is made in [161], where two synchronous non-labelled robots have to meet on the anonymous, oriented ring.

It is shown that robots with $2t$ states can rendezvous on an n nodes ring in time $O(n^2/2^t + 2^t)$, and any pair of robots with $t/2$ states require at least $\Omega(n^2/2^t)$ to ensure rendezvous. It is furthermore observed that $\Theta(\log \log n)$ bits or memory are required to ensure meeting in linear time.

In [41], synchronous labelled robots are trying to meet in a graph of size n, without bounds on the size of n. Robots know their own label, but not the label of the other robot. Furthermore, the notion of *delay fault* is introduced: at each round, it is possible for a robot to be subject to a delay fault, in which case the agent will remain idle for the entirety of the round. The robot is aware of the fault, and can react accordingly. Three fault scenarios are considered: the random scenario (at each round, robots have a probability $0 < p < 1$ of being delayed); the unbounded adversarial scenario (an adversary may delay the agents indefinitely); and the bounded adversarial scenario (an adversary may delay a robot for at most c consecutive rounds), and c is unknown to the robots. The performance of any given algorithm is measured by the number of edges traversed. For random faults, an algorithm polynomial in n and polylogarithmic in the larger label L is provided. It is shown that rendezvous is not possible in the unbounded adversarial scenario, even on rings. Finally, for bounded adversarial faults, an algorithm polynomial in n and logarithmic in c and L is provided.

Rendezvous between heterogeneous robots is studied in [85]. Heterogeneous is understood as follows: the environment is a weighted connected graph; however, the weights are not the same for both agents. Agents have full knowledge of the graph, as well as their own starting position and the starting position of the other robot. They are now aware of the cost of edge traversals for the other robot, and have to ensure a rendezvous for an edge-traversal cost that is as low as possible. Meeting is allowed both in an edge an in a node. The efficiency of the algorithm is measured

as the ratio between the required time and the optimal required time for the offline scenario, where the robots have knowledge both of their own costs for edge traversals and the cost for the other robot. An algorithm ensures rendezvous in an n-nodes graph in $O(nT_{opt})$, where T_{opt} is the optimal offline time. It is furthermore shown that if the agents have the capability to exchange n bits of information at the start of the algorithm, this bound can be lowered to $\Theta(T_{opt})$.

In [50], rendezvous of two anonymous synchronous robots that are aware of their own starting position is studied for two environments: the graph environment and the geometric environment. The efficiency of algorithms is measured by the number of synchronous rounds that pass before the rendezvous happens. As for most synchronous studies, rendezvous must happen in a node, and is not allowed inside and edge. Is is shown that on the line, the tree and in multi-dimensional Euclidean spaces and grids, agents can rendezvous in $O(d)$ with d the initial distance between the agents. In all n-node graphs, the rendezvous can happen in $O(d \log^2 n)$, and there exists an infinite family of graphs in which rendezvous in this setting requires at least $\omega(d)$.

Rendezvous of asynchronous agents with limited (unit) visibility in the two-dimensional Euclidean space is discussed in [51]. The paper considers agents equipped with a common compass, and are aware of their own starting position (given by a set of coordinates). The efficiency of an algorithm is measured by the sum of both agents' travelling length. Rendezvous is considered achieved when both agents are within line of sight of one another. An algorithm solving this problem in $O(d^{2+\epsilon})$, with d the distance between the agents, is provided.

[87] considers the rendezvous for synchronous labelled agents in the graph. The paper studies two scenarios: in the first one, both agents start the execution of their algorithm simultaneously, and in the second one, there is a delay chosen by and

adversary between the start of the algorithm of one robot and the apparition of the second robot on the graph. The efficiency of an algorithm is measured by the number of steps from the apparition of the second robot to the rendezvous for a given initial configuration. In trees, it is shown that $O(n + \log l)$ is required, with l the smallest label, for arbitrary startup time. On a ring with simultaneous startup, $\Theta(D \log l)$ is required, with D the initial distance between the agents. On general graphs, an algorithm that ensures rendezvous in time linear in n, τ and logarithmic in l is provided, with τ the delay between the start of the first robot and the apparition of the second robot.

[159] studies the gathering problem for k labelled synchronous robots in a connected graph with possible delay in the startup time. It is shown that this is no harder than the same problem for two robots, and an algorithm is provided to support this claim. The efficiency of the algorithm is measured by the number of steps required before all robots meet. An asymptotically optimal algorithm working in $O(n \log l)$ with l the smallest label is provided for the ring environment, and an algorithm polynomial in n and l and independant of τ the difference between startup times is provided for general graphs, thus improving the bound presented in [87].

In [90], the gathering problem for synchronous anonymous robots in the graph is studied. The number of robots k is unknown. All gatherable configurations are identified, and two universal gathering algorithms are provided. Those algorithms allow the gathering to happen for all configurations in which such gathering is possible.

[173] studies the problem of searching m concurrent rays by a group of p robots, in order to identify a treasure located on one of the rays. The efficiency of an algorithm is measured by the ratio of time required by an algorithm without knowledge of the location of the treasure to time required with prior knowledge of its location. An optimal algorithm achieving a competitive ratio of $1 + 2/(m/p - 1)(m/(m - p))^{m/p}$

is provided.

2.4.2 Look, Compute, Move Model

We consider robots modelled as points in their environment that are continuously obtaining information from their surroundings, and move either in synchronous fashion, or have their walk controlled by an adversary, which means that though the path they would take is determined by the algorithm and could not be steered from, the speed they use to walk on it is controlled by an adversary (which could use negative speeds as well) at all time, provided that they covered the entirety of the path. The following papers consider a different model: in this model, robots will in turn obtain information about their surroundings, often obtaining information with perfect visibility of the entirety of the environment (LOOK), use a deterministic algorithm to decide of their next destination (COMPUTE), and finally, move in a given direction (MOVE). The notion of synchronism is different here as well: for robots to be asynchronous means that they execute their LCM cycles independently from one another. If they are synchronous, they start each LCM cycle exactly at the same time. This model also allows for the introduction of a new synchronism pattern: the semi-synchronous model.

The problem of searching the plane by a mobile agent has first been identified as a field of study in [11]. The model used can be considered a precursor to the Look-Compute-Move (LCM) model that will be discussed later on. In this paper, a mobile robot is tasked with a search problem in an unknown, unbounded environment, and its performances are evaluated by considering the total distance covered between every probe position taken. Various scenarios are explored, with varying level of *a priori* information given to the robot, such as the distance to the target and the

general direction of the target. The paper also examines the possibility of errors in the interpretation of the environment by the robot.

The gathering problem for asynchronous robots operating in LCM cycles is studied in [72]. Robots are equipped with multiplicity detection: during their Look phase, they see if any node is occupied by zero robots, one robot, or two or more robots (without knowing the specific amount). This paper identifies the initial configurations in which the gathering problem can be solved, and provides an algorithm that generates a solution whenever possible.

The graph exploration problem by a swarm of robots operating in asynchronous LCM cycles is studied in [115]. In a given laps of time, the swarm of robots must explore all nodes, then put an end to their activities (quiescent state). The efficiency of the algorithm is measured by the number of robots required to complete the exploration in the given time. It is shown the even in n-node trees with a maximum degree of 4, $\Omega(n)$ robots are necessary to complete the exploration. If the maximum degree of the tree is 3, $O(\frac{\log n}{\log \log n})$ robots are sufficient, and this solution is asymptotically optimal.

[191] studies rendezvous of two asynchronous oblivious anonymous robots in the two-dimensional Euclidean space in the absence of a common coordinate system. An algorithm that allows for rendezvous in a finite number of cycle is provided under the assumption that the compasses differ from at much $\pi/4$.

Based on the fact that rendezvous between two semi-synchronous robots is trivially solvable when their coordinate systems are consistent, [145] explores the magnitude of consistency between coordinate systems required robots to ensure rendezvous in the semi-synchronous and asynchronous model. In the paper, robots have unreliable compasses: their bearings may deviate from an absolute reference direction. From there, two scenarios are considered: in the first one, the compass is static, and the

deviation remains the same throughout the execution of the algorithm. In the second one, the compass is dynamic, and the deviation may vary at the start of every new LCM cycle. For robots with static compass in both the semi-synchronous and asynchronous setting, a deviation of $\Phi < \pi/2$ allows for a rendezvous; a deviation of at most $\Phi < \pi/4$ is required for semi-synchronous with dynamic compasses to meet; and a deviation of at most $\Phi < \pi/6$ is required for asynchronous with dynamic compasses to meet.

In a similar fashion, [144] considers gathering of more than two robots with unreliable compasses, both for the case of static and dynamic compasses, under the semi-synchronous setting. An algorithm allowing gathering is provided for a compass that accepts deviations of up to $\pi/2 - \epsilon$, with $\epsilon > 0$. This algorithm is proven to be optimal.For any deviation greater than $\pi/2$, it is proven that no algorithm can ensure either rendezvous or gathering.

[200] considers anonymous, oblivious robots operating in LCM pattern with an interesting added feature: both robots carry coloured lights that are visible to both robots. They are placed either in the plane or on a line, and their purpose is to reduce (or increase) the distance between them by a constant factor without using distance information. A solution is provided with specifications on the number of colours required to ensure rendezvous. Every synchronism model is explored. In the asynchronous model, three colours are enough to ensure rendezvous for any initial configuration.

In [123], anonymous robots have to meet in the plane under the asynchronous setting. Two settings are studied: in the first one, robots are incapable of remembering the layout of the environment in their previous cycle, but can communicate with constant-size messages (*finite-communication model*), and in the second, robots can remember the layout of the environment in their previous cycle using constant-size

memory, but are unable to communicate(*finite-state model*). Those settings can be modelled as robots carrying lights: in the finite-communication model, robots can only see the other robot's light, whereas in the finite-state model, robots can only see their own. It is shown that finite-communication model allows rendezvous for asynchronous robots, and finite-state model allows rendezvous for semi-synchronous robots.

The relationship between synchronism models is studied in [77], under the assumption that robots are equipped with lights of different colours that are persistent - that is, the light is not automatically reset at the end of each LCM cycle. It is shown that asynchronous robots with a constant number of colours are more powerful than semi-synchronous robots without colours. Furthermore, it is shown that there is no difference between asynchronous and semi-synchronous robots when they are equipped with visible lights. Asynchronous robots with visible lights are also more powerful than synchronised robots under the assumption that they have the ability to remember a single snapshot of the graph.

Census of the capability of robots operating in LCM cycles under synchronous, semi-synchronous and asynchronous settings are presented in [184] and [182], and the problems of rendezvous, gathering and pattern formation are discussed.

Gathering in the ring environment is discussed in [152], under the asynchronous setting. It is shown that for an odd number of robots, gathering is only possible if their initial configuration is not periodic, and an algorithm is provided to solve the gathering problem. For even number of robots, the feasibility is decided except for one type of symmetrical initial configurations, and algorithms that ensure gathering are provided.

The impact of symmetries on gathering possibilities for asynchronous robots in the graph are studied in [151]. On an undirected ring with at least 18 robots, it is

proven that an approach that focuses on preserving symmetries solves the gathering problem for all starting positions when the gathering is feasible.

[46] studies the gathering problem for oblivious asynchronous disoriented (no common compass) robots operating in LCM cycles in the two-dimensional Euclidean space. The paper solves the gathering problem for $n > 2$ robots for any initial configuration, even under such weak assumptions.

In [183], the minimal assumptions required for anonymous oblivious robots without communication to gather in the two-dimensional Euclidean space are studied. It is shown that in the synchronous case, robots require either multiplicity detection or infinite time, and in the asynchronous case, robots require either multiplicity detection, a common compass, unbounded memory or infinite time in order to meet.

The assumption of unlimited visibility is dropped in [121]. The paper considers the gathering problem for asynchronous oblivious robots operating in LCM cycles. The paper provides an algorithm that allows gathering in a finite amount of time for robots with limited visibility, provided that they share a common compass. From this result, the authors show that orientation is as powerful as instantaneous movement with respect to gathering.

Similarly, in [80], the gathering problem with synchronous robots with limited visibility in the two-dimensional Euclidean space is considered. The robots must, when choosing their next move, ensure that the unit disk graph defined by the viewing range of the robots remains connected at all time. An algorithm that ensures gathering in time $\Theta(n^2)$ is provided for the gathering.

In [74] and [73], oblivious robots operating in LCM cycles are placed on the discrete ring. They are asynchronous, and during their Move phase, they can decide either to stay in place or to move to an adjacent node. Two problems are considered: the gathering problem, and the *exclusive searching* problem, in which all edges are

either traversed, or both nodes adjacent to an edge are occupied, without two robots ever occupying the same node. A description of the initial configurations that allow resolution of those two problems is provided, and algorithms solving the problems are provided.

The notion of *rendez-vous with detection* is explored in [102], where two agents have to meet in a graph and then become aware of this meeting. They must then declare simultaneously that the meeting took place, then stop. This paper uses the synchronous model. Two variations are considered: the local beeping model, and the global beeping model. Feasibility is discussed under various premisses, and algorithms are provided when possible.

2.4.3 Deployment

Rendezvous and gathering problems can be seen as a specific subset of a larger problem: the pattern formation problem. In the pattern formation problem, a group of $k \geq 2$ robots must form a specific pattern. This includes rendezvous, but also the formation of a regular k-gon, a formation in which no more than two robots form a line, a formation in which robots are equidistant, etc.

In [89], robots operating in LCM cycles are located in the two-dimensional Euclidean space. Two robots can only see one another if there is no other robot between them. Their purpose is to develop a formation in which all robots can see all other robots, in a finite laps of time. This problem is called the *Mutual Visibility Problem*. Each robot is equipped with visible lights that can take up to c different colours. Various scenarios are considered: robots with prior knowledge of their environment, number of available colours, and synchronism of the robots. It is proven that in the semi-synchronous case, the Mutual Visibility Problem can be solved without collision

with $c = 2$, and with $c = 3$ if the robots are asynchronous.

[101] considers the even deployment of oblivious agents with no prior knowledge of the environment on the discrete ring. The size of the swarm is unknown as well. Two variations of this problem: the first one, called *dynamical uniform deployment*, requires for the agents to spread evenly, then keep moving as they hold this formation. The second variation, called *quiescent spread*, requires for the agents to stop moving once they are evenly spread. For the first variation, it is shown that the problem can only be resolved if either the ring is oriented, or the agents have a visibility range of at least $\lfloor n/k \rfloor$. An optimal algorithm is proposed for an oriented ring and agents with a visibility range of at least $\lceil n/k \rceil$. For the second variation, the problem cannot be solved if the agent can only measure the distance to its two neighbours.

In [120], a set of $n \neq 4$ asynchronous anonymous oblivious robots with no common compass must arrange themselves to form a regular n-gon. This problem is called the *uniform circle formation problem*. The paper proves that this problem can be solved without any further assumption for any initial configuration.

Arbitrary pattern formation is studied in [122], where asynchronous oblivious robots working in LCM cycles operating in the two-dimensional Euclidean space must form a pattern that is given in advance. It is show that without common compass, this task is impossible. With a compass, an odd number of robots can solve this problem, but not an even number of robots. With two different compasses (say, one pointing North and one pointing East), then an even number of robots can solve the problem.

In [205], robots operating in LCM cycles must form a predetermined pattern. It is shown that oblivious robots can form any pattern non-oblivious robots can, with the exception of two robots trying to form a point, which is only possible if the robots are not oblivious. The paper therefore proves that memory is not useful in the task

of pattern formation, apart from the aforementioned exception.

2.5 Varying Environments

So far, situations in which the environment was static were considered. In this section, we consider variations where the environment may present a danger or be destructive or misleading. We study time-varying graph, where the edges between nodes appear and disappear; environment with black holes, where some nodes or edges of a graph immediately destroy any robot that travels it; and scenarios where the agents themselves may be subject to faults. Two main type of faults are generally considered: in the crash fault scenario, robots may lose the capability to move, communicate or sense their environment correctly. In the Byzantine fault scenario, robots controlled by an adversary may decide to purposefully share wrong information and fail to convey the appropriate information, or may change their path arbitrarily. Those situations call for algorithms that include an element of resilience, and have a significant impact on the feasibility and bounds of many well-studied problems.

2.5.1 Time-Varying Graphs

Time-varying graphs (TVG) are graphs whose edges may be existent or non-existent, as a factor of time. In most TVG, there is a cycle that dictates the moments where edges are existing and non-existing, that repeats itself periodically. [38] presents an effort to unify concepts, formalisms and results concerning various aspects of TVG, all the while providing a hierarchical classification of those variations.

In [118], the exploration problem by a single robot on a periodically varying graph is studied. The edges of the graph go from existing to non-existing on an unknown, periodic basis. Various scenarios are considered, depending on knowledge of the

length of the longest route, memory of the robot, knowledge of a bound on the size of the graph and uniformity of the length of the routes. Necessary conditions for the exploration problem to be solvable are established.

[188] model social networks as time-varying graphs in order to provide tools to understand and evaluate their dynamics, and use this perspective to model concepts such as distance and connectivity.

[204] proposes a model of TVG that has asymptotic memory complexity in the order of complexity of the set of edges, and can be used to represent many previously existing models, while being intrinsically able to model cyclical behaviour as well.

The spreading of k tokens of information to all nodes of a TVG is studied in [98]. In this model, each round, each node is able to broadcast one token of information to all its neighbours. The problem is solved in $O(n+k)$ in static graphs of size n. The paper shows that at least $\Omega(nk/\log n+n)$ rounds are required for the information to spread to every node for any randomised algorithm when the adversary is strongly adaptive, i.e. the adversary chooses the available edges with knowledge of the information that will be disseminated by every node. The weakly adaptive model forces the adversary to choose the layout of the network first, then allows the algorithm to choose which token of information to broadcast, and for this model, an algorithm spreads all token of information to all nodes in $O((n+k)\log n \log k)$ rounds with high probability. If the entire sequence of graph layout is known in advance, an algorithm solves the problem in $O((n+k)\log^2 n)$.

In [119], the exploration problem is studied for *carrier graphs*, where the edges between nodes only exist at unknown periodic times. Conditions required for the problem to be solvable are established. Lower bounds for the amount of time required are presented, and matching upper bounds are provided.

2.5.2 Black Holes

In the black hole problem, a team of agents must identify a black hole on a graph. A black hole refers to a node or an edge that will instantly destroy any robot that travels on it without leaving any trace. The purpose of any algorithm trying to solve this problem is to identify the location of the black hole without entering it. The efficiency of an algorithm is usually measured in one of three ways: either by the amount of robots required to identify the black hole; by the time required before the black hole is found; or by the amount of additional resources required (pebbles and whiteboards being the most common example). In the usual scenario, all agents have the same starting location. Variations of this problem include synchronism between agents, possibility to use a pebble or a whiteboard, prior knowledge of the topology of the graph and memory of each agent.

[116] examines the use of pebbles in the search for a black hole in a graph of known topology by two asynchronous agents. It is proven that the pebble model has the exact same complexity as the whiteboard model. Furthermore, two agents equipped with one pebble each can locate the black hole in $\Theta(n \log n)$, using a technique referred to as *ping-pong*.

The situation where a team of synchronous agents have to locate a black hole in a tree is studied in [66]. Algorithms are provided, and their efficiency is discussed.

In [95], a team of asynchronous agent with no prior knowledge of the graph must identify a single black hole. Agents are equipped with whiteboards. It is proven that $\Delta + 1$ agents are required and sufficient to identify the black hole for Δ the maximum degree of a node, and that $\Theta(n^2)$ is required to find the black hole in a graph of size n. With sense of direction (where each port of an edge is labelled in such way that agents can determine whether two edges lead to the same node), only two agents are

required to find the black hole in $\Theta(n^2)$.

Various graph environments are studied in [94]. The paper proves that the black hole can be identified in linear time by two asynchronous agents in hypercubes, cube-connected cycles, star graphs, wrapped butterflies, chordal rings, multidimensional meshes and tori of restricted diameter.

In [40], robots have to solve the gathering problem in a graph that may contain one or more black holes. The agents are asynchronous and anonymous, and have knowledge of a bound on the size of the network. A characterisation of the situations where the problem is solvable is presented, and upper and lower bounds are offered for those situations.

[54] studies the black hole problem on directed graphs, where the ping-pong technique is not possible. It is shown that for asynchronous agents, at least 2^{Δ} agents are required to identify the black hole, with Δ the maximum in-degree of the black hole. Is is also shown that this lower bound applies to synchronous agents. In a planar graph with a planar embedding known to the agents, a lower bound of 2Δ and an upper bound of $2\Delta + 1$ are provided.

In [117], asynchronous labelled agents starting at different locations are trying to construct the map of a graph with multiple black holes. At each node, there is a whiteboard. An algorithm requiring $O(n_s m)$ moves is provided, with m the number of edges and n_s the number of safe nodes.

The number of pebbles required to identify the black hole is studied in [13]. A group of asynchronous anonymous agents are trying to find the black hole in an unknown graph with knowledge on an upper bound on the size of the graph and the number of edges. For 3 tokens, at least $\Delta+2$ agents are required, with Δ the maximum degree of a node, and an algorithm solving the black hole problem is provided. If the number of agents is unknown, five tokens become necessary.

2.5.3 Faulty Robots

Time-Varying Graphs and black holes problems discuss potential variation in the fixed part of the environment: the graph of geometric environment in which the robots evolve. Problems involving faulty robots introduce the possibility of variations in the mobile part of the environment: the robots themselves. Those variation imply unreliability, which may significantly increase the amount of resources required for an algorithm to solve a problem.

The notion of Byzantine faults, that became a central topic in exploration problems with unreliable robots, was first introduced in [168]. The paper introduces the problem as follows: an army captain must make a decision based on the information he receives from his generals. However, some of those generals may be liars attempting to make him take the wrong decision. In order for all loyal generals to reach an agreement, there must be more than two third of all generals that are loyal. The notion of Byzantine general has later been applied to the exploration problem in the context of Byzantine faults: a robot may behave in such way as to distribute false information and hinder the accomplishment of a given algorithm. Similarly, asynchronous systems with Byzantine processes are studied in [111], and a weaker version of the problem is discussed in [167].

The study of faulty robots has two principal subdivisions: the crash faults, and the Byzantine faults. In the crash fault model, the robot may be unable to move, communicate or sense its environment properly, but will not willingly hinder the execution of the algorithm for other robots. In the Byzantine scenario, based on [168], [111], and [186], the robot will actively attempt to hinder the other robots by either changing its path, communicating false information or ignoring relevant information.

Most papers studying robots operating in LCM cycles assume that during their Look phase, robots are able to collect perfect data from the environment, and during their Move phase, they exactly move at the expected location. [48] drops those assumptions, and considers robots that may be experience inaccuracy in their readings or in their movements. Several impossibility theorems are introduced. It is shown that robots are unable to gather in a finite number of steps. An algorithm allowing convergence is presented, assuming bounded measurement, movement and calculation errors.

Similarly, [2] considers the gathering problem for robots operating in LCM cycles, and prove that all previously known algorithms fail in the presence of either crash faults or Byzantine faults. More so, a gathering of 3 robots, one of which is Byzantine, is impossible in the asynchronous setting. In the synchronous setting, an algorithm solves the gathering problem for $k \geq 3$ robots with at most one faulty robot. A general algorithm also solves the problem if there are at least $3f + 1$ good robots, with f the number of faulty robots.

Gravitational algorithms for robots operating in LCM cycles are studied in [47]. The paper focuses on the asynchronous setting and shows correctness of the gravitational algorithm to solve the gathering problem. Analysis of its convergence rate and resilience to crash faults is also discussed.

Gathering of labelled synchronous agents in the graph, f of which are byzantine, is studied in [91]. Two levels of Byzantine behaviours are studied: a strong Byzantine robot can choose its port when moving and convey arbitrary information to other agents, whereas a weak Byzantine robot can do the same, but is unable to lie about its label. For weak Byzantine robots, if the size of the graph is known, any number of good agents can gather. If the size is unknown, at least $f + 2$ agents must be good agents for the gathering problem to be solvable. This bound is tight. For strong

Byzantine agents, a lower bound of $f+1$ good agents is provided, even when the graph is known. Algorithms that ensures gathering are provided and require $2f+1$ good agents if the size of the graph is known, and $4f+2$ good agents if the size of the graph is unknown. An open question left in this article is solved in [30], where it is determined that the minimum number of good agents required to guarantee deterministic gathering of all good agents, with termination, is $f+2$.

In [199], a swarm of k labelled robots must solve the gathering problem while being resilient to faults in their sensors and Byzantine agents. For small swarms, the gathering problem is solved and is resilient to any number of Byzantine agents. For larger swarms, an algorithm solving gathering is provided with the assumption that the number of Byzantine agents around a good agent is bounded.

A probabilistic point of view of the gathering problem is introduced in [79]. The paper considers a group of disoriented robots operating in LCM cycles. Deterministic algorithms are studied under the additional assumption that robots may experience crash faults or Byzantine faults. A large set of scheduling strategies are considered and lower bounds are provided.

In [31], a swarm of oblivious robots operating in LCM cycles on the line must solve the convergence problem (that is, be located at a distance that is no more that ϵ apart) despite Byzantine faults. An algorithm allowing gathering for at least $2f+1$ good robots is provided for the synchronous setting, and an algorithm allowing gathering for at least $3f+1$ good robots is provided for the asynchronous setting, with f the number of Byzantine robots.

Exploration on the line by a group of n robots, f among which are faulty, is studied in [67]. This paper focuses on crash-faults. In other words, robots may fail to see the exit, but will not communicate false information. For $n \geq 2f+2$, an algorithm with a competitive ratio of 1 is provided. For larger f, an algorithm called *proportional*

schedule algorithm is provided, and is proven to be optimal for $n = f + 1$.

A variation of the patrolling problem discussed later in this survey is discussed in [58]. The paper considers the patrolling problem by a group of k robots, f of which are crash-faulty. The environment is a weighted graph, and the purpose of the algorithm is to minimise the visit time by a good robot of all nodes in the graph. An optimal algorithm is provided for a line segment and for Eulerian graph. For cubic graph, the problem is shown to be NP-Hard by reduction from the 3-colouring problem.

[201] presents a survey on fault-tolerance and the similar problem of fault detection.

An interesting take on fault tolerance is presented in [130], where k robots prone to faults must explore an infinite sequence of boxes in order to find a treasure. The authors propose non-coordinating algorithms and discuss contexts in which it may be favourable to implement non-coordinating algorithms rather than the faster coordinating algorithms. Building upon those results, [158] study a variation where the treasure is placed uniformly at random in a finite, large number of boxes, and propose algorithms that achieve optimal speed-up.

Resilience to sensor faults are discussed in [187], where two robots observing each other and the environment and sharing this information can reduce odometry errors and better detect obstacles, thus increasing the quality of the map they create. Algorithms are introduced and supported by experimental results and simulations.

In [37], the exploration of a network with faulty edges by a single robot is considered. A perfectly competitive algorithm is provided for ring environment, and for networks modelled by Hamiltonian graphs, it is shown that the overhead (the worst-case ratio between the cost of a given algorithm and the cost of an optimal algorithm which knows where the faults are located) for a Depth-First Search is at most 10/9 times larger than that of a perfectly competitive algorithm.

The notion of robots controlled by a market economy is used in [209] to solve the exploration problem. Using this approach, the authors allow for dynamic inclusion and departure of robots during the execution of the algorithm. The algorithm they provide is resilient to communication faults.

2.6 Other Variations Involving Search by Multiple Robots

Aside from the standard exploration problem and its popular variation, the rendezvous problem, many other areas of distributed computing involve autonomous mobile robots. Similar problems may include the evacuation problem, where robots first have to identify an exit (which is very similar to the exploration problem), and must then gather at the location of this exit; the deployment problem, where robots must form a specific pattern rather than gather at the same point; patrolling, where robots must periodically explore all sections of the environment; in some problems, robots are heterogeneous and may have different speeds; in some others, robots have two different speeds: one for travelling and one for searching, that is always slower than its travelling speed. The bouncing robots problem considers robots that are unable to control their movements, and exchange their speed when they meet one another. Those variations are presented here because they are closely related to the exploration problem and have significant commonalities.

2.6.1 Evacuation or Group Search by Collections of Mobile Robots

In the evacuation problem, a group of robots must first identify an exit located in the environment, which is a problem extremely similar to the group exploration. The additional difficulty of the evacuation problem resides in the fact that robots must then communicate this information to each other, then gather at the location of the exit before the problem is considered resolved.

In [61], k identical robots are placed inside a disk of unitary radius. The exit is located somewhere on the boundary of the disk, and the robots start the execution of the algorithm at the center of the disk. Two communication models are considered: in the local communication model, robots can only exchange information when they meet each other. In the wireless communication model, the information is exchanged freely and instantly between all robots. The goal of the algorithm is to gather all robots at the location of the exit. Lower bounds on the required time for both communication models are provided for $k = 2$ and $k = 3$. Almost-tight bounds are provided for large k. More precisely, an algorithm that ensures completion in $3+\frac{2\pi}{k}$ is provided for the local communication model, and a lower bound of $3+\frac{2\pi}{k}-O(k^{-2})$ is required. In the wireless communication model, an algorithm allows the evacuation in $3+\frac{\pi}{k}+O(k^{-4/3})$ is provided, and at least $3+\frac{\pi}{k}$ is required.

Evacuation from the line is considered in [15], where two robots with distinct maximal speed initially placed at the same point on the infinite line must locate an exit and gather at its location. Two forms of communication are considered: local communication and wireless communication. For all possible maximal speeds in the local communication model, optimal algorithms are provided. For the wireless model, an optimal algorithm is provided when the fastest robot is at most 6 times faster than

the slowest robot.

Evacuation from the line by a group of k identical robots is considered in [44]. The surprising result of this paper is that the number of robots used to complete the evacuation problem has no impact on the time required to complete the evacuation, that remains $9d - o(d)$, with d the distance from the initial location of the robots to the exit. It is furthermore shown that the bound of $9d$ can be achieved with one robot moving at unit speed an a second robot moving at a speed no slower than $1/3$.

In [61], two identical robots located at the center of a disk must evacuate it through an exit located on its boundary. The local communication model is considered: robots can only exchange information when they are located at the exact same location. The paper improves on previously existing results by providing an algorithm that forces a meeting between the robots even if the exit has not been found by either robots.

The same problem is considered under the wireless communication setting in [169]. An additional difference is that the two robots may now have different speeds. An optimal algorithm is provided if the fastest robot is approx. 2.75 times faster than the slowest robot or more. For closer speeds, upper and lower bounds are provided.

In [53], two robots are placed on a circle that contains k exits. Both robots have a map of the circle, but are not aware of their starting location. The purpose of an algorithm is to evacuate both robots as fast as possible. Robots use the wireless communication model. Two variations are studied: in the first one, robots control the distance that separate them at the beginning of the problem; in the second one, they do not. Upper and lower bounds are provided for some subsets of the problem.

2.6.2 Patrolling

The patrolling problem is similar to the exploration problem in the sense that the map must be fully explored. In the patrolling problem, however, this is not sufficient: the map must be fully explored on a periodic basis, and the efficiency of any given algorithm is measured by the maximal period of time elapsed between two visitations of the same point, called *idle time*.

[107] considers the problem of patrolling a closed polygon (ring) by identical robots, and introduce the cycling strategy, where robots always patrol in the same direction, and the partition strategy, where robots choose a segment of the environment and go back-and-forth on this segment. Efficiency of those algorithms are evaluated for different values of k the number of robots and different visibility ranges for the robots.

In [57], both the decidability and optimisation problem are considered. The decidability problem asks whether it is possible for a group of k agents to maintain an idle time lower than a certain value τ. The optimisation problem asks what is the minimum possible idle time for a given environment and a set of k agents. In this paper, agents have distinct maximal speeds, and the environment is either an open curve (a line segment) or a closed curve (a ring). Various strategies, including the cycling strategy, the partition strategy, and some newly described strategies, are evaluated. The bounds established in this paper are later improved in [45] and [150].

The problem is applied to robots with distinct visibility range in [70], both for robots with equal speeds and robots with different speeds. An optimal algorithm is provided both for the closed and open environment for robots with the same speed. It is shown that the case of robots with different speeds is fundamentally different from the case of robots with identical speeds. An optimal algorithm is given for the case

of two robots with both distinct speeds and distinct visibility ranges. It is also shown that the patrolling of general graphs with different visibility ranges is NP-Hard.

In [49], a different environment in considered: this environment includes vital segments, that must be patrolled, and neutral segments that can be traversed by robots as part of their patrolling, but it is not mandatory for the robots to patrol those segments. It is proven that either the cycling strategy or patrolling strategy can yield optimal idle time for identical robots.

[137] considers the patrolling of a graph by a single robot with constant memory. The robots must visit every node periodically, but not necessarily every edge. The efficiency of the algorithm is measured by the number of edges traversed before all points are revisited. The paper presents an algorithm arranging port numbers in such way that the robot can revisit all nodes in time $3.75n - 2$ for n the number of nodes in the graph.

In [14], the environment is a partial grid (a finite grid with possible missing vertices or edges). A group of k synchronous identical robots must patrol the environment in such way that as many robots as possible visit all nodes in the graph on a periodical basis, without there ever be two robots on the same edge or node at the same time. Each robot has a visibility radius of ρ. The paper shows that for $\rho = 0$, the problem is unsolvable, and at least $\rho = 1$ is required. For $\rho = \infty$, it is shown that at least $k \leq p - q$ robots can visit all nodes on a periodic basis, with p the number of vertices and q a parameter whose value depends on the topology of the environment.

2.6.3 Pursuit

The pursuit problem is a variation of the collaborative exploration problem. In the pursuit problem, sometimes referred to as the *Cops and Robbers problem*, the target

that must be found it mobile and actively tries to avoid being captured, as opposed to the collaborative exploration problem, where the target is static, or the rendezvous problem, where the mobile agents collaborate to ensure a rendezvous. A taxonomy of various pursuit problems is discussed in [45] and [125], with discussion about various results. Generally, the measure of efficiency of an algorithm is the capture time of the environment. In other words, it is the amount of time required to capture the target in the environment by a group of robots using the given algorithm. A graph is called *cop-win* if the capture is possible, and *robbers-win* if the capture of the target is impossible.

In [27], the environment studied is either the finite or countably infinite graph. The notion of capture time is explored in relation to the number of vertices of the graph and special properties of the graph. The notion of capture time density is applied to infinite graphs. It is proven that the problem: "can k cops capture a robber in no more than t moves?" is NP-complete.

[163] introduces the notion of *monotonicity* for pursuit problem. A game is considered monotone if it is possible for the cops to catch the robber before the robber reaches a place that has been previously explored by the cops. The paper shows that two type of games are non-monotone: the game on directed graphs where the robber is invisible and lazy, and the game on directed graphs where the robber is visible and fast.

[133] studies scenarios where the robber is fast i.e. it can move $R > 1$ edges at a time. A general upper bound is presented. For finite R, on a graph with n nodes, it is shown that at least $n^{1-\frac{1}{R-2}}$ cops can be necessary to catch a robber. For infinite R, the number of required cops is linear in n. For $R = 1$ on directed graphs, an algorithm requiring $O(n(\log\log n)^2/\log n)$ is provided.

It is shown in [124] that the minimum number of cops required to catch a robber

on a given graph is NP-Hard. On split graphs, the problem requires a polynomial amount of time if the robber is as fast as the cops, and is NP-Hard if the robber is twice as fast as the cops. On graphs of bounded clique width, the problem only has a polynomial solution if the robber is at most twice as fast as the cops. On planar graphs, there is no bound on the minimum number of cops required to catch a robber that is faster than the cops.

In [26], a variation of the cops and robbers game called the *distance k cops and robbers* is presented. In this variation, cops (with c the number of cops) win if they can get at a distance of at most k of the robber. An algorithm is given for the decidability problem ("given k and c, can the robber be caught?"), and it is proven that the optimisation problem ("what is the minimum number of cops required to be at a distance at most k of the robber on the graph?") is NP-Hard.

[175] and [29] study the cops and robbers game on random graphs, based on the fact that the required number of cops to catch the robber expressed as a function of the average node degree of a graph forms a zigzag shape.

[6] considers a similar problem, where a single searcher is trying to catch a single hider. The searcher can search a unit area in unit time, or can enter an "ambush mode". The searcher wins if it the hider is inside the searched area, or if the hider moves while the searcher is in "ambush mode". The efficiency of a given algorithm is measured by the amount of time required before the hider is caught.

A variation of the cops and robbers game called *Graph-Clear* is presented in [157], where many robbers have to be caught. The cops have access to two different actions: the sweep actions, that allows them to catch a robber, and the block action, that prevent a robber to use an edge of the graph. The goal is still to capture all robbers using as few robots as possible. It is proven that the general case of the Graph-Clear game is NP-Hard.

2.6.4 Other Variations

Tether / Fuel: The exploration problem by a single robot on a graph in the presence of a tether or with fuel is studied in [97]. The paper considers two scenarios: in the first and more restraining one, a robot is linked to its starting position by a tether of length l, and can therefore go no further than l away from its starting node. In the second scenario, a robot has a limited fuel tank of capacity C, and must return to its starting point after traversing C edges. For both scenarios, an algorithm solving the exploration in $\Theta(|E|)$ is provided.

Freeze-Tag: [8] introduces the *Freeze-Tag Problem* (FTP). In this problem, a set of robots starting at different locations are inactive. At the start of the algorithm, a single robot is active, and is able to awaken inactive robots, rendering them active as well. The purpose of the algorithm is to awaken all robots as early as possible. An active robot awakens another simply by moving to its location. On graphs, this problem is proven to be NP-Hard, even for star graphs.

It is observed in [9] that any algorithm that is not purposefully unproductive yields an $O(\log n)$ approximation of the optimal solution. The paper also provides an $O(1)$-approximation algorithm for unweighted graphs with one robot at each node, and explore the scenario where there is more than one robot at each node.

Colliding Robots: The problem of bouncing or colliding robots is set in a continuous ring of unitary circumference. A group of robots with different initial positions move in a fixed direction (in other words, they have positive or negative speeds). Whenever two robots meet, they exchange their speeds, thus "bouncing off" one another. Robots may not change their speed on their own volition, and can only gain information by colliding with other robots. Robots have an internal clock that informs them of the time of each of their collision, and have unlimited memory. In [59],

all robots have the same constant speed. The paper characterises all initial configurations that allow all robots to be informed of the initial position of all other robots, and further considers the same problem on the line segment. [69] extends the problem on the ring by allowing robots to have different initial speeds. Whenever two robots collide, their speeds are exchanged. The initial positions and velocities that allow for all robots to obtain knowledge of the initial position and velocity of all other robots are discussed. It is proven in this paper that the configuration is feasible if and only if no robot has an initial velocity that is equal to the average of the velocities of all robots.

Beachcomber: In the beachcombers problem, each robot has two different speeds: one that the robot uses to travel without searching the environment, and a slower one where the robot performs a search of the environment. The goal of a beachcomber algorithm is for a group of heterogeneous robots to fully search the environment. In [55], a group of k heterogeneous robots with different walking and searching speeds must explore a line segment. All robots start at one end of the line segment, and have full knowledge of the number of robots in the group, and all of their respective speeds. In the offline scenario, robots know the length of the line segment in advance, and in the online scenario, robots do not have this information. An optimal algorithm is given for the offline scenario, and a 2-competitive algorithm is provided for the online scenario. [17] extends the environment to the cycle, and prove that the capability for a robot to change its direction does not help robots in their search.

2.7 Practical Results

An algorithm called the *Mapping algorithm* is presented in [190], and uses the notion of hill-climbing to create maps that are maximally consistent with sensor data and

odometry. The algorithm tries to minimise the redundancy between the information gathered by the group of robots, thus maximising overall utility. Real-world trials and simulations support the claims. A similar technique is used in [34] to assign target points to mobile robots while taking into account both the utility of the point and the cost of reaching it.

The difference in effectiveness between random and coordinated search algorithms by a group of mobile robots is discussed in [134] under various criteria, such as cost effectiveness, interplay of sensor cost and uniform search coverage.

Unmanned aerial vehicles (UAVs) are used in [146] to search an area with targets of several types (suspected, unknown). At each location, a task to be performed may require several coordinated UAVs. Not all UAVs have the same capabilities: some have better searching capabilities, while other are better at taking care of tasks once the target has been found. Algorithms that use dynamic modelling are used to solve those tasks efficiently, and simulations support the claims.

[203] studies the amount of energy required to ensure the motion of autonomous mobile agents. A complete energy model is presented to describe the energy consumption during the movements, and an optimal velocity schedule is proposed to minimise the energy expense on uniform roads. Near optimal velocity schedule is proposed for variable road conditions. Simulation results are provided.

A survey of various mapping strategies using a single or multiple mobile agents in both theoretical and real-world environments is presented in [197].

[35] and [36] consider a variation of the exploration problem in an unknown environment where the robots needs to reach a certain *target point* before using their scanners. Target points have various values based on their location (for example, a robot will prefer a point that allows it to see around a corner), and those values are decreased as other robots collaborate to the mapping of the same area of the envi-

ronment. This paper claims to be the first to consider both the cost to reach a point and the utility of the point as part of the provided algorithm, and provides real-world experiments to show that the algorithm significantly reduces exploration time.

Chapter 3

Searching the Plane with Faulty Robots

3.1 Introduction

Searching for a target is a common task in several domains of human activity and has been modelled as such in mathematics, theoretical computer science, and robotics in particular. It has been studied for graphs and various geometric domains when the target is either mobile or stationary. The overall goal is to minimize the time required by the searcher(s) to find the target. The robots may co-operate by exchanging messages and using one of the following two models: *wireless*, in which instantaneous communication between the robots is possible at any distance, and *face-to-face* (F2F), which requires for the robots to be at the same location at the same time. In this chapter we design fault-tolerant search algorithms for robots, some of which may have either crash- or byzantine-faults, and show they are optimal despite the fact that the robots can move with possibly distinct maximal speeds.

3.1.1 Preliminaries and notation

In this subsection we introduce the basic locomotive and communication models as well as the behaviour of the robots and discuss all necessary assumptions and notation that will be used throughout the chapter.

There are k robots labelled $r_1, r_2, \ldots, r_k$. They start at the same point, considered to be the origin O in the plane and are searching for a target assumed to be at an unknown distance d from the origin. The robots are equipped with a compass and share a common coordinate system. They furthermore evaluate distance in the same manner, thus having the same notion of unit length. Robots may have different speeds. We say that robot r_i has a maximal speed v_i, for $i = 1, 2, \ldots, k$, and we can assume without loss of generality that for every $1 \leq i \leq k-1$, $v_i \leq v_{i+1}$. A robot can choose to move with any speed that does not exceed its maximal speed or even choose to remain completely still. Each robot has visibility range 1 and is able to identify the target when it belongs to its visibility range.

In the general setting being studied here, k robots are searching for a target located at an unknown location in the two-dimensional Euclidean plane and at a priori unknown distance d from the starting location of the robots; during their search the robots may inform each other of the location of the target, and gather there. The design and analysis of the search algorithms takes into account the communication models being used by the robots and the behaviour of the robots during the search which are defined in the sequel.

Two communication models are being considered. In the wireless communication model, robots can exchange information instantly, regardless of their distance in the Euclidean plane. In the Face-to-Face communication model, robots can exchange information instantly provided they are located at exactly the same point

in the Euclidean plane. We note that some models (e.g. the one described in [42]) use the notion of wireless communication collisions. In the wireless communication collisions model, difficulties may arise if a robot receives multiple wireless messages simultaneously. Our model does not consider the possibility of such collisions.

To make our presentation more intuitive we first consider and analyse search algorithms for *Non-Faulty* (NF) robots, in which the robots follow the algorithm as intended, without any violations of their protocol(s). We then analyse two scenarios concerning the behaviour of the robots: *crash faults* and *Byzantine faults*. The choice of the faulty robots is made by the adversary, which knows our algorithm in advance and attempts to maximize its worst-case search time. In both cases up to f among the k robots may be faulty, where $0 \leq f \leq k-1$. In the case of *Crash-Faults* (CF), the faulty robots behave like reliable robots, but may be subject to various kinds of faults in that they may not be able to receive information from other robots, they may be unable to share the information they know with other robots, they might be unable to move, or to identify the target if it is within their visibility range. They will always, however, follow the instructions given by the algorithm at the best of their capabilities. They will never, for example, communicate false information, or err from the path they were assigned.

Byzantine robots behave like robots with crash-faults, with the difference that an adversary may have them move as he sees fit; share and/or propagate true or false information, and relay or refuse to relay information. A byzantine robot may not lie about its own label. In both scenarios it is required for the algorithm to finish so that every reliable robot gathers at the location of the target.

The overall goal is to solve the *group search* problem which is to minimize the time required for the last non-faulty robot to reach the target, asymptotically as the distance d tends to infinity. Our methods also solve the classical *search* problem which

requires that only the first non-faulty robot reaches the target.

3.1.2 Related work

Search has been studied extensively in mathematics, computer science, robotics, and operations research and is generally concerned with minimizing the time to find a hidden target under various conditions on the terrain being searched, and capabilities of the searcher(s). When the terrain is unknown to the robots (in advance) then the search must imply exploration [4, 82, 140]) and often involves mapping of the environment [154, 180]. For useful surveys on search algorithms we refer the reader to [23, 125] as well as to the book [7] which studies search games. Several papers investigated search in geometric environments, (e.g. [82, 140, 154]) or, similar to our work, the two-dimensional plane, [11, 12, 103, 106, 110].

Exploring or searching the plane by a team of robots involves coordination and it is one of the main themes of investigation not only in computational geometry [12, 174], but also in robotics research [196, 206] and in distributed computing [103, 106, 110]. However, collaborative exploration may be hard even for simple environments, which are not known in advance (e.g. see [126]). In the case, similar to one studied in the present chapter, when the search is completed by the arrival of the last robot, it happens that having more robots does not help (e.g. [44]) or that achieving optimal search time is non-trivial (e.g. disc evacuation in [56]).

Fault tolerance in distributed computing problems has been extensively studied in the past (see, e.g., [142, 168, 176]). However, the question of reliability has been mostly investigated in cases where failures arise from the static elements of the environment (network nodes and links) rather than from its mobile components (robots). However such malfunctions are sometimes modelled by dynamic alteration of the

network (e.g., [38, 164]). Failures due to mobile robots were investigated in the context of the problems of gathering [2, 91, 191], convergence [47], flocking [207], linear search [67], plane search by ANTS represented by finite automata [192], patrolling [58], etc. Some papers investigated the case of unreliable or inaccurate robot sensing devices such as in [48, 145, 191].

The collection of robots that may have distinct speeds were used in [18] to design fast converging protocols, e.g. for gathering. [203] used varying mobile sensor speed to achieve sensor energy efficiency. However, as in the case of [57, 150], which considered distinct speeds for robots performing boundary patrolling, to achieve optimal algorithms for larger collection of robots turns out to be quite difficult. [44] investigated a pair of distinct-speed robots searching a line, when the search time is determined by the arrival of the last robot. Again, besides some interesting limited cases, a general optimal algorithm has not been proposed.

3.1.3 Outline and results

We propose and analyze search algorithms for robots some of which may be faulty in the wireless and face-to-face communication models. In Section 3.2, we initiate our investigations by discussing the special case of non-faulty robots.

We introduce an algorithm for the wireless model, and one for the F2F model, both of which are asymptotically optimal.

In Section 3.3, we introduce the problem of robots which may be subject to crash faults. For both the wireless and the F2F communication model, we introduce an algorithm that is asymptotically optimal. Finally, in Section 3.4, we introduce algorithms for robots with byzantine faults. In the wireless case, we present an algorithm that is asymptotically optimal.

For the F2F model, we present two algorithms. The first one is asymptotically 2-competitive for $k \geq 2f + 1$, while the second requires that $k \geq 2f + 2$, but it is asymptotically optimal when the robots have identical speeds.

3.2 Non-Faulty (NF) Robots

In this section, we introduce an algorithm for non-faulty robots, prove its correctness and analyze its complexity, both for the wireless and F2F setting.

3.2.1 Wireless communication

The algorithm is divided into four phases, Initialisation, Exploration, Communication, and Gathering. In the sequel we describe them in more detail.

Initialisation phase. All robots start at the same location, considered to be the origin O of the plane. The robots divide the plane into angular sectors originating at O and whose angles are proportional to robot speeds. The angle α_i of the sector covered by r_i is given by the formula: $\alpha_i = \frac{2\pi v_i}{\sum_{j=1}^{k} v_j}$, for $i = 1, 2, ..., k$. Robots first move at a distance 2 from the origin, taking position on the edge of the sector. If i is odd, the robot goes to the left edge of its sector, and if i is even, it goes to the right edge of its sector.

Exploration phase. Each robot searches its own sector in a zig-zag pattern, as shown in Figure 3.1. We call a *step*, the part of the trajectory of a robot composed of an arc covering the angle allocated to the robot followed by a straight line segment of length 2. When the robots are moving along the arc, they move at their maximal speed v_i, and when they are moving in straight line, they move at the speed v_1, i.e., the speed of the slowest robot. This ensures that all robots begin their next step at the same time. A robot stays in this phase until it is informed of the location of the target,

either because it found it itself, or because a neighbour robot shared this information. Once the location of the target is known, the robot first finishes its current step, then enters the communication phase.

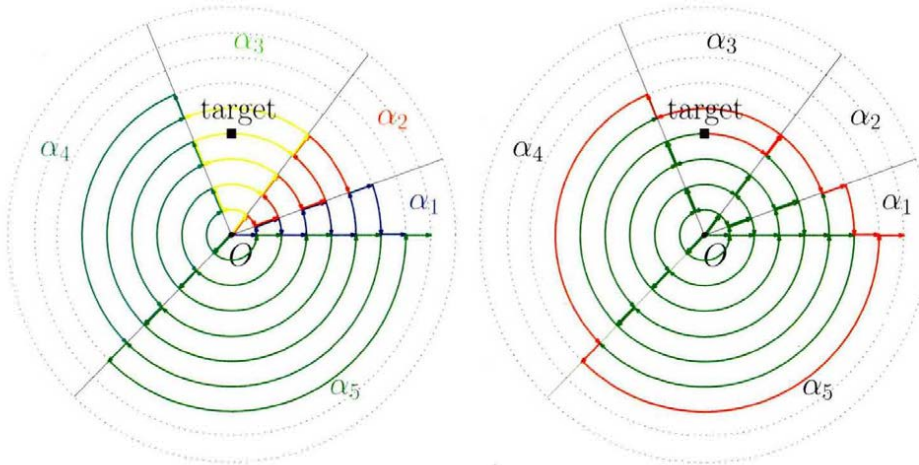

Figure 3.1: The execution of the non-faulty robots with face-to-face communication algorithm, where r_3 identifies the target, with the exception of the gathering phase. The figure in the left describes the trajectories of the robots (one color per robot) while the right figure describe the propagation of the information of the location of the target (green for uninformed robots and red for informed robot).

Communication phase. In the wireless communication model, communication is instantaneous, regardless of distance. As soon as a robot identifies the target, it informs all other robots instantly, then executes its gathering phase.

Gathering phase. A robot in the gathering phase moves towards the target in a straight line, using its maximal speed.

3.2.2 Face-to-face communication

The face-to-face communication variation of this scenario has the same initialisation phase, exploration phase and gathering phase. However, as the communication can only be done when the robots meet, the algorithm must be adapted to ensure that all

robots meet, and therefore receive the information about the location of the target. An execution of the face-to-face variation of this scenario can be observed in Figure 3.1. Note that the gathering phase is not depicted in order to make the figure clearer. The communication phase now unfolds as follows:

Communication phase. Once a robot is aware of the location of the target, it will share this information both with the robot on its right and with the robot on its left before starting its gathering phase. Any given robot will meet either its left or right neighbour at the beginning of a step, and the other neighbour at the end of the current step, as shown in Figure 3.1. Therefore, the communication phase of a robot consists of the execution of a step as described above, at the end of which the robot will execute its gathering phase.

There is a special case to consider: if k is odd, then r_1 and r_k will have synchronous patterns, and will not meet at the end of every two steps. Their communication phase will therefore unfold as follows:

1. The robot first informs the neighbour it meets normally (that is, r_1 informs r_2, and r_k informs r_{k-1}).

2. The robot completes another step as defined in the *exploration phase.*

3. The robot then stays idle for the part of the step where it moves along an arc, and simply executes the part of the step where it moves in a straight line. In doing so, it will meet the other robot at the end of the step.

4. The robot shares the information, then proceeds to its gathering phase.

Algorithm 1 describes the behaviour of the robots in the face-to-face communication variation. The procedure **MoveLine**(*dist*, *dir*, v) moves the robot in direction

dir of distance *dist* at speed v and procedure **MoveArc**(c, α, r, v) moves the robot at speed v along the circular arc centered in c of radius r subtending the angle α.

Algorithm 1: NonFaultyRobots

```
1  Initialisation phase
2  V ← Σ_{i=1}^{k} v_i;
3  dir ← 0;
4  for i = 1 to k do
5  |  α_i ← 2πv_i / V;
6  |  if i mod 2 = 0 then
7  |  |  dir_i ← dir; β_i ← 1;
8  |  else
9  |  |  dir_i ← dir + α_i; β_i ← −1;
10 |  dir ← dir + α_i;
11 Exploration phase (every robot r_i executes this in parallel)
12 j ← 1;
13 while Target location unknown do
14 |  MoveLine(2, dir_i, v_1);
15 |  MoveArc(O, α_iβ, 2j, v_i);
16 |  dir_i ← dir_i + α_iβ;
17 |  β ← −β;
18 |  j ← j + 1;
19 Communication phase (every robot r_i executes this in parallel)
20 MoveArc(O, α_iβ, 2j, v_i);
21 dir_i = dir_i + α_iβ;
22 MoveLine( 2, dir_i, v_1);
23 j ← j + 1;
24 if (i = 1 or i = k) and k mod 2 = 1 then
25 |  stay still for time 4jπ/V ;
26 |  MoveLine(2, dir_i, v_1);
27 Gathering phase (every robot r_i executes this in parallel)
28 Go to target
```

Lemma 1. *Algorithm 1 is correct, i.e., the target is always found and the robots eventually gather at the location of the target, for both variations of the algorithm (wireless communication and face-to-face communication).*

Proof. In order to prove the correctness of the algorithm, we have to show first, that

the algorithm always finds the target, and second, that all the robots can gather at the target.

To prove the first part, we note that at the j-th step of the exploration phase, the robots will scan an annulus where the inner circle has a radius of $2j-2$, and the outer circle has a radius of $2j$. It is obvious that the robots cover the entirety of the annulus. Therefore, after j steps, a disc of radius $2j$ will have been entirely scanned by all the robots combined.

Next we prove that the robots will always gather at the target. For the wireless variation, communication is instantaneous, and the robots are therefore informed immediately. They can then gather at the location of the target. We prove that the robots are all informed in the face-to-face communication variation as follows. As the robots begin each step simultaneously, they are guaranteed to meet either their left or right neighbour at the end of each step. More precisely, all odd robots will meet their left neighbour at the end of every even step, and their right neighbour at the end of every odd step. The opposite is true for the even robots. The only exception to this rule is the first and last robots, if there is an odd number of robots, in which case the information is spread at the cost of at most two extra steps. By following the algorithm, this ensures that every robot is informed of the location of the target eventually, and can therefore gather at the target, which completes the proof of Lemma 1. □

Lemma 2. *The search performed by Algorithm 1 is completed in time*

$$\frac{4\lfloor d/2 \rfloor + d}{v_1} + \frac{2\pi \lfloor d/2 \rfloor \left(\lfloor d/2 \rfloor + 1\right)}{V}$$

for the wireless communication variation, and in time

$$\frac{4\lfloor d/2\rfloor + d + 4\lceil k/2\rceil}{v_1} + \frac{2\pi(\lfloor d/2\rfloor + \lceil k/2\rceil)\,(\lfloor d/2\rfloor + \lceil k/2\rceil + 1)}{V}$$

for the face-to-face communication variation.

Proof. First, we show the following claim.

Claim 1. *The end of step j occurs at time: $\frac{2j}{v_1} + \frac{2\pi j(j+1)}{V}$.*

The trajectory of a robot until step j consists of j arcs of growing radius (line 15) as well as j segments of length 2 resulting from the linear moves from lines 14. As robot r_i uses its full speed v_i for circular moves and speed v_1 for straight line moves, this takes time

$$\frac{2j}{v_1} + \frac{\alpha_i(2 + 4 + \cdots + 2j)}{v_i},$$

which is equal to:

$$\frac{2j}{v_1} + \frac{2\pi j(j+1)}{V}.$$

This ends the proof of the claim.

Suppose that the target is found at a point situated at distance d (w.l.o.g. d is an integer) from the origin O, say by robot r_i. We will show that this happens in step $\lfloor d/2\rfloor$ of the exploration phase (the while loop from line 13). For simplicity, we assume that the target may only be found during the circular movement of the robots. Indeed, the union of the robots' positions during circular movements cover the entire plane. Hence by Claim 1, the target is found before time

$$\frac{2\lfloor d/2\rfloor}{v_1} + \frac{2\pi\lfloor d/2\rfloor\,(\lfloor d/2\rfloor + 1)}{V}. \tag{3.1}$$

Once the location of the target has been identified by a robot, the information

needs to be shared. Wireless communication allows this to happen instantly; face-to-face communication requires more time. Apart from robots r_1 and r_k in the case of an odd number of robots, the robots will communicate the information to their neighbour at the end of each step, both clockwise and counter-clockwise. This takes at most $\lfloor k/2 \rfloor$ extra steps if the information does not have to spread from r_1 to r_k, or the other way around and k is odd. If there is an odd number of robots, Algorithm 1 will add one extra step in the worst case (for the information to spread from r_1 to r_k, or the other way around), for a total of $\lceil k/2 \rceil$, bringing the total time by Claim 1 (including the exploration phase) to

$$\frac{2(\lfloor d/2 \rfloor + \lceil k/2 \rceil)}{v_1} + \frac{2\pi(\lfloor d/2 \rfloor + \lceil k/2 \rceil)(\lfloor d/2 \rfloor + \lceil k/2 \rceil + 1)}{V}. \tag{3.2}$$

Finally, for both variations, the last robot informed (say, r_i) will have to go back to the location of the target. By the time r_i is informed, in the wireless case, it will be at a distance $2\lfloor d/2 \rfloor$ from the origin, and in the F2F case, it will be at a distance $2(\lfloor d/2 \rfloor + \lceil k/2 \rceil)$ from the origin O, that we obtain by considering only the linear movements of the robot. Supposing that the target is located at a point directly opposed to r_i, then r_i will have to cover a distance of $2\lfloor d/2 \rfloor + d$ in the wireless case, and $2\lfloor d/2 \rfloor + d + 2\lceil k/2 \rceil$ in the F2F case, which it will do at its maximal speed. Therefore, the worst gathering time for the wireless case is:

$$\frac{2\lfloor d/2 \rfloor + d}{v_1}, \tag{3.3}$$

and the worst gathering time for the F2F model is:

$$\frac{2\lfloor d/2 \rfloor + d + 2\lceil k/2 \rceil}{v_1} \tag{3.4}$$

Altogether, we obtain the total time required by the wireless variation by summing Formulas (3.1) and (3.3); whereas we obtain the total time required by the face-to-face variation by summing Formulas (3.2) and (3.4). We conclude that the total time for the wireless communication variation is

$$(3.1) + (3.3) = \frac{4\lfloor d/2 \rfloor + d}{v_1} + \frac{2\pi \lfloor d/2 \rfloor \left(\lfloor d/2 \rfloor + 1\right)}{V}.$$

The total time for the face-to-face communication variation is:

$$(3.2) + (3.4) = \frac{4\lfloor d/2 \rfloor + d + 4\lceil k/2 \rceil}{v_1} + \frac{2\pi(\lfloor d/2 \rfloor + \lceil k/2 \rceil)\left(\lfloor d/2 \rfloor + \lceil k/2 \rceil + 1\right)}{V}$$

This proves Lemma 2. □

Next we discuss a lower bound on any search algorithm.

Lemma 3. *For any search algorithm with robots starting at the origin, before time* $\frac{\pi(d^2-1)}{2V}$ *there are still some unexplored points of the plane at distance smaller than* d.

Proof. As all robots start at origin, at time 0 an area π of the unit circle is explored. During one unit of time, robot r_i travels a distance of at most v_i and, since it has radius of visibility 1, it can explore a new area of at most $2v_i$. In total, in time t the robots can explore an area of size at most $2tV$. In order to explore all points at distance d, the robots need to explore the new area of size $\pi(d^2 - 1)$. Hence, the time needed to explore all points at distance at most d from the starting point O equals at least $\frac{\pi(d^2-1)}{2V}$. □

To sum up we have proved the following theorem.

Theorem 1 (Non-Faulty Robots). *Algorithm 1 completes the search successfully for k robots in asymptotically optimal worst-case time :*

$$\frac{\pi d^2}{2V} + o(d^2)$$

where d is the distance from the origin O to the target, and V is the sum of speeds of all robots.

3.3 Robots with Crash-Faults (CF)

In this section, we introduce an algorithm for robots with crash faults, prove its correctness and analyse its complexity.

3.3.1 Wireless communication

The algorithm works in a fashion similar to the algorithm for non-faulty robots described in Section 3.2, with the following differences.

Initialisation phase. All robots start at the same location, considered to be the origin O of the plane. The robots divide the plane into angular sectors originating at O and whose angles are computed from robots' speeds as follows:

- Consider each robot, from the slowest to the $(k-f)$-th, and sum their speeds, such that $V' = \sum_{i=1}^{k-f} v_i$.
- For each subsequent robot r_{k-f+j}, for j from 1 to f, compare its speed to $\frac{V'}{j}$. If it is greater than $\frac{V'}{j}$, set the speed of all remaining robots (including r_{k-f+j}) to $\frac{V'}{j}$. Else add v_j to V'.

- For each robot, set the angle α_i of the sector covered by r_i as follows: $\alpha_i = \frac{2\pi v_i(f+1)}{\sum_{j=1}^{k} v_j}$

Notice that the angular sectors of all robots now cover the plane $f+1$ times. Moreover each point of the plane belongs to the angular sectors attributed to at least $f+1$ different robots since by construction $\alpha_i \leq 2\pi$. Robots then move at a distance 2 of the origin, taking position on the edge of the sector. If i is odd, the robot goes to the left edge of its sector, and if i is even, it goes to the right edge of its sector, as it was done in the algorithm for non-faulty robots. The Exploration and Gathering phases are identical to the Algorithm 1, and the Communication phase is instantaneous in the wireless model.

3.3.2 Face-to-face communication

In addition to the modification to the initialisation phase described above, the communication phase is changed. It is now described as follows.

Communication phase. Once a robot identifies the location of the target, it communicates this location to every robot it meets (the exchange of information is still instantaneous). For this purpose after finding the target, it executes $\lceil \frac{2\pi}{\alpha_i} \rceil$ extra steps, moving only counter-clockwise (instead of changing direction at each step). One extra step is necessary to ensure a meeting with a robot moving in the same direction, after an angle of 2π has been covered. A robot that learns about the location of the target without seeing it, immediately starts its gathering phase. Since the target is seen by at least one reliable robot, this ensures that all robots are informed of the location of the target.

Algorithm 2 which can be found below describes the behaviour of the robots in the face-to-face communication variation.

Algorithm 2: CrashRobots

```
1  Initialisation phase
2  V' ← 0;
3  for i from 1 to k − f do
4  |  v'_i ← v_i;
5  |  V' ← V' + v'_i
6  j ← 1;
7  while V'/j ≥ v_{k−f+j} do
8  |  v'_{k−f+j} ← v_{k−f+j};
9  |  V' ← V' + v'_{k−f+j};
10 |  j ← j + 1;
11 for i from k − f + j to k do
12 |  v'_i ← V'/j;
13 for i from k − f + j to k do
14 |  V' ← V' + v'_i
15 dir ← 0;
16 for i from 1 to k do
17 |  α_i ← 2πv'_i(f+1)/V';
18 |  if i mod 2 = 0 then
19 |  |  dir_i ← dir; β_i ← 1;
20 |  else
21 |  |  dir_i ← dir + α_i; β_i ← −1;
22 |  dir ← dir + α_i;
23 Exploration phase (every robot r_i executes this in parallel)
24 j ← 1;
25 while Target location unknown do
26 |  MoveLine(2, dir_i, v'_1);
27 |  MoveArc(O, α_iβ, 2j, v'_i);
28 |  dir_i ← dir_i + α_iβ;
29 |  β ← −β;
30 |  j ← j + 1;
31 Communication phase (every robot r_i executes this in parallel)
32 for i from 1 to ⌈2π/α_i⌉ + 1 do
33 |  MoveLine(2, dir_i, v'_1);
34 |  MoveArc(O, α_i, 2j, v'_i);
35 |  dir_i ← dir_i + α_i;
36 |  j ← j + 1;
37 Gathering phase (every robot r_i executes this in parallel)
38 Go to target
```

Lemma 4. *Algorithm 2 is correct, i.e., the target is always found and the robots eventually gather at the location of the target, both for the wireless and the face-to-face variation.*

Proof. In order to prove the correctness of the algorithm, we have to show first, that the algorithm always finds the target, and second, that all the robots can gather at the target.

To prove the first part, we note that at each step j of the exploration phase, the robots will scan an annulus where the inner circle has a radius of $2j-1$, and the outer circle has a radius of $2j+1$. Each point of this annulus will be scanned by $(f+1)$ different robots. Therefore, after j steps, a disc of radius $2j+1$ will be entirely scanned by $(f+1)$ different robots. As all parts of the disc are visited at least once by a reliable robot, the target is guaranteed to be found.

We can use the proof from Lemma 1 to show that the robots will always gather at the target for both variations, which completes the proof of Lemma 4. □

Lemma 5. *The search performed by Algorithm 2 is completed in time*

$$\frac{4\lfloor d/2\rfloor + d}{v_1} + 2\pi\lfloor d/2\rfloor(\lfloor d/2\rfloor + 1)\left(\max_{0\le j\le f}\left\{\frac{f+1-j}{V_{k-j}}\right\}\right)$$

for the wireless communication variation and

$$\frac{4\lfloor d/2\rfloor + 8\left\lceil\frac{2\pi}{\alpha_i}\right\rceil + d + 4}{v_1} + 2\pi\left(\lfloor d/2\rfloor + 2\left\lceil\frac{2\pi}{\alpha_1}\right\rceil + 1\right)\left(\lfloor d/2\rfloor + 2\left\lceil\frac{2\pi}{\alpha_i}\right\rceil + 2\right)\left(\max_{0\le j\le f}\left\{\frac{f+1-j}{V_{k-j}}\right\}\right)$$

for the face-to-face communication variation.

Proof. Let $V_l = \sum_{i=1}^{l} v_i$ and $V'_l = \sum_{i=1}^{l} v'_i$ for all $1 \le l \le k$. First, we show the following claim:

Claim 2.

$$\max_{0\leq j\leq f}\left\{\frac{f+1-j}{V_{k-j}}\right\}=\frac{f+1}{V'_k}$$

Let $0 \leq l \leq f$ be the maximal value such that $v_{k-l+1} > \frac{V_{k-l}}{f+1-l}$ (we set $l = 0$ if the inequality is false for every $l \geq 1$). First, we show that l is the value of j that maximizes $c(j) = \frac{f+1-j}{V_{k-j}}$. For $l < j \leq f$, we have: $v_{k-j+1} \leq \frac{V_{k-j}}{f+1-j}$ by definition of l.

For $l < j \leq f$, we have:

$$V_{k-(j-1)} = V_{k-j} + v_{k-j+1} \leq \left(1+\frac{1}{f+1-j}\right)V_{k-j} = \left(\frac{f+1-(j-1)}{f+1-j}\right)V_{k-j}$$

and so we have:

$$\frac{V_{k-(j-1)}}{f+1-(j-1)} \leq \frac{V_{k-j}}{f+1-j}$$

and

$$c(j-1) = \frac{f+1-(j-1)}{V_{k-(j-1)}} \geq \frac{f+1-j}{V_{k-j}} = c(j)$$

By induction on j, for every $j > l$, we have $c(l) \geq c(j)$.

For $0 \leq j < l$, we have: $v_{k-j} > \frac{V_{k-l}}{f+1-l}$ since $v_{k-(l-1)} > \frac{V_{k-l}}{f+1-l}$ and for all $1 \leq i \leq k-1$, we have $v_i \leq v_{i+1}$. Hence for $0 \leq j < l$, we have :

$$V_{k-j} = V_{k-l} + \sum_{i=j}^{l-1} v_{k-i} > V_{k-l} + (l-j)\left(\frac{V_{k-l}}{f+1-l}\right) > \left(\frac{f+1-j}{f+1-l}\right)V_{k-l}$$

and

$$c(j) = \frac{f+1-j}{V_{k-j}} < \frac{f+1-j}{\left(\frac{f+1-j}{f+1-l}\right)V_{k-l}} = c(l)$$

We have shown that for $0 \leq j \leq f$, we have $c(j) \leq c(l)$.

Observe that for any $i < k-l+1$, we have $v'_i = v_i$ and so $V'_i = V_i$. For $i \geq k-l+1$, we have $v'_i = \frac{V_{k-l}}{f+1-l}$. We obtain:

$$V'_k = l\left(\frac{V_{k-l}}{f+1-l}\right) + V_{k-l} = \left(\frac{f+1}{f+1-l}\right) V_{k-l}$$

an so :

$$\frac{f+1}{V'_k} = \frac{f+1}{\left(\frac{f+1}{f+1-l}\right) V_{k-l}} = \frac{f+1-l}{V_{k-l}} = \max_{0 \leq j \leq f} \left\{ \frac{f+1-j}{V_{k-j}} \right\}$$

This ends the proof of the claim.

Suppose that the target is found at a point situated at distance d from the origin O, say by robot r_i. This happens in step $\lfloor d/2 \rfloor$ of the exploration phase (the while loop from line 25). For simplicity, we assume that the target may only be found during the circular movement of the robots. Indeed, the union of the robots' positions during circular movements cover the entire plane $(f+1)$ times. Consequently, the trajectory of the robot finding the target consists of $\lfloor d/2 \rfloor$ arcs (line 27) as well as $\lfloor d/2 \rfloor$ segments of length 2 resulting from the linear moves from lines 26. As r_i uses its full speed for circular moves and speed v_1 for straight line moves, this takes time

$$\frac{2\lfloor d/2 \rfloor}{v_1} + \frac{\alpha_i(2+4+\cdots+2\lfloor d/2 \rfloor)}{v'_i},$$

which is equal to

$$\frac{2\lfloor d/2 \rfloor}{v_1} + \frac{2\pi(f+1)\lfloor d/2 \rfloor(\lfloor d/2 \rfloor + 1)}{V'} \tag{3.5}$$

Once the location of the target has been identified by a robot, the information needs to be shared. Wireless communication allows this to happen instantly; face-to-face communication requires more time. The time required for the face-to-face

variation is calculated as follows: Once a robot identifies the location of the target, it will execute enough steps to cover the entirety of the circle, which will require $\lceil \frac{2\pi}{\alpha_i} \rceil$ extra steps at most, and we need one extra step to ensure we meet all robots, as two synchronised robots may move in the same direction and miss one another in the last step. Say that the robot to identify the target also happens to be the slowest robot r_1. Hence any robot will execute at most $\left(\lfloor d/2 \rfloor + 2\lceil \frac{2\pi}{\alpha_1} \rceil + 1\right)$ steps and the total time required for both exploration and communication phases amounts to:

$$\frac{2\left(\lfloor d/2 \rfloor + 2\lceil \frac{2\pi}{\alpha_1} \rceil + 1\right)}{v_1} + \frac{2\pi(f+1)\left(\lfloor d/2 \rfloor + 2\lceil \frac{2\pi}{\alpha_1} \rceil + 1\right)\left(\lfloor d/2 \rfloor + 2\lceil \frac{2\pi}{\alpha_i} \rceil + 2\right)}{V'}. \quad (3.6)$$

Finally, for both variations, the last robot informed (say, r_i) will have to go back to the location of the target. By the time r_i is informed, in the wireless case, it will be at a distance $2\lfloor d/2 \rfloor$ from the origin, and in the F2F case, it will be at a distance at most $2\left(\lfloor d/2 \rfloor + 2\lceil \frac{2\pi}{\alpha_1} \rceil + 1\right)$ from the origin O, that we obtain by considering only the linear movements of the robot. Supposing that the target is located at a point directly opposed to r_i, then r_i will have to cover a distance of $2\lfloor d/2 \rfloor + d$ in the wireless case, and $2\left(\lfloor d/2 \rfloor + 2\lceil \frac{2\pi}{\alpha_1} \rceil + 1\right) + d$ in the F2F case, which it will do at its maximal speed. Therefore, the worst gathering time for the wireless case is:

$$\frac{2\lfloor d/2 \rfloor + d}{v_1}, \quad (3.7)$$

and the worst gathering time for the F2F model is:

$$\frac{2\left(\lfloor d/2 \rfloor + 2\lceil \frac{2\pi}{\alpha_1} \rceil + 1\right) + d}{v_1} \quad (3.8)$$

when we consider that r_1 is the last robot informed.

We can therefore conclude that the total required time for wireless communication

is

$$(3.5) + (3.7) = \frac{4\lfloor d/2 \rfloor + d}{v_1} + \frac{2\pi(f+1)\lfloor d/2 \rfloor (\lfloor d/2 \rfloor + 1)}{V'}$$

which by Claim 2 is equal to :

$$\frac{4\lfloor d/2 \rfloor + d}{v_1} + 2\pi \lfloor d/2 \rfloor (\lfloor d/2 \rfloor + 1) \left(\max_{0 \le j \le f} \left\{ \frac{f+1-j}{V_{k-j}} \right\} \right).$$

For for face-to-face communication, the total time is obtained by summing Formulas (3.6) and (3.8) we conclude that the total time is:

$$(3.6) + (3.8) = \frac{4\lfloor d/2 \rfloor + 8\left\lceil \frac{2\pi}{\alpha_i} \right\rceil + d + 4}{v_1} + \frac{2\pi(f+1)\left(\lfloor d/2 \rfloor + 2\left\lceil \frac{2\pi}{\alpha_1} \right\rceil + 1\right)\left(\lfloor d/2 \rfloor + 2\left\lceil \frac{2\pi}{\alpha_i} \right\rceil + 2\right)}{V'}$$

which by Claim 2 is equal to :

$$\frac{4\lfloor d/2 \rfloor + 8\left\lceil \frac{2\pi}{\alpha_i} \right\rceil + d + 4}{v_1} + 2\pi \left(\lfloor d/2 \rfloor + 2\left\lceil \frac{2\pi}{\alpha_1} \right\rceil + 1 \right) \left(\lfloor d/2 \rfloor + 2\left\lceil \frac{2\pi}{\alpha_i} \right\rceil + 2 \right) \left(\max_{0 \le j \le f} \left\{ \frac{f+1-j}{V_{k-j}} \right\} \right).$$

□

Next we discuss a lower bound on any search algorithm.

Lemma 6. *Every search algorithm with f crash-faulty robots has time complexity at least:*

$$\left(\frac{\pi(d^2-1)}{2} \right) \left(\max_{0 \le j \le f} \left\{ \frac{(f+1-j)}{\sum_{i=1}^{k-j} v_i} \right\} \right).$$

Proof. For any j such that $0 \le j \le f$, the adversary may choose the j fastest robots $r_{k-j+1}, \ldots, r_k$ to be faulty. Observe that in order to explore all points at distance at most d from the origin, each such point must be viewed by at least $f+1-j$ other robots. Indeed, if any such point is seen by at most $f-j$ robots, the adversary

can make them all faulty and the point remains unexplored. Therefore, the robots (except the j fastest ones) altogether have to cover the area of $\pi(d^2-1)(f+1-j)$. As, similarly as in Lemma 3, in time t the robots can explore a new area of size at most $2t\left(\sum_{i=1}^{k-j} v_i\right)$, the time needed to complete the exploration equals:

$$\frac{\pi(d^2-1)(f+1-j)}{2\sum_{i=1}^{k-j} v_i}$$

Since the adversary can choose any j such that $1 \leq j \leq f$, the lower bound is the maximum among all j:

$$\max_{0\leq j\leq f}\left\{\frac{\pi(d^2-1)(f+1-j)}{2\sum_{i=1}^{k-j} v_i}\right\} = \left(\frac{\pi(d^2-1)}{2}\right)\left(\max_{0\leq j\leq f}\left\{\frac{(f+1-j)}{\sum_{i=1}^{k-j} v_i}\right\}\right)$$

□

To sum up we have proved the following theorem.

Theorem 2 (Crash-Faults Robots). *Algorithm 2 completes the search successfully for k robots, including at most f of which are crash-faulty, in worst-case time*

$$\left(\frac{\pi d^2}{2}\right)\left(\max_{0\leq j\leq f}\left\{\frac{(f+1-j)}{\sum_{i=1}^{k-j} v_i}\right\}\right) + o(d^2)$$

where v_i for $1 \leq i \leq k$ is the speed of robot r_i and d is the distance from the origin O to the target, when the communication is face-to-face. This is asymptotically optimal.

3.4 Robots with Byzantine Faults (BF)

In this section, we introduce an algorithm for robots with weak Byzantine faults, prove its correctness and analyse its complexity. Weak Byzantine robots may experience

crash-faults, as described above. In addition to those faults, they may choose to propagate information that will hinder the progress of the algorithm. They can lie about everything but their own identity (a Byzantine robot couldn't pretend to be a healthy robot). The notion of weak Byzantine robot exists in contrast to the notion of strong Byzantine robots, which have the capability to lie about everything, including their own identity. In this thesis, we focus on weak Byzantine robots.

3.4.1 Wireless communication

The algorithm works in a fashion similar to the one for Crash-Faulty Robots for the wireless communication model. All phases are in fact identical. However, in the CF case each phase is executed once only while in the case of Byzantine faults, the sequence of phases is iterated $f + 1$ times. In each iteration, when a target signalled turns out to be false, the robot responsible for its announcement is rejected and the remaining robots reorganize and continue to search the next annulus. Eventually, the target is found in one of the subsequent iterations or the robots eliminate all faulty robots and in the $(f + 1)$-st iteration the real target is effectively found. With respect to the CF case an extra cost is paid for the robots reorganization, between consecutive iterations of the algorithm. The total time is asymptotically dominated by the exploration cost, which does not increase. This is based on the fact that the search by k robots containing at most f faulty ones is more costly than the search by $k - i$ robots with $f - i$ faulty ones for $i > 0$

If two targets are identified at the same time, all robots agree on one of the targets to visit (using any given algorithm), then go visit the other.

Lemma 7. *The wireless algorithm for byzantine robots described above completes the*

gathering of all reliable robots at the location of the target in time

$$\frac{2\lfloor d/2\rfloor + 4(d+1)(f+1)}{v_1} + 2\pi\lfloor d/2\rfloor(\lfloor d/2\rfloor + 1)\left(\max_{0\le j\le f}\left\{\frac{f+1-j}{\sum_{i=1}^{k-j} v_i}\right\}\right).$$

Proof. The total time required for all reliable robots to gather at the location of the target is equal to the total time required to identify the target, plus the total time required to investigate each potential target.

The time required to identify the location of the target is again at most

$$\frac{2\lfloor d/2\rfloor}{v_1} + \frac{\alpha_i(2+4+\cdots+2\lfloor d/2\rfloor)}{v_i'},$$

which is equal to

$$\frac{2\lfloor d/2\rfloor}{v_1} + 2\pi\lfloor d/2\rfloor(\lfloor d/2\rfloor + 1)\left(\max_{0\le j\le f}\left\{\frac{f+1-j}{\sum_{i=1}^{k-j} v_i}\right\}\right). \tag{3.9}$$

If the adversary decides to place the location of all possible targets on the last annulus to be explored (that is, every potential target is at a distance exactly $(d+1)$ of the origin), then r_1 has to cover a distance of at most $4(d+1)(f+1)$ before arriving at the location of the real target.

This takes time of at most

$$\frac{4(d+1)(f+1)}{v_1} \tag{3.10}$$

and the total execution time is bound by

$$(3.9) + (3.10) = \frac{2\lfloor d/2\rfloor + 4(d+1)(f+1)}{v_1} + 2\pi\lfloor d/2\rfloor(\lfloor d/2\rfloor + 1)\left(\max_{0\le j\le f}\left\{\frac{f+1-j}{V_{k-j}}\right\}\right)$$

This proves Lemma 7. □

3.4.2 Face-to-face communication

We introduce the following two algorithms to solve the byzantine face-to-face communication problem.

The $(f+1)$-Confirmations Algorithm

In this algorithm, we cover each annulus $2f+1$ times, instead of $f+1$ as we did with the crash-faulty robots. Each robot follows its own exploration path until it receives $f+1$ confirmations of the same location of the target, before switching to the gathering phase. Specifically, we have:

Initialisation phase: the robots divide the plane in angular sectors, in a fashion similar to what was done in the crash-faults scenario. Each point of the plane is covered by $2f+1$ different robots.

Exploration phase: identical to the crash-fault scenario.

Communication phase: each robot that finds the target executes $\lceil \frac{2\pi}{v_i} \rceil + 1$ extra steps, going only counter-clockwise (instead of changing direction at every step), thus meeting every other robot which is still in its exploration phase. Once this is done, they execute their gathering phase. A robot that does not find the target stays in its exploration phase until it is informed about the same location of the target by $f+1$ incoming robots, then executes its gathering phase.

Gathering phase: the robot moves in a straight line toward the target.

Lemma 8. *The algorithm described above is correct, i.e. the target is always found and the robots eventually gather at the location of the target.*

Proof. In order to prove the correctness of the algorithm, we have to show 1) that the target is always found, and 2) that all the robots are eventually informed and gather

at the target. To prove the first part, we note that at each step of the exploration phase, the robots scan an annulus whose inner circle has radius $2i-1$, and the outer circle has radius $2i+1$, with i being the current step. Each point of the annulus is visited by $2f+1$ robots, thus ensuring that at least $(f+1)$ reliable robots see the target. As all points are visited at least once by $(f+1)$ reliable robots, the target is guaranteed to be found.

Once the robots have found the target, they start informing every other robot. As the algorithm does not allow for a robot to deviate from its exploration course before receiving $(f+1)$ confirmations, the robots are guaranteed to meet every other robot, and the byzantine robots will be unable to sway any reliable robot. Since each robot that saw the target will eventually meet every other reliable robot, it is guaranteed that every reliable robot that did not see the target directly will hear about the target at least $(f+1)$ times. Once this is done, they have a confirmation of the location of the target. □

Lemma 9. *The search performed by the algorithm described above is completed in time*

$$\frac{4\left(\lfloor d/2\rfloor + 2\left\lceil\frac{2\pi}{\alpha_1}\right\rceil + 1\right) + d}{v_1} + 4\pi\left(\lfloor d/2\rfloor + 2\left\lceil\frac{2\pi}{\alpha_1}\right\rceil + 1\right)\left(\lfloor d/2\rfloor + 2\left\lceil\frac{2\pi}{\alpha_i}\right\rceil + 2\right)\left(\max_{0\le j\le f}\left\{\frac{f+1-j}{V_{k-j}}\right\}\right).$$

Proof. Suppose that the target is found at a point situated at a distance d from the origin O, say by robot r_i. This happens in step $\lfloor d/2\rfloor$ of the exploration phase. For simplicity, we assume that the target may only be found during the circular movement of the robots. Indeed, the union of the robots' positions during circular movements cover the entire plane $(2f+1)$ times. Once the location of the target has been identified by a robot, the information needs to be shared. The robot which holds the information needs to go in the counterclockwise direction covering arcs of the entire

circle, and then one extra step to meet a robot he may have missed because both robots were moving in the same direction. Consequently, we need $\lceil \frac{2\pi}{\alpha_i} \rceil + 1$ extra steps at most. As the speed of the robot to identify the target is also at least v_1, the total time required for both exploration and communication phase is bound by:

$$\frac{2\left(\lfloor d/2 \rfloor + 2\left\lceil \frac{2\pi}{\alpha_1} \right\rceil + 1\right)}{v_1} + \frac{2\pi(2f+1)\left(\lfloor d/2 \rfloor + 2\left\lceil \frac{2\pi}{\alpha_1} \right\rceil + 1\right)\left(\lfloor d/2 \rfloor + 2\left\lceil \frac{2\pi}{\alpha_i} \right\rceil + 2\right)}{V'}. \tag{3.11}$$

The last robot informed (say, r_i) will have to go to the location of the target. By the time r_i is informed, it will be at a distance $2\left(\lfloor d/2 \rfloor + 2\left\lceil \frac{2\pi}{\alpha_1} \right\rceil + 1\right)$ from the origin O. In the worst case, the target is located at a point directly opposed to r_i, hence r_i will have to cover that distance plus d to reach the target, walking at its maximal speed. Therefore, the worst gathering time (when the slowest robot r_1 is the last one informed) is bound by

$$2\left(\lfloor d/2 \rfloor + 2\left\lceil \frac{2\pi}{\alpha_1} \right\rceil + 1\right) + d. \tag{3.12}$$

The total time is obtained by summing Formulas (3.11) and (3.12); we conclude that the total time is

$$(3.11) + (3.12) = \frac{4\left(\lfloor d/2 \rfloor + 2\left\lceil \frac{2\pi}{\alpha_1} \right\rceil + 1\right) + d}{v_1} + \frac{2\pi(2f+1)\left(\lfloor d/2 \rfloor + 2\left\lceil \frac{2\pi}{\alpha_1} \right\rceil + 1\right)\left(\lfloor d/2 \rfloor + 2\left\lceil \frac{2\pi}{\alpha_i} \right\rceil + 2\right)}{V'}$$

which by Claim 2 is equal to :

$$\frac{4\left(\lfloor d/2 \rfloor + 2\left\lceil \frac{2\pi}{\alpha_1} \right\rceil + 1\right) + d}{v_1} + 4\pi\left(\lfloor d/2 \rfloor + 2\left\lceil \frac{2\pi}{\alpha_1} \right\rceil + 1\right)\left(\lfloor d/2 \rfloor + 2\left\lceil \frac{2\pi}{\alpha_i} \right\rceil + 2\right)\left(\max_{0 \le j \le f}\left\{\frac{f+1-j}{V_{k-j}}\right\}\right).$$

This proves Lemma 9. □

The Fault-Reduction Algorithm

In this algorithm, we cover each point by $f+1$ robots, as we in the CF case. The

initialization phase and exploration phase are identical to the crash-fault scenario. We have the following communication and gathering phases:

Communication phase. Once a robot finds the target, it takes circular movements in the counter-clockwise direction, attempting to meet every robot still in its exploration phase. It takes $\lceil \frac{2\pi}{v_i} \rceil + 1$ steps which involve straight-line moves retreating from the origin. Once a robot that did not find the target is informed, it will itself start doing the same counterclockwise moves to inform every other robot of a possible location of the target. Once a robot is done with its sequence of informing moves, it will execute its gathering phase.

Gathering phase. The robot goes to the origin, and waits until $(k - f)$ robots are present there. It then waits $\frac{2\pi}{\alpha_1} + 1$ steps, and then $\frac{d + \frac{2\pi}{\alpha_1} + 1}{v_1}$ additional steps (thus ensuring that every reliable robot arrives at the origin). Every robot that did not arrive at the origin after this delay is immediately considered byzantine, and is therefore ignored for the rest of the algorithm. More so, every robot that arrived $\frac{2\pi}{\alpha_1} + 1 + \frac{d + \frac{2\pi}{\alpha_1} + 1}{v_1}$ steps earlier is also considered byzantine, and is ignored for the rest of the algorithm.

Once every reliable robot has arrived, they all move to the location of the potential target. If more than one target location is identified, the robots go to the location which was found first.

Once all reliable robots gathered at the location of a target, and the target turns out to be false the robots apply a procedure to identify the liar. By elimination, a pair of robots which are contradicting each other is identified: the one that created the message, and the one that received it. Both such robots, the one that sent the message and the one that received it, are considered byzantine, and are ignored for the rest of the algorithm. Note that a reliable robot may be eliminated this way, but as two robots are contradicting each other and the remaining robots cannot collectively identify who lies, they are both discarded from future actions.

Once this is done, the remaining robots re-arrange their sectors of the plane, and restart the exploration of the next annulus. In such a way, after x false targets were identified, and x pairs of robots were discarded as being byzantine, the annulus is covered $(f-x+1)$ times, and $(k-2x)$ robots are still working to continue the search.

Lemma 10. *Fault-Reduction Algorithm is correct, i.e. the target is always found and the robots eventually gather at the location of the target, under the condition that* $k > 2f$.

Proof. In order to prove the correctness of the algorithm, we have to show 1) that the target is always found, and 2) that all the robots are eventually informed and gather at the target. To prove the first part, we note that at each step of the exploration phase, the robots will scan an annulus whose inner circle has radius $2i-1$, and the outer circle has radius $2i+1$, with i being the current step. Each point of the annulus is visited by $f+1$ different robots, thus ensuring that at least one reliable robot sees the target. As all points are covered by at least one reliable robot, the target is guaranteed to be found. Once a reliable robot has found the target, it will start informing every other robot, thus ensuring that every robot receives the information. Byzantine robots that want to mislead the reliable robots may do so f times, thus multiplying the number of required communication and gathering phases by $f+1$. However, each time this is done, we get rid of at least one byzantine robot. Robots will never be de-synchronized from the rest of the swarm, since as soon as a reliable robot is informed, it also informs every other robot, and they all gather at the origin within a time which is linear in d. Robots will never wait indefinitely in the origin, as the waiting time is bound linearly in d as well. Therefore, robots can never be stalled indefinitely, and they will eventually all gather at the location of the real target. □

Lemma 11. *If every robot of the collection has speed $v = 1$, then the search performed by the Fault-Reduction Algorithm is completed in time $\frac{f+1}{2k}\pi d^2 + o(d^2)$*

Proof. The total time required by the Fault-Reduction Algorithm is equal to the sum of the exploration time, and the communication and gathering phases, which can both be done up to $(f+1)$ times.

Each time a byzantine robot claims to have found a target, we will eliminate two robots, including at least one robot which is byzantine. Therefore, for each occurrence i of a byzantine robot declaring a false target at distance d_i, we execute a communication phase, a gathering phase, and we redistribute the next annulus to cover between the remaining robots. The total time spend in the exploration phase can therefore be bound by

$$\frac{f+1}{2k}A(0,d_1)+\frac{f}{2(k-2)}A(d_1,d_2)+\cdots+\frac{2}{2(k-2(f-1))}A(d_{f-1},d_f)+\frac{1}{2(k-2f)}A(d_f,d)$$

where $A(a,b)$ represents the time required to cover an entire annulus by a robot, with the inner circle of radius a and the outer circle of radius b. Observe that $\frac{f+1}{2k} \geq \frac{f+1-i}{2(k-2i)}$, for $k \geq 2(f+1)$ and $1 < i < f+1$. Therefore, we can bound the expression above by:

$$\frac{f+1}{2k}A(0,d)$$

Consequently, we obtain that the exploration phase takes at most

$$2\lfloor d/2 \rfloor + \frac{2\pi(f+1)\lfloor d/2 \rfloor(\lfloor d/2 \rfloor + 1)}{2k} \tag{3.13}$$

Each time a robot claims to have identified a target, it must inform every other robot, thus following counterclockwise arcs summing up to 2π. The last robot in-

formed must in turn cover counterclockwise arcs summing up to 2π, then gather at the origin. From the moment the target is identified initially, at a distance at most d of the origin O, to the moment where all reliable robots are gathered at the origin O, the time elapsed is at most

$$d + \frac{2\left\lceil \frac{2\pi}{\alpha_1} \right\rceil + 2}{v_1} + \frac{2\pi(f+1)}{k}\left(\left\lceil \frac{2\pi}{\alpha_1} \right\rceil + 1\right)^2 + 2\left(\frac{2\pi(f+1)}{k}\right)\left(\left\lceil \frac{2\pi}{\alpha_1} \right\rceil + 1\right)d$$

Once every robot is at the origin, they will travel a distance of at most d to verify if the target is the real one. If it isn't, the farthest robot will have to travel a distance of at most $2d$ to take its position to resume the exploration. This is done at speed $v = 1$. As this entire process may have to be repeated at most $f + 1$ times, we can bound the maximal amount of time required to identify the real target with:

$$(f+1)\frac{2\left\lceil \frac{2\pi}{\alpha_1} \right\rceil + 2}{v_1} + (f+1)\frac{2\pi(f+1)}{k}\left(\left\lceil \frac{2\pi}{\alpha_1} \right\rceil + 1\right)^2 + (f+1)\left(2\left(\frac{2\pi(f+1)}{k}\right)\left(\left\lceil \frac{2\pi}{\alpha_1} \right\rceil + 1\right)d\right) + (f+1)4d \tag{3.14}$$

The total time is obtained by summing Formulas (3.13) and (3.14); we conclude that the total time is

$$\begin{aligned}(3.13) + (3.14) &= 2\lfloor d/2 \rfloor + \frac{2\pi(f+1)\lfloor d/2 \rfloor(\lfloor d/2 \rfloor + 1)}{2k} \\ &\quad + (f+1)\frac{2\left\lceil \frac{2\pi}{\alpha_1} \right\rceil + 2}{v_1} + (f+1)\frac{2\pi(f+1)}{k}\left(\left\lceil \frac{2\pi}{\alpha_1} \right\rceil + 1\right)^2 \\ &\quad + (f+1)\left(2\frac{2\pi(f+1)}{k}\left(\left\lceil \frac{2\pi}{\alpha_1} \right\rceil + 1\right)d\right) + (f+1)4d \\ &= \frac{f+1}{2k}\pi d^2 + o(d^2)\end{aligned}$$

This proves Lemma 11. □

Observe that the lower bound of Lemma 6 for f crash-faulty robots is clearly valid for f byzantine robots. To sum up we have proved the following theorem.

Theorem 3 (Byzantine Robots). *In the wireless case, our algorithm completes the search successfully for k robots, including f byzantine ones, in worst-case time*

$$c = \left(\frac{\pi d^2}{2}\right)\left(\max_{0\le j\le f}\left\{\frac{(f+1-j)}{\sum_{i=1}^{k-j} v_i}\right\}\right) + o(d^2),$$

*where v_i for $1 \le i \le k$ is the speed of robot r_i and d is the distance from the origin O to the target. More so, in the face-to-face case, the $(f+1)$-**Confirmations Algorithm** completes the search successfully for k robots, including f byzantine ones such that $2f < k$, in worst-case time $2c$ and the **Fault-Reduction Algorithm** described above for the F2F model completes the search successfully for k robots of equal speed $v = 1$, including f byzantine ones such that $2f + 2 \le k$, in worst-case time $\frac{(f+1)\pi d^2}{2k} + o(d^2)$.*

3.5 Additional Remarks and Conclusion

We studied search in the plane with faulty robots under two different scenarios. In the first one, some robots can be subject to crashes, and stop working properly, whereas in the second one, robots may be byzantine, and actively work against our goal. For both scenarios, we considered the wireless and the F2F communication model. We designed an algorithm which has asymptotically optimal search time, for all of those cases, with the exception of the byzantine F2F model. In this case, we offer two algorithms: the first one works for any $2f + 1 \le k$ and is 2-competitive, and the second one works for $2f + 1 < k$, and assumes that all robots have equal speed. This second algorithm is optimal.

The algorithms we presented consider robots with a visibility range of 1. We

suggest that our algorithms can be adjusted to take into account robots with visibility range that vary from one to the other, by modifying the angular value of each robot's coverage sector so as to take their visibility range into account. We leave this subject as an open problem.

Chapter 4

Searching for a Non-adversarial, Uncooperative Agent on a Cycle

4.1 Introduction

Due to their fundamental nature, the problems of searching and exploration have been investigated in many areas of mathematics and computer science, especially in robotics and autonomous mobile agent computing. The robots move with certain speeds (not necessarily the same) and the objective of the search is to find a (usually static) target placed at an unknown location of the domain in a (provably) optimal time. This search problem was first proposed by Bellman [20] and independently by Beck [19].

In this chapter we consider a similar search problem concerning k mobile autonomous robots which are initially located on the perimeter of a unit cycle and which can move with maximum speed 1 on its perimeter. Unlike previous research which considers a static target, in our work the robots are aware that a bus (non-adversarial, uncooperative agent) is moving with constant speed, say s, along the

perimeter of the unit cycle but do not know its exact location and may or may not know its direction of movement. We assume that during their search the robots can move in any direction anywhere on the perimeter of the cycle and can also communicate wirelessly at any distance during their trajectories.

More specifically, we are interested in investigating the following search problem: Give an algorithm which places the robots on the perimeter of the cycle and minimizes the search time so that at least one of the robots can catch the bus. By the "robot catching the bus" we mean that the robot and the bus are at the same location at the same time.

4.1.1 Preliminaries and model of computation

We assume that the robots are traversing a cycle (the perimeter of a disk of unit radius). Furthermore, there is a bus which is rotating around the cycle at constant speed s and its location is unknown to the robots. The robots can communicate wirelessly and when a robot finds the bus it can broadcast its location to the rest of the robots. Note that in the single robot case ($k = 1$), wireless communication is unnecessary since the search algorithm is executed by the robot alone.

The robots can move at speed at most 1 on the perimeter of the disk and can change direction at will at any time during the search depending on the specifications of the algorithm. An algorithm specifies the initial position and trajectories of the robots. For k robots, their movement is specified by a k-tuple $(f_1(t), f_2(t), \ldots, f_k(t))$ of k continuous functions such that $f_i(t)$ gives the precise location of the i-th robot on the cycle at time t, where $i = 1, 2, \ldots, k$. Without loss of generality we may assume that the robots start at the same time while the bus is always in motion around the cycle.

4.1.2 Related work

Our problem can be seen as a rendezvous/meeting problem with an uncooperative, but not adversarial agent, a middle case between rendezvous and cops and robbers. In the standard rendezvous model, all agents fully cooperate to the common meeting goal. Indeed, this is the case in the related paper [109] on rendezvous of two robots with different speeds in a cycle (our problem is different in that one of the two vehicles—namely the bus—has a fixed speed and cannot change direction). At the other extreme, in cops and robbers problems (e.g., see [28]), the cops have the same goal of meeting the robber, but the robber is adversarial and actively tries to avoid meeting. Here (at least for search), we are also trying to meet with an agent. However, that agent does not cooperate, but goes doing its own business, not caring whether it is met or not.

The underlying domain which is traversed by the robots is a continuous curve (in our case the perimeter of a disk of unit radius). In this setting, in addition to the rendezvous paper [109], related to our rendezvous problem is the work on probabilistic rendezvous for robots with different speeds [160], rendezvous for multiple robots with different speeds in [143], and rendezvous for two robots with different speeds in arbitrary graphs [162].

Related is the literature on search involving a robot and a static exit in the seminal papers [11, 19, 20] as well as extensive discussions and models in the books on search problems [3], on the theory of rendezvous games [7], and on the game of cops and robbers [28]. More recently, there is research on robot evacuation which is like search but measures the quality of search by the time it takes the last robot to find an exit; this has been investigated in the wireless model as well as in the face-to-face model [56]. Related papers on robot evacuation include two robots in the face-to-face

model [32, 62] and [68] in the wireless model when the underlying domain is a triangle or a square.

There is also related work on gathering a collection of identical memoryless, mobile robots in one node of an anonymous unoriented ring. Robots start from different nodes of the ring and operate in Look-Compute-Move cycles and have to end up in the same node [152], as well as oblivious mobile robots in the same location of the plane when the robots have limited visibility [121].

4.1.3 Outline and results

Depending on the model being considered the robots may or may not have knowledge of the direction of movement and speed of the bus. In particular, in this chapter we determine the search time for the following cases. The robots

1. do not know the direction of movement but know the speed of the bus, and

2. know neither the direction of movement nor the speed of the bus.

We note that if the robot knows the direction of movement of the bus, then it has been proved in [109] that $2\pi/(s+1)$ is a tight bound on the search time.

In Section 4.2 we provide tight upper and lower bounds for single robot search, while in Section 4.3 we provide tight upper and lower bounds for multiple robot search. In both sections we consider the impact of knowing the direction of movement of the bus. Table 4.1 summarizes the results of Section 4.2 for a single robot and Table 4.2 summarizes the results of Section 4.3 concerning multiple robots.

Direction	Speed	Search Time	Theorem
Known	s	$2\pi/(s+1)$	Theorem 4 [109]
Unknown	$s \geq 1$	$2\pi/s$	Theorem 5
Unknown	$1/3 \leq s \leq 1$	$4\pi/(s+1)$	Theorem 5
Unknown	$s \leq 1/3$	$2\pi/(1-s)$	Theorem 5
Unknown	Unknown	4π	Theorem 6

Table 4.1: Optimal search time for a single robot of maximum speed 1. The column "Speed" refers to what the robot knows about the speed of the bus.

Direction	Speed	Search Time	Theorem
Known	s	$2\pi/k(s+1)$	Theorem 7
Unknown	$s \geq 1$	$2\pi/ks$	Theorem 8
Unknown	$s \leq 1$	$2\pi/k$ (k even)	Theorem 8
Unknown	$s \leq 1$	$2\pi/k$ (k odd)	OPEN
Unknown	Unknown	$2\pi/k$ (k even)	Theorem 9
Unknown	Unknown	$2\pi/(k-1)$ (k odd)	OPEN

Table 4.2: Optimal search time for k robots of maximum speed 1. The column "Speed" refers to what the robots know about the speed of the bus.

4.2 One Robot

Consider the case of a single robot R and let B denote the bus and P, the path followed by the robot. Throughout this section we assume that the bus is moving at constant speed s and cannot change direction, while the robot is moving with speed 1. Our analysis is divided into three subsections depending on the knowledge the robot has about the bus. In Subsection 4.2.1 we assume only that the robot knows the direction of movement of the bus, in Subsection 4.2.2 the robot does not know the direction of movement of the bus but knows its speed, while in Subsection 4.2.3 the robot knows neither the direction nor the speed s of the bus. Table 4.1 summarizes the results of Section 4.2 for a single robot.

4.2.1 Known direction of movement of the bus

In this subsection we assume that the robot knows only the direction of movement of the bus. The following theorem was first proved in [109] but we state it for completeness.

Theorem 4 ([109]). *If the robot knows the direction of movement of the bus then*

$$\frac{2\pi}{s+1}$$

is the worst-case optimal search time.

Proof. (Theorem 4) Without loss of generality assume the bus is moving in the CCW direction. To prove the upper bound consider the following algorithm (depicted in Figure 4.1) which is executed by the robot.

Search Algorithm (Direction Known).
1. Move along the perimeter in CW direction;
2. Stop when bus is found;

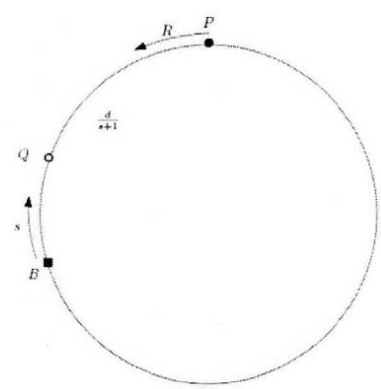

Figure 4.1: Robot search for a moving bus B. The bus is moving with speed s along the perimeter of a cycle and the robot is moving with speed 1 searching for the bus.

Since the robot chooses to move in a direction opposite to the direction of movement of the bus, the two will meet. The initial distance (when the robot reaches the perimeter) between robot and bus is at most 2π and therefore the two will meet in time not exceeding $\frac{2\pi}{s+1}$.

The proof of the lower bound $\frac{2\pi}{k(s+1)}$ follows from the work of [109]. This proves Theorem 4. □

4.2.2 Unknown direction of movement but known speed of the bus

In this subsection we assume that the robot does not know the direction of movement but knows the speed s of the bus.

Theorem 5. *If the robot knows the speed s of the bus but does not know its direction of movement then the optimal search time is exactly*

1. $2\pi/s$ *if* $s \geq 1$.
2. $4\pi/(s+1)$ *if* $\frac{1}{3} \leq s \leq 1$.
3. $2\pi/(1-s)$ *if* $s \leq \frac{1}{3}$.

Proof. (Theorem 5) We prove separately the upper and lower bounds in all three cases in the statement of the theorem.

Upper bounds. The upper bounds are relatively simple and we present below three simple algorithms.

To prove Statement 1 assume that $s \geq 1$. In the search algorithm below, the robot stays put and waits for the bus to arrive.

Search Algorithm (Direction Unknown: $s \geq 1$).

1. The robot waits for the bus to arrive.

The upper bound $\frac{2\pi}{s}$ in Statement 1 is immediate since the bus travels with speed s and the robot is at distance at most 2π from the bus.

To prove Statement 2, assume that $\frac{1}{3} \leq s \leq 1$ and consider the following search algorithm.

Search Algorithm (Direction Unknown: $1/3 \leq s \leq 1$).

1. The robot chooses a direction and walks for time $\frac{2\pi}{s+1}$;
2. If no bus found then it changes direction and walks until bus is found;

The upper bound $\frac{4\pi}{s+1}$ in Statement 2 is easy since in the first part of the algorithm the robot walks for time $\frac{2\pi}{s+1}$. If it did not meet the bus by this time it is because the bus is moving in the same direction as the robot. Therefore at the moment the robot changes direction, in the second part of the algorithm, it is certain that it is moving against the bus. Therefore it will meet the bus in additional time $\frac{2\pi}{s+1}$.

To prove Statement 3, assume that $s \leq \frac{1}{3}$ and consider the following search algorithm.

Search Algorithm (Direction Unknown: $s \leq 1/3$).

1. Robot chooses an arbitrary direction and walks until bus is found;

The upper bound $\frac{2\pi}{1-s}$ in Statement 3 is easy since in the worst case the bus and the robot are moving in the same direction with initial distance at most 2π.

Lower bounds. Next we proceed to prove the lower bounds for all three cases in the statement of the theorem.

Let us introduce a visualisation that will be used in the other lower bounds as well. The x-axis represents time and the y-axis represents positions in the circle. The bus trajectory is then represented by a line passing though the initial position of the

bus, with the slope determined by the speed and direction of the bus (let downslope mean counterclockwise direction).

The robot's trajectory from time 0 until time T will be represented by a contiguous curve P (possibly consisting of straight line segments) in this time-space diagram. Let p_s and p_e denote the start and end-points of P. In order for the robot to catch the bus, its trajectory will have to cross the bus lines corresponding to all possible initial positions, directions (and possibly speeds, if the speed is unknown) of the bus. Such a robot trajectory will be called a *valid* one. Note that the validity of the trajectory depends on the assumptions/knowledge about the bus's speed s, i.e. a trajectory valid for a given s might not be valid for different s.

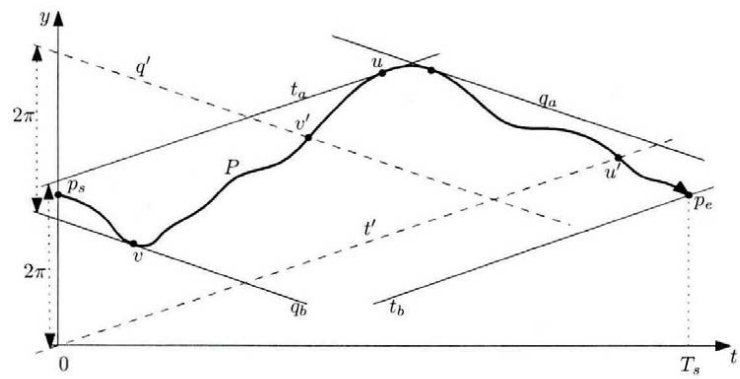

Figure 4.2: Trajectory and other concepts: u, v, u' and v' are *key points*. t_a, t', q_q and q' are *support lines*.

Consider first the case where the speed of the bus is known to be s, but its direction is unknown. For a fixed P, let t_a^s and t_b^s denote tangents of slope s touching P from above and below, respectively. Since here we deal with fixed s, we will omit superscripts, and use shorthands q_a and q_b for t_a^{-s} and t_b^{-s}, respectively (refer to Figure 4.2).

Let $z(x)$ denote the y-coordinate of line z at time x. Hence, $t_a(0)$ and $t_b(0)$ represent the starting positions of the buses moving at speed s that touch the trajectory

of the robot from above and from below.

Lemma 12. *If the bus speed s is known but its direction is unknown, then P is valid if and only if $t_a(0) - t_b(0) \geq 2\pi$ and $q_a(0) - q_b(0) \geq 2\pi$*

Proof. As the y axis represents the unfolded perimeter of the cycle, every point of the cycle is represented in any segment of y-axis of length at least 2π. In particular, the segment $\langle t_b(0), t_a(0)\rangle$. Note that since P is contiguous, every line of slope s lying between t_b and t_a represents a bus starting at this segment intersecting P. As the same argument holds for direction $-s$, P is valid.

If (without loss of generality) $t_a(0) - t_b(0) < 2\pi$, there is a point in the circle not covered by the segment $\langle t_b(0), t_a(0)\rangle$. A bus line of slope s crossing this point does not intersect P, therefore P is not valid. □

Let u be the earliest (in time) of the intersections of t_a or t_b with P. Let t' be a line of slope s at vertical distance exactly 2π from u and lying between t_a and t_b. Let u' be the earliest intersection of t' with P. Note that since t' is between t_a and t_b, it is ensured to intersect P, hence u' is well defined. Define v and v' for the slope $-s$ analogously (see Figure 4.2). Let us call points u, v, u', v' the *key* points of P.

Let P' be a trajectory starting at the earliest of the key points, following P and finishing at the latest key point (v to u' in Figure 4.2). We will call such trajectory a *pruned* trajectory. By Lemma 12 and its construction, P' is also a valid trajectory for s. (Note that translating a trajectory does not change its validity status, as translations correspond to different starting time and position of the robot.)

We are now ready to prove Statement 1 of Theorem 5: Since $s \geq 1$, we immediately get that $u = v = p_s$. By Lemma 12, in order for P to be valid, it must touch both t' and q' (see Figure 4.3). Note that the robot can do so in time $2\pi/s$ by simply waiting at the starting location. Assume, on the contrary, that it moves and first touches

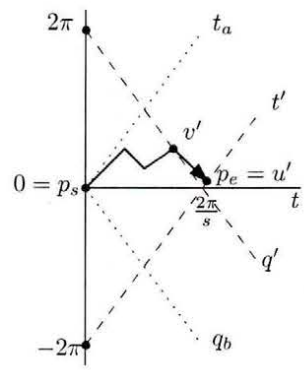

Figure 4.3: The case $s \geq 1$.

(without loss of generality) q' at point v'). Then the earliest it can touch t' is by moving towards it (counterclockwise). However, it will not be able to reach t' sooner than q' (which is also moving counterclockwise, but at speed $s \geq 1$), which reaches t' at time exactly $2\pi/s$.

Consider now the case $s < 1$. We know that we can prune the trajectory without increasing its span or breaking its validity. In fact, the following Lemma tells us that it is sufficient to consider only trajectories with robot moving at full speed at all time:

Lemma 13. *Let P' be a valid pruned trajectory of span T. Then there exists a valid trajectory P'' of span at most T in which the robot always travels at full speed.*

Proof. Let a, b, c and d be the key points of P', ordered by time. Let r_a, r_b, r_c and r_d be the corresponding support lines. P' will be modified as follows (see Figure 4.4):

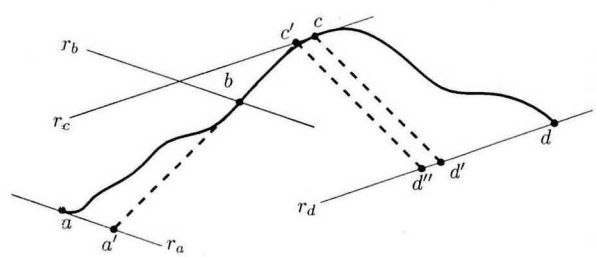

Figure 4.4: Speeding up pruned trajectory.

- consider the robot moving from c to r_d at full speed and let d' be the point where it reaches r_d
- consider the robot moving from b to r_a at full speeed back in time and let a' be the point where it reaches r_a (alternatively, a' is the point on r_a from where a robot moving at full speed towards r_b reaches b)
- finally, let c' be the point where the robot moving at full speed from b to r_c reaches r_c and let d'' be the point where the robot moving from c' at full speed towards r_d reaches it

As the support lines did not change, the trajectory $a'bc'd''$ is still valid. Note that a' is not earlier than a and d' is not later than d. Also, as c' is not later than c, then d'' is not later than d' and hence the resulting span $a'd''$ has not increased. □

Hence, it is sufficient to consider only trajectories consisting of at most three line segments of slope 1 and -1. In fact, it is sufficient to consider only one- and two-segment trajectories:

Lemma 14. *Let P be a valid pruned full-speed trajectory consisting of three segments of alternating directions. Then there exists a valid pruned full-speed trajectory of at most two segments with smaller span.*

Proof.

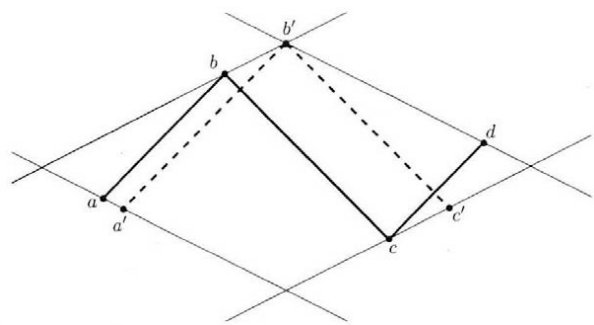

Figure 4.5: Optimizing a three-segment trajectory.

As P has three segments, it must change direction in each of its interior key points and each key point touches exactly one support line. Without loss of generality the first segment is of speed 1. As P changes direction whenever it touches a support line, the only possibility is shown in Figure 4.5. However, the trajectory $a'b'c'$ touches all four support lines and hence is valid, and its span is smaller than then span od P. □

Now we are ready to prove Statements 2 and 3. We need to consider only two strategies:

- **One segment:** the robot travels in one direction until it meets the bus, or
- **Two segments:** the robot travels in one direction for distance $2\pi/(1+s)$ (any two segment trajectory needs to cross from the starting support line to the opposing one; if it reverses sooner, it can only achieve the needed 2π separation between support lines by travel of $2\pi/(1-s)$ since reversal, at which point one segment strategy is better), then reverses and travels until it meets the bus

In the one segment strategy, the worst case time to meet the bus is $2\pi/(1-s)$, in the two segments strategy, it is $4\pi(1+s)$. The first is better for $s < \frac{1}{3}$, the latter for $s \in \langle \frac{1}{3}, 1 \rangle$, which proves Statements 2 and 3. This completes the lower bound in all three cases and proves Theorem 5. □

4.2.3 Robot knows neither the direction nor the speed of the bus

In this subsection we assume that the robot knows neither the direction of movement nor the speed of the bus.

Theorem 6. *If the robot knows neither the speed s of the bus being used nor its direction of movement then there is an algorithm with search time*

$$2\pi\left(1+\frac{1}{s+1}\right).$$

Moreover, for any $\epsilon > 0$, it is possible to assign a speed s to the bus such that no algorithm can achieve rendezvous with the bus in time less than $4\pi - \frac{\epsilon}{2}$.

Proof. (Theorem 6) Consider the following search algorithm.

Search Algorithm (Direction and Speed Unknown).

---

1. Choose a(ny) direction and walk for at most time 2π;
2. If no bus found then reverse direction and walk until bus is found;

To prove the upper bound, note that if the algorithm failed in time 2π to find the bus in Step 1 it is because it is traversing the cycle in the same direction as the bus. Therefore by reversing direction it is guaranteed that bus and robot are moving in opposite direction. Hence, they will meet in additional time $\frac{2\pi}{s+1}$, which proves the upper bound.

To prove the lower bound, first observe that for all s it must hold $t_a^s(0) - t_b^s(0) \geq 2\pi$ (i.e. the width of the strip of slope s encompassing the whole P must be at least 2π). Consider an s for which the width $w = t_a^s(0) - t_b^s(0)$ is minimized. Since w cannot be reduced by choosing infinitesimally smaller or larger s (i.e. rotating the strip), there must be three support points a, b, c placed on the tangents t_a^s and t_b^s such that the points alternate the tangent lines (two points, one each on the support lines for $s = 0$, can also define a strip of minimal width, but those points would need to have the same time coordinate, which is impossible, as P represents robot's trajectory). Without loss of generality assume a and c are on t_a^s and b is on t_b^s (see Figure 4.6).

Let P be an optimal trajectory. This means that $\forall \epsilon > 0$ there exists a bus speed

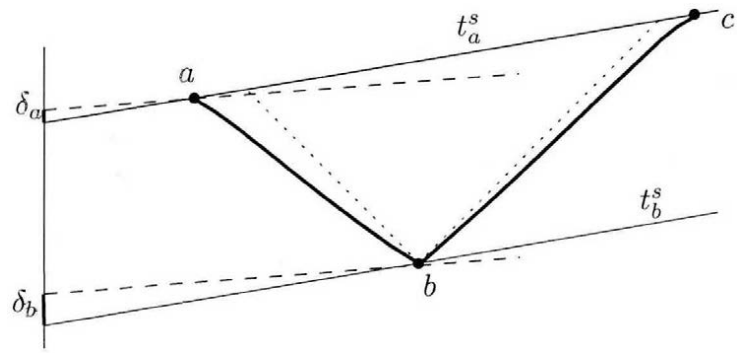

Figure 4.6: Lower bound for unknown bus speed. The dotted lines correspond to robot traveling at speed 1 to/from b. The dashed lines show slight rotation of the strip (if c was not on the top support line), yielding reduced width (as $\delta_a < \delta_b$). If a was not on the top support line, slight rotation in other direction will yeild smaller strip width.

and initial placement such that the robot meets the bus not earlier than ϵ before completing P, i.e. $|P| - \epsilon$ is a lower bound on the meeting time, where $|P|$ is the span of the trajectory P. Let us calculate how long will it take the robot to cross from a to b and then from b to c, as this is a lower bound on the span of P. It takes at least $2\pi/(1+s)$ to reach b from a, and at least $\frac{2\pi}{1-s}$ to reach c from b (or the other way around, depending on the direction of s and which support line has two points) i.e altogether

$$|P| = \frac{2\pi}{s+1} + \frac{2\pi}{1-s} = \frac{4\pi}{1-s^2}$$

We now indicate how to select the speeds. Note that by the geometric series expansion of $\frac{1}{1-s^2}$ we have that $\frac{4\pi}{1-s^2} = 4\pi + 4\pi s^2 + o(s^2)$. Therefore $|P| \leq 4\pi + 4\pi s^2 - \epsilon \leq 4\pi - \frac{\epsilon}{2}$ provided that $4\pi s^2 < \frac{\epsilon}{2}$ (i.e., $s < \sqrt{\frac{\epsilon}{8\pi}}$).

This completes the proof. □

4.3 Multiple Robots

In this section we consider the case of k robots. Throughout we assume that the bus is moving at constant speed s and cannot change direction, while the robots are moving with speed 1. Our analysis is divided into three subsections depending on the knowledge the robots have about the bus. In Subsection 4.3.1 we assume only that the robots know the direction of movement of the bus, in Subsection 4.3.2 the robots do not know the direction of movement of the bus but know its speed, while in Subsection 4.3.3 the robots know neither the direction nor the speed s of the bus.

4.3.1 Known direction of movement of the bus

In this subsection the robots know the direction of movement of the bus.

Theorem 7. *If the robots know the direction of movement of the bus then the search can be completed in time*

$$\frac{2\pi}{k(s+1)}.$$

This is optimal.

Proof. (Theorem 7) If the robots know the direction of movement of the bus, say CCW, then consider the following algorithm.

Search Algorithm (Direction Known).

---

1. The robots are initially placed on the perimeter of the cycle at distance $\frac{2\pi}{k}$ from each other.
2. The robots move in CW direction;

Observe that at the start, the bus is located within one of these k arcs delimited by the trajectories of the k robots. Therefore one of the robots will meet the bus in

its movement against the bus. It is clear that this algorithm takes time

$$\frac{2\pi}{k(s+1)}.$$

The proof of the lower bound $\frac{2\pi}{s+1}$ follows easily from the work of [109] and is omitted. This proves Theorem 7.

□

4.3.2 Unknown direction of movement but known speed of the bus

In this subsection we assume that the robots do not know the direction of movement but know the speed s of the bus.

Theorem 8. *If the robots know the speed s of the bus but do not know its direction of movement then the optimal search time is exactly*

1. $2\pi/ks$ *if* $s \geq 1$.
2. $2\pi/k$ *if* $s \leq 1$ *and* k *is even*
3. *Furthermore, if* $s \leq 1$ *and* k *is odd, then the lower bound is* $2\pi/(k-1)$ *and the upper bound is*
   (a) $\frac{2\pi}{ks}$ *for* $s \in \left(\frac{k}{k+1}, 1\right)$
   (b) $\frac{2\pi}{k-s}$ *for* $s \leq \frac{k}{k+1}$

Proof. (Theorem 8) We prove separately the upper and lower bounds for all cases in the statement of the theorem.

Upper bounds. First consider Statement 1. Assume $s \geq 1$. Consider the following algorithm.

Search Algorithm (Direction Unknown, Speed Known $s \geq 1$).

---

1. The robots are initially placed on the perimeter of the cycle at distance $\frac{2\pi}{k}$ from each other and wait motionless for the bus to arrive.

It is clear that the bus will meet one of the robots in time at most

$$\frac{2\pi}{ks}.$$

This upper bound is valid regardless of the parity of k.

Next consider Statement 2. Recall that in this case k is even. Assume $s \leq 1$. Consider the following algorithm.

Search Algorithm (Direction Unknown, Speed Known $s \leq 1$: k even).

---

1. The robots are initially placed in pairs on the perimeter of the cycle at distance $\frac{2\pi}{k}$ from each other;
2. The robots in each pair move in opposite directions.

For k even, the resulting distance between pairs is exactly $\frac{2\pi}{k/2} = \frac{4\pi}{k}$. Observe that the bus is located between two robots moving against each other. Since these two robots will meet no later than in time $\frac{4\pi}{2k} = \frac{2\pi}{k}$, the resulting running time for this algorithm will be

$$\frac{2\pi}{k}.$$

Finally, consider the case of k odd. We evaluate two algorithms and choose the best, depending on s. The first option is to use the algorithms for speed larger than 1, i.e. spread the agents evenly and wait until the bus meets you. The meeting time is $2\pi/ks$ in such case, The second option is to use the following algorithm:

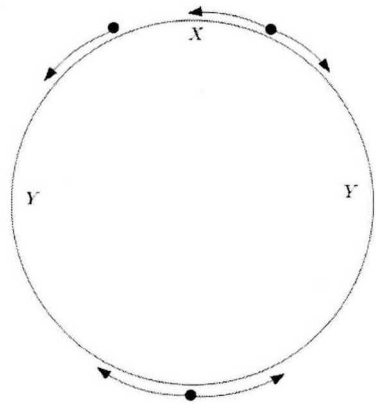

Figure 4.7: The algorithm for odd number of agents. Here $k = 5$.

Search Algorithm (Direction Unknown, Speed Known $s \leq 1$: k odd)

---

1. Let $X = \frac{2\pi(1-s)}{k-s}$ and $Y = \frac{2(2\pi-X)}{k-1}$.
2. $k+1$ robots are initially placed in pairs on the perimeter of the cycle at distance Y from each other; One robot at a node u neighbouring a segment of length X is then removed to bring down the number of robots used to k (see Figure 4.7)
3. The robots in each pair move in opposite direction, the lone robot moves away from the X segment.

Observe that if the bus started in a Y segment, two robots will be traveling towards each other from the opposite ends of this segment and will meet it at time at most $Y/2$. If the bus started in the X segment, the lone robot crossing the X segment will catch it in time at most $\frac{X}{1-s}$. X was selected so that these two times are equal:

$$T = \frac{Y}{2} = \frac{2\pi - X}{k-1} = \frac{\frac{2\pi(k-s)-2\pi(1-s)}{k-s}}{k-1}$$
$$= \frac{2\pi(k-1)}{(k-s)(k-1)} = \frac{2\pi}{k-s} = \frac{X}{1-s}$$

For $s > \frac{k}{k+1}$, the bound $2\pi/k$ of the waiting algorithm is better, while for $s < \frac{k}{k+1}$, this algorithm yields a better bound of $\frac{2\pi}{k-s}$.

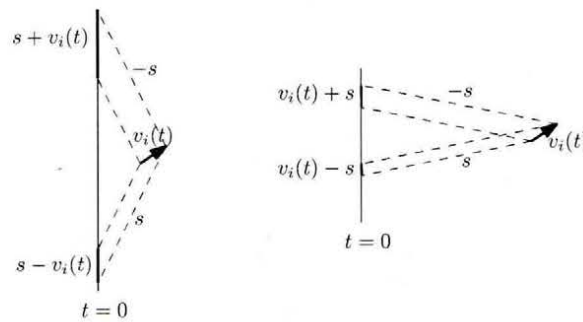

Figure 4.8: Sizes of excluded regions. Left: case $s \geq 1$. Right: case $s < 1$.

Lower bounds. First consider the lower bound in Statement 1. Assume that $s \geq 1$. Let $v_i(t)$ denote the speed of the i-th robot at time t as it is searching for the bus. Recall that always $v_i(t) \leq 1$ and hence also $v_i(t) \leq s$. Further, consider the movement of the robot at an arbitrary time dt. Let us express how much of the possible initial bus positions can the robot exclude in time dt, i.e. if the robot does not meet the bus at time interval dt, then the bus could not have started at those positions (see Figure 4.8). Summing up the size of exluded regions for both bus directions we obtain

$$(s + v_i(t))dt + (s - v_i(t))dt = 2sdt.$$

Let T be the time it takes for at least one of the robots to find the bus according to the execution of an optimal search algorithm. Therefore in time T, the i-th robot can cover at most (if at every time moment it excluded different regions) length

$$\int_0^T 2sdt = 2Ts$$

Thus, all k robots taken together can cover at most a length of $2Tks$, and only if all of them cover different areas. However, this last quantity must be at least 4π (2π for clockwise and another 2π for counterclockwise bus directions, otherwise there is a trajectory of the bus which will escape the robots' search). It follows that $2Tks \geq 4\pi$, which yields $T \geq 2\pi/ks$. This proves the lower bound in Statement 1.

Now consider the lower bound in Statement 2. Assume that $s \leq 1$. Let $v_i(t)$ denote the speed of the i-th robot at time t as it is searching for the bus. Using a similar argument we observe that the i-th robot in time dt covers a length equal to

$$(s + v_i(t))dt + (v_i(t) - s)dt = 2v_i(t)dt.$$

Therefore in time T, the i-th robot covers a length

$$\int_0^T 2v_i(t)dt \leq 2T,$$

where the last inequality is valid since the speed of the robot never exceeds 1, It follows that k robots can cover a length of at most $2Tk$. However, this last quantity must be at least 4π (otherwise there is a trajectory of the bus which will escape the robots' search). It follows that $2Tk \geq 4\pi$, which yields $T \geq 2\pi/k$. This proves the lower bound in Statement 2.

The lower bound for the case of $s < 1$ and odd k follows directly from the lower bound of Statement 2 by ignoring the last robot.

The proof of Theorem 8 is now complete.

□

Remark 1. *We note the open problem arising from the fact that the lower bound in Theorem 8 is not tight for k odd.*

4.3.3 Robots know neither the direction nor the speed of the bus

In this subsection we assume that the robots know neither the direction of movement nor the speed s of the bus.

Theorem 9. *If the robots know neither the direction of movement nor the speed of the bus then the search can be completed in time*

$$\frac{2\pi}{k} \text{ for } k \text{ even, and } \frac{2\pi}{k-1} \text{ for } k \text{ odd.}$$

Moreover, the lower bound $\frac{2\pi}{k}$ is valid regardless of the parity of k.

Proof. (Theorem 9) The algorithm below depends on the parity of k, the number of robots. First we look at the upper bound.

Assume k is even. Consider the following algorithm for k even.

Search Algorithm (Direction Unknown, Speed Unknown).

---

1. The robots are placed along the perimeter in pairs (two robots per position) and at consecutive distances $\frac{2\pi}{k/2} = \frac{4\pi}{k}$;
2. The two robots in each pair move in opposite directions;

As before this algorithm completes search in time $\frac{2\pi}{k}$.

Assume k is odd. The algorithm is the same as above by using only $k-1$ robots, which gives the upper bound $\frac{2\pi}{k-1}$.

The lower bound of $2\pi/k$ for k even and $2\pi/(k-1)$ for k odd follows from the lower bounds in Theorem 8 by setting $s = 0$ (in the case k is odd).

This completes the proof of Theorem 9.

□

The exact answer is not known for k odd. However it is conjectured that the search time for k robots should be the same as the search time for $k-1$ robots.

Remark 2. *We note the open problem arising from the fact that the lower bound in Theorem 9 is not tight for k odd.*

4.4 Additional Remarks and Conclusion

In this chapter we considered a search problem concerning k robots searching for a non-adversarial, uncooperative agent, called *bus*, which is moving with constant speed s along the perimeter of the cycle.

Chapter 5

Evacuation from a Disc in the Presence of a Faulty Robot

5.1 Introduction

Searching an environment to find an exit (or target) placed at an unknown location has been studied extensively in computer science and robotics. The searchers are autonomous robots which (may) cooperate during their search by exchanging messages so that at least one of them can find the target in minimum possible time. Another form of search recently introduced in [56] is called *evacuation* and it has the additional requirement that all the robots must go to the exit. Thus, optimality in evacuation is measured by the time it takes for the last robot to reach the exit, whereas in traditional search, optimality is measured by the time it takes the first robot to reach the exit.

In this chapter we consider an evacuation problem for three robots which are able to communicate wirelessly. Initially, the robots are located at the center of a disc of radius one and must find an exit located on the circumference of the disc and then

gather at the location of the exit. We consider two scenarios in which exactly one robot is faulty. In the first scenario, one robot can experience crash faults, which prevent it from either communicating or locating the exit. In the second scenario, one robot can experience Byzantine faults, which allow it to lie, e.g., to claim to have found an exit–where there is none– or even to fail to report (communicate) the location of the exit to the other robots. Note that the evacuation problem is considered to be solved when both non-faulty robots find the exit. For both scenarios, we provide upper and lower bounds.

5.1.1 Preliminaries and notation

There are three robots initially located at the center of a unit disc. The robots can move with maximum speed 1 (thus, they may stop or change direction at no cost), and are required to find an exit (whose location is unknown to the robots) located somewhere on the circumference of the disc and then gather at this location as fast as possible. A robot can find the exit only when it is in the same location as the exit. During their search the robots employ a *wireless communication model*, which means that they can exchange information instantaneously and at no cost and at any time, no matter the distance that separates them during their search.

The search problem to be studied is concerned with all non-faulty robots evacuating from the (unknown) exit. The search task is complicated by the fact that one of the three robots, chosen by an adversary, experiences faults, chosen by the adversary as well. We consider two scenarios. In the first scenario, the faulty robot experiences *crash* faults while in the second the robot experiences *Byzantine* faults. In both cases, the goal is to minimize the time till the last non-faulty robot reaches the exit.

- CRASH-EVACUATION: A *crash* fault can be thought of as a passive fault rending:

a robot is either unable or incapable to either detect or report the exit when it reaches it. Thus, such a robot is not expected to find the exit, only non-faulty robots can. However, we assume that in other aspects, a faulty robot moves like a non-faulty robot, and thus non-faulty robots cannot detect which robots are faulty.

- BYZANTINE-EVACUATION: A *Byzantine* faulty robot not only can fail to detect or report the target even after reaching it, it can also make malicious claims about having found the target when in fact it has not. Given the presence of such a faulty robot, the search for the target can only be concluded when the two non-faulty robots have sufficient verification that the target has been found.

All the messages being transmitted by the robots are tagged with the robot's unique identifier, which cannot be altered.

5.1.2 Related work

Searching an environment to find an exit placed at an unknown location is a well studied problem in computer science and robotics. The searchers are autonomous mobile robots that may also possess partial knowledge of their environment. Many researchers, starting with the seminal work of Bellman [20] and Beck [19], have studied the optimal (length) trajectory traced by a single robot when searching for a target placed at an unknown location on a line. The aim of the algorithmic designer is to minimize the competitive ratio, that is, the supremum, over all possible target locations, of the ratio between the distance traveled by the robot until it finds the exit, and the distance of the exit from the robot's starting position. For the case of a single robot on a line, the optimal trajectory uses a zig-zag, doubling strategy according to which if the robot fails to find the exit after travelling a certain distance

in a particular direction it returns to its starting position and doubles its searching distance in the opposite direction. This trajectory has a competitive ratio of 9 and this can be shown to be optimal (e.g., see Baeza-Yates et al. [12]).

Several authors considered the problem of searching in the two-dimensional plane by one or more searchers, including [11, 12]. The evacuation problem on a unit disc for multiple robots considered in our present work is a form of two-dimensional search that was first considered in [56]. In that paper the authors studied evacuation algorithms in the wireless and face-to-face communication models. New algorithms for the face-to-face communication model were subsequently analyzed for two robots in [61] and later in [32]. The problem has also been considered in other domains, like triangles and squares in [68]. However, all these papers concern evacuation only for non-faulty robots.

One of the novelties of our current work is that we consider the two-dimensional evacuation problem with fault tolerance. There are numerous studies of fault tolerance in distributed computing, (see, e.g., [142, 168, 176]). Network failures were most frequently related to static elements of the networked environment (i.e., nodes and links) as opposed to its mobile components. Malfunctions of this kind were sometimes modelled by dynamic alteration of the network [38, 164]. Distributed computation arising when having some of the mobile robots are faulty were investigated in the context of the problems of gathering [2, 79, 91, 191], convergence [31, 47], flocking [207], and patrolling [58]. Several researchers also investigated the case of unreliable or inaccurate robot sensing devices, e.g., [48, 145, 191]. Related to our study is also the research in [58], where a collection of robots, some of which are unreliable, perform efficient patrolling of a fence. Most relevant to our current study for its perspective on search and fault tolerance is the research of [67] and [60] which propose search algorithms for faulty robots that may suffer from crash and Byzantine faults, respec-

tively.

5.1.3 Outline and results

An outline of this chapter can be described as follows. Section 5.2 is dedicated to upper bounds. In Sections 5.2.1 and 5.2.2 we provide evacuation protocols along with their (worst case) analyses for the CRASH-EVACUATION problem and the BYZANTINE-EVACUATION problem, respectively. Then, in Section 5.3 we give lower bounds for both problems. Section 5.4 gives a discussion of possibilities for further research. The main results of the section are summarized in Table 5.1. Notably, since the

Problem	Lower Bound	Upper Bound
CRASH-EVACUATION	≈ 5.082 (Theorem 12)	≈ 6.309 (Theorem 10)
BYZANTINE-EVACUATION	≈ 5.948 (Theorem 12)	≈ 6.921 (Theorem 11)

Table 5.1: Comparison of Crash vs Byzantine: the first column gives the type of fault, the middle column lower bounds, and the right column upper bounds for the corresponding type of faults.

optimal offline algorithm for both problems CRASH-EVACUATION and BYZANTINE-EVACUATION would have the robots move directly to the exit at time 1, the time bounds of Table 5.1 can be also understood as bounds for the competitive ratio of the underlying online problems.

It is interesting to compare the results obtained in our work to the case of non-faulty robots. It is known (see [56]) that in the case of three *non-faulty* robots with wireless communication we have a lower bound of 4.159, and an upper bound.of 4.219 for evacuation, while for two non-faulty robots $1+2\pi/3+\sqrt{3} \approx 4.779$ is a tight upper and lower bound for evacuation.

5.2 Evacuation Protocols

In this section we propose evacuation algorithms for crash and Byzantine faults, respectively.

5.2.1 Evacuating with Crash-Faults

The main contribution is as follows.

Theorem 10. CRASH-EVACUATION *can be solved in time* ≈ 6.309.

We prove Theorem 10 by identifying the *best* among a special family of natural algorithms that we call *persistent*. These are algorithms that force all robots to immediately go to the circumference of the disc, and only allow a robot to stop exploring its segment of the disc (either by changing direction, by becoming idle or by leaving the circumference entirely) when it receives information about the exit. Since in this model, a faulty robot can only stay silent, any report about the exit has to be valid. As such, once the location of the exit is received by a robot, the robot moves along the shortest chord toward the reported exit, and evacuates.

We further classify persistent algorithms in two categories: the *symmetric-persistent* that have all the robots begin their exploration in the same direction (either all clockwise or all counter-clockwise), and the *asymmetric-persistent* that have one robot go in a direction, and the other two robots go in the opposite direction. It turns out that the best *asymmetric-persistent* algorithm outperforms the best *symmetric-persistent* algorithm (and also proves Theorem 10). Nevertheless, and as a warm-up, we begin by providing a tight analysis for the family of *symmetric-persistent* algorithms.

Lemma 15. *The best* symmetric-persistent *algorithm deploys the three robots at equidistant points on the disk (at arc-distance* $4\pi/3$*), and its performance is* $1 +$

$\frac{4\pi}{3} + \sqrt{3}$.

Proof. (Lemma 15) Consider a symmetric-persistent algorithm that deploys robots r_1, r_2, r_3 so that their pairwise anti-clock-wise distance is β, γ and α respectively, as also depicted in Figure 5.1 (where also arcs A, B, C are defined). Without loss of generality, assume the robots move in clockwise direction.

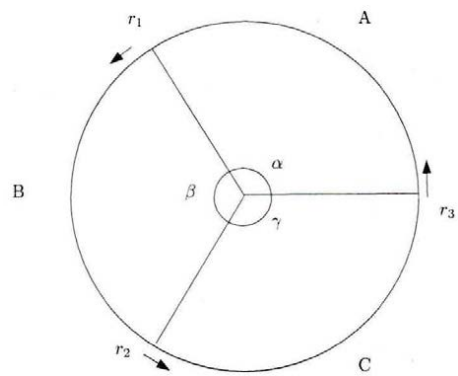

Figure 5.1: All robots move counter-clockwise. Arc A includes r_3 and excludes r_1; arc B includes r_1 and excludes r_2; and arc C includes r_2 and excludes r_3.

Consider the case where r_1 is faulty and the robots traverse the arcs depicted in Figure 5.1. Clearly, there are two cases to consider depending on whether the exit is located in one of the arcs A or B, or the exit is located on arc C. If the exit is located in one of the arcs A or B, then r_3 will discover it. If the exit is located in C, then r_2 will discover it. We say that the exit is either located at a counter-clockwise arc distance of $0 \leq x < \gamma$ from r_2 if r_2 discovers the exit, or a counter-clockwise arc distance of $0 \leq y < \alpha+\beta$ from r_3 if r_3 discovers the exit. Therefore, the total amount of time required to find the exit is given by the formula

$$1+\max\left\{\sup_{0\leq x<\gamma}\left(x+2\sin\frac{\gamma}{2}\right), \sup_{0\leq y<\alpha+\beta}\left(y+2\sin\frac{\alpha+\beta}{2}\right)\right\} = 1+\max\left\{f(\gamma), f(\alpha+\beta)\right\},$$

where we define $f(x) := x + 2\sin\frac{x}{2}$.

Similarly, if r_2 or r_3 is faulty, then the algorithm terminates in time $1+\max\{f(\gamma), f(\beta+\gamma)\}$ and $1+\max\{f(\beta), f(\alpha+\gamma)\}$ respectively. We conclude that the best symmetric-adaptive algorithm would choose α, β, γ (partitioning the perimeter of the circle, of length 2π) so as to minimize quantity

$$1+\max\{f(\alpha), f(\beta), f(\gamma), f(\alpha+\beta), f(\beta+\gamma), f(\alpha+\gamma),\} \tag{5.1}$$

By choosing $\alpha=\beta=\gamma=\frac{4\pi}{3}$, expression (5.1) gives completion time $1+\frac{4\pi}{3}+\sqrt{3}$ as promised.

Finally, we argue that no values of α, β and γ respecting α, β and $\gamma \geq 0$ and $\alpha+\beta+\gamma=2\pi$ can improve on this bound. Say, we set $\alpha > \frac{2\pi}{3}$. Then it is clear that either $\alpha+\beta > \frac{4\pi}{3}$ or $\alpha+\gamma > \frac{4\pi}{3}$, since $\alpha+\beta+\gamma=2\pi$. Observe that function $\alpha+\beta+2\sin\frac{\alpha+\beta}{2}$ is increasing in $\alpha+\beta$, and when $\alpha+\beta=\frac{4\pi}{3}$, then (5.1) is upper bounded by $1+\frac{4\pi}{3}+\sqrt{3}$. Observe also that function $\alpha+\gamma+2\sin\frac{\alpha+\gamma}{2}$ is increasing in $\alpha+\gamma$, and when $\alpha+\gamma=\frac{4\pi}{3}$, then expression (5.1) is upper bounded by $1+\frac{4\pi}{3}+\sqrt{3}$. We conclude that function (5.1) strictly increases for $\alpha > \frac{2\pi}{3}$. A similar argument shows that function (5.1) increases if either β or γ exceed $\frac{2\pi}{3}$. This completes the proof of Lemma 15. □ □

In order to proceed with the analysis of assymetric-persistent algorithms, we need a simple technical lemma, providing a worst case analysis for a special configuration of healthy searching robots.

Lemma 16. *Consider two robots at arc distance $2\pi - s$ that are about to explore an arc of length s moving in opposing directions (toward each other). Assume also that an exit is located somewhere at the arc of length s. Then, the worst case termination*

time $g(s)$ is given by the formula

$$g(s) = \begin{cases} 2\sin(s/2) & ,if\ s < 2\pi/3 \\ s/2 - \pi/3 + \sqrt{3} & ,otherwise. \end{cases}$$

Proof. (Lemma 16) By symmetry, we may assume that the exit is found after time x by one of the robots, where $0 \leq x \leq s/2$ (see Figure 5.2). When the message is transmitted that the exit is found, the two robots are at the endpoints of an arc of length $s-2x$, hence at chord distance $2\sin(s/2-x)$. Hence, the time elapsed till both

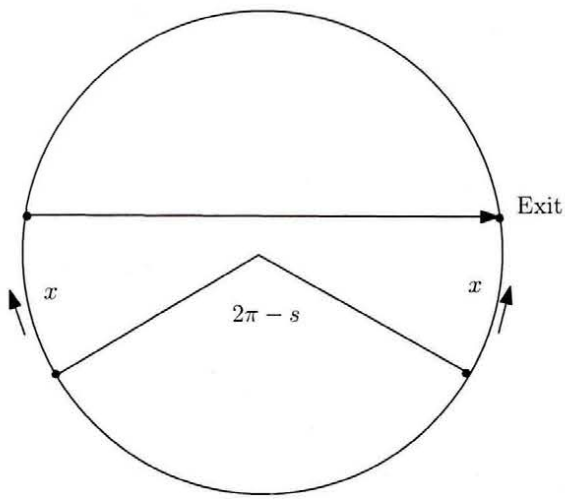

Figure 5.2: Exit found and reported after time x. Worst case is $x = 0$, if $s \leq 2\pi/3$, and $x = s/2 - \pi/3$ otherwise.

robots reach the exit is $x + 2\sin(s/2 - x)$. The claim follows by the monotonicity of the latest function with respect to x in the interval $[0, s/2]$. This completes the proof of Lemma 16.

□

We are now ready to prove Theorem 10, by determining the optimal asymmetric-persistent algorithm.

Lemma 17. *The best* asymmetric-persistent *algorithm has performance* ≈ 6.309. *The algorithm achieving this bound deploys two robots to the same location on the disc, which they explore in opposing directions. The third robot is deployed at arc-distance* β_0 *from any of the robots, and starts exploring in opposite direction of the closest robot, where* β_0 *is the unique root of* $3\beta/2 + \sqrt{3} = 4\pi/3 + 2\sin(\beta/2)$ *in the interval* $[0, 2\pi]$.

Proof. (Lemma 17) Consider an asymmetric-persistent algorithm that deploys robots r_1, r_2, r_3 as depicted in Figure 5.3, where $\alpha, \beta > 0$ (the case $\beta = 0$ can be easily seen to induce worse termination time, while the case $\alpha = 0$ is identical to $\gamma = 0$).

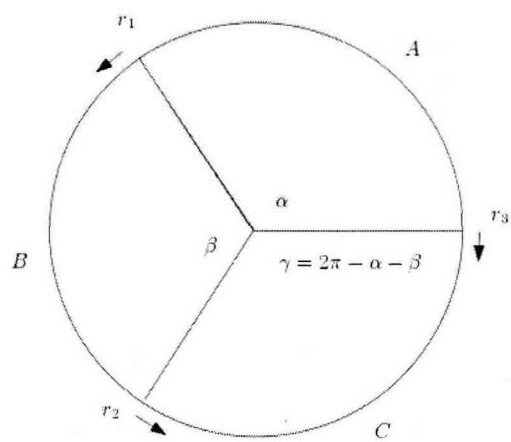

Figure 5.3: Robots r_1 and r_2 move counter-clockwise; r_3 moves clockwise. A excludes the starting position of r_1 and r_3; B excludes the starting position of r_2, but includes the starting position of r_1; C includes the starting position of both r_2 and r_3.

There are a number of cases as to which the faulty robot is and where the exit is located. All the cases are summarized in Table 5.2, where identical cases are also grouped together.

For each case we will determine the worst case running time. Then we will choose α, β, γ so as to minimize the maximum of all these running times.

- *Case 1.* After time γ, robots r_2, r_3 will be at arc distance γ and they will be

	A	B	C
r_1	Case 1	Case 1	Case 2
r_2	Case 3	Case 4	Case 4
r_3	Case 5	Case 6	Case 5

Table 5.2: The columns indicate the location of the exit. The rows indicate the faulty robot. r_1's initial search position is in B, r_2 and r_3's initial search position are in C.

about to explore an arc of length $\alpha+\beta = 2\pi-\gamma$ moving in opposing directions. Also the exit is located somewhere at the arc of length $2\pi - \gamma$. Hence, by Lemma 16, the (worst case) total termination time will be $1 + \gamma + g(2\pi - \gamma)$ which simplifies to

$$e(\gamma) := \begin{cases} 1+\gamma+2\sin(\gamma/2) & \text{,if } \gamma > 4\pi/3 \\ 1+\gamma/2+2\pi/3+\sqrt{3} & \text{,otherwise.} \end{cases}$$

Also, it is easy to see that $e(\gamma)$ is strictly increasing, a fact we will use later on.

- *Case 2.* The setup is identical to that of Lemma 16 where the arc that holds the exit has arc length $s = \gamma$. Hence, the (worst case) total termination time will be $1+g(\gamma)$, which is easily seen to be dominated by $e(\gamma)$ of case 1, for every $0 \leq \gamma \leq 2\pi$.

- *Case 3.* This situation is similar to Case 1, where (instead of γ) robots are at distance $\beta+\gamma$, and they are moving toward each other, and in an arc segment that does not contain the exit. Hence, the worst case termination time is equal to $e(\beta+\gamma)$. Since $e(\cdot)$ is strictly increasing, this case dominates the cost of case 1.

- *Case 4.* This situation is similar to Case 2, where (instead of γ) robots are at distance $\beta+\gamma$ and they are moving toward one another and toward the segment

that contains the exit. The maximal total required time is therefore given by the function $1 + g(\beta + \gamma)$, which is easily seen to be dominated by $e(\beta + \gamma)$ of case 3, for all $0 \leq \beta + \gamma \leq 2\pi$.

- *Case 5.* We treat the case when r_3 is faulty and the exit is either in C or A together. It is clear that r_2 will be the robot that finds the exit. Assume that the exit is located at distance $0 \leq x < \alpha + \gamma$ from the initial searching position of r_2 (to ensure that the exit is located in A). Then the total required search time is given by $1 + x + 2\sin\frac{\beta}{2}$, since the distance between r_1, r_2 remains invariant. Clearly, in the worst case, the total required search time is $1 + \alpha + \gamma + 2\sin\frac{\beta}{2}$.

- *Case 6.* This case is identical to case 5, where r_1 will find the exit (instead of r_2, but still β remains their invariant distance), and where the arc that contains the exit has length β (instead of $\alpha + \gamma$). Hence, worst case termination time is equal to $1 + \beta + 2\sin\frac{\beta}{2}$

It follows that the best asymmetric-persistent algorithm is determined by α, β, γ that minimize

$$\max\left\{e(\beta + \gamma), 1 + \alpha + \gamma + 2\sin(\beta/2), 1 + \beta + 2\sin(\beta/2)\right\},$$

i.e. the costs of cases 3, 5, and 6.

First we show that the promised upper bound is achievable. Indeed, we set $\gamma = 0$, so that $\alpha + \beta = 2\pi$. Now we define β_0, by equating the costs of cases 3,5, i.e. as the root of the equation $e(\beta) = 1 + 2\pi - \beta + 2\sin(\beta/2)$. Numerical calculations yield that $\beta_0 \approx 2.96603$, or in other words (by looking at the definition of function $e(\beta)$), β_0 is defined as the solution to the equation $3\beta/2 + \sqrt{3} = 4\pi/3 + 2\sin(\beta/2)$. We conclude that $\gamma = 2\pi - \beta_0 \approx 3.31716 < 4\pi/3$, which induces worst termination time to be the

same in cases 3,5 and equal to $1 + 2\pi - \beta_0 + 2\sin(\beta_0/2) \approx 6.30946$, as promised.

Now we prove the above choices are optimal. Indeed, if $\beta + \gamma > 4\pi/3$, then the total termination time cannot be better than the situation where cases 3,5 induce the same cost. Equating the resulting costs, we obtain that $\beta + \gamma + 2\sin((\beta+\gamma)/2) = \alpha + \gamma + 2\sin(\alpha/2)$. Using that $\beta + \gamma = 2\pi - \alpha$, the previous equation yields $\beta - 2\sin(\beta/2) = \alpha - 2\sin(\alpha/2)$, i.e that $\alpha = \beta$. But then $\gamma = 0$ as well. Since $\beta > 4\pi/3$, the induced cost, by case 3, is at least $1 + 4\pi/3 + \sqrt{3} \approx 6.92084$.

Finally, assume that $\beta + \gamma \leq 4\pi/3$. For any fixed γ, the total termination time cannot be better than the situation where cases 3,5 induce the same cost. Equating the resulting costs, we obtain that $(\beta+\gamma)/2 + 2\pi/3 + \sqrt{3} = \alpha + \gamma + 2\sin(\beta/2)$. Since $\alpha = 2\pi - \beta - \gamma$, the optimal choice for β should be β_γ satisfying $3\beta_\gamma/2 + \gamma/2 + \sqrt{3} = 4\pi/3 + 2\sin(\beta_\gamma/2)$. Note that β_γ is a function of γ, hence differentiating both sides of last equation with respect to γ, and after elementary calculations, we obtain that $\beta'_\gamma(3/2 - \cos(\beta_\gamma/2)) = -1/2$. Since $\beta_\gamma > 0$, we obtain that $\cos(\beta_\gamma/2) < 1$ and hence $\beta'_\gamma > -1$. This implies that expression $\beta_\gamma + \gamma$ is strictly increasing in γ, and this linear term appear in the termination time of case 3. Hence, choosing $\gamma = 0$ is indeed optimal. This concludes the proof of Lemma 17. □ □

5.2.2 Evacuating in the presence of Byzantine Faults

The main contribution is as follows.

Theorem 11. BYZANTINE-EVACUATION *can be solved in time* $1 + \frac{4\pi}{3} + \sqrt{3} \approx 6.92084$.

Proof. (Theorem 11) The analysis relies on Figure 5.4. Assume that all three robots r_k, for $k \in \{1, 2, 3\}$, execute the main evacuation Algorithm 3.

The idea of the algorithm is for the robots to traverse the circumference of the disk for a time of $2\pi/3$. Depending on the calls that have been received, the robots

have information to either go to the exit or continue traversing the circumference of the disk for another period of time time $\frac{2\pi}{3}$. They can now verify conflicting messages of the correct location of the exit based on the calls that have been made by the other robots so far. Details are being discussed in the sequel.

Algorithm 3: Evacuation with Byzantine Faults

```
1  Go to the circumference, at position 2πk/3;
2  while r_k's location is not the same as the exit's location do
3      for 2π/3 do
4          follow the circumference clockwise
5      if One robot claims to have found more than one exit then
6          Continue execution of algorithm as though the robot remained silent
7      if No information about exit then
8          for 2π/3 do
9              follow the circumference clockwise till exit is either found or
                reported. Finish
10     if One robot claims to have found the exit then
11         Go to closest part of the segment that is claimed to contain the exit;
12         Explore entire segment. Finish.
13     if Two robots claim to have found the exit then
14         Investigate both exits. Finish.
15 Inform all robots of the location of the exit.
```

First note that one time unit is required to reach the circumference of the disc. After $\frac{2\pi}{3}$ additional time units, the entire disc has been explored once. The areas explored by the robots are contiguous but not overlapping. Observe that a Byzantine robot that claims to have found more than one exit is immediately identified as faulty by the healthy robots. Both potential exits are ignored, and the algorithm continues as though the robot had remained silent. If a non-faulty robot finds the exit, it immediately informs all other robots, then stop its exploration. Say without loss of generality that r_1 is healthy. If r_1 finds the exit during the first $\frac{2\pi}{3}$ part of the exploration, then it stops and is done with the execution of its algorithm, in a time

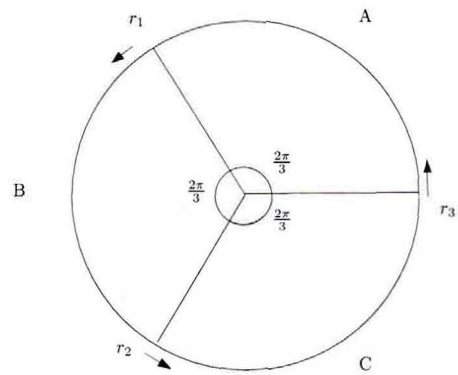

Figure 5.4: The initial searching position for r_1, r_2 and r_3 in the Byzantine faults model

at most $1 + \frac{2\pi}{3}$. If it does not find an exit during the first $\frac{2\pi}{3}$ part of the exploration, then we must consider three cases:

- *No exit location reported:* If no exit was found, then keep exploring the circumference of the disk for time $\frac{2\pi}{3}$. Notice that this means that the exit cannot be in B. If the exit is in C, then r_1 has found the exit, and its execution is complete in a time at most $1 + \frac{4\pi}{3}$. If the exit is in A, then we learn that r_3 is Byzantine (otherwise, it would have claimed to have found the exit during the first $1 + \frac{2\pi}{3}$ of the execution of the algorithm), and r_2 will have correctly identified the location of the exit (Notice that r_1 needs to finish exploring the second arc C to make sure that it was r3 that lied.) Say the exit is located at an arc distance of $0 < x < \frac{2\pi}{3}$ from r_1's current position. Then $2\sin\frac{x}{2}$ is required for r_1 to reach the exit. Since this function is monotone in x for $x \leq \pi$, r_1 can reach the exit in a total time of at most $1 + \frac{4\pi}{3} + \sqrt{3}$.

- *One exit location reported:* If one robot other than r_1 claims to have found the exit, we consider two situations: (1) the robot is healthy, in which case the exit is indeed located on the segment where the announcement was made; or (2)

the robot is Byzantine, in which case the other two segments have been entirely explored by healthy robots (and are therefore reliably proven to be empty), and the exit is located on the segment where the announcement was made. Notice that in both situations, the only possible location for the exit is on the segment where the announcement was made. If the announcement was made on the segment C, then r_1 explores C immediately, for a total time of at most $1 + \frac{4\pi}{3}$. If the announcement was made on the segment A, then r_1 must first reach one end of segment A, which requires $2\sin\frac{2\pi}{3} = \sqrt{3}$ (both ends of the segment are equidistant from r_1's position), then explore the segment, for a total time of at most $1 + \frac{4\pi}{3} + \sqrt{3}$.

- *Two exit locations reported:* If both r_2 and r_3 claim to have found an exit, then we know that one of those two claims is true. r_1 will investigate both claims, starting by the closest one. Say r_2 claims to have found the exit at a distance x from its initial searching position, and r_3 claims to have found the exit at a distance y of its initial searching position. Then r_1 must travel an additional $2\sin\frac{x}{2} + 2\sin\frac{\frac{2\pi}{3} - x + y}{2}$ to reach both exits. This function is maximised for $x = y = \frac{2\pi}{3}$, for a total time of at most $1 + \frac{2\pi}{3} + 2\sqrt{3}$.

Observe that both robots r_2 and r_3 execute the same algorithm, and the maximal time required is therefore the same. The adversary will choose the location of the exit and the Byzantine robot in such way as to maximise the total time of execution of the algorithm. Therefore, since $\sqrt{3} < \frac{2\pi}{3}$, this algorithm solves the evacuation problem in total time $1 + \frac{4\pi}{3} + \sqrt{3}$. This completes the proof of Theorem 11. □ □

5.3 Lower Bounds for Evacuation Protocols

This section is devoted to proving our main negative results.

Theorem 12. *The following lower bounds are valid.*

(a) Problem CRASH-EVACUATION *requires time at least* 5.082.

(b) Problem BYZANTINE-EVACUATION *requires time at least* 5.948.

The lower bound proofs for Crash and Byzantine faults, respectively, admit a unified approach that we detail in the form of a few preliminary lemmata below.

It is easy to observe that if we consider three robots starting from the center of a unit disc then for any $\epsilon > 0$, at time $1 + \frac{2\pi}{3} - \epsilon$ there is an equilateral triangle inscribed in the circle not all of whose vertices have been explored by a robot. However, in the main proof we will make use of an even stronger property of the three robots.

Next we define a useful property $P(T)$, where $T > 0$ denotes time, to be used in the rest of the proof for a lower bound.

Definition 1 (Property $P(T)$). *For any algorithm and any time less than T there are two points on the circle at distance at least $\sqrt{3}$ and each of which was visited at most once by anyone of the three robots.*

Since Property $P(T)$ ensures the existence of two points at distance at least $\sqrt{3}$ which have been visited at most once by the robots, a simple adversarial argument will guarantee that $T + \sqrt{3}$ is a lower bound on evacuation for Byzantine faults (see Lemma 20), while $T + \sqrt{3}/2$ is a lower on evacuation for Crash faults (see Lemma 19). However, before proving these last statements, we are interested to find a T which satisfies property $P(T)$.

Note that property $P(T)$ is monotone increasing in T, in that $P(T) \wedge T' \leq T \Rightarrow P(T')$. Hence, the larger the value of the parameter T for which $P(T)$ is valid the better the lower bound that can be derived.

Lemma 18. *Property* $P(1 + 13\sqrt{3}/7)$ *is valid.*

Proof. (Lemma 18) In the sequel, to help our intuition, we prove first the weaker statement that $P(4)$ is valid and then we improve this to $P(1 + 13\sqrt{3}/7)$. Let us consider some algorithm at time $< T$, where $T = 4$, and assume by contradiction that all points that have been visited at most once by a robot are at distance less than $\sqrt{3}$ from each other. Clearly, all these points must lie on an arc of length less than $2\pi/3$. Therefore looking at the complement of this arc we find an arc of length longer than $4\pi/3$. In turn, this gives rise to a regular hexagon with five of its vertices inside this last arc each visited twice by a robot. Therefore these five vertices of the hexagon have been visited ten times in total by the three robots. Since there are three robots, it follows that at least one robot must have visited four of these vertices. However this is impossible as $T = 4$. It follows that property $P(4)$ is valid.

Now we derive the main result of the lemma by showing that $P(1 + 13\sqrt{3}/7)$ is valid. We argue as in the previous paragraph, however, instead of selecting five vertices of a regular hexagon we will choose the five points more carefully.

As in the proof of $P(4)$ above, let three points A, B, C be vertices of an equilateral triangle such that every point in the perimeter of the disc which is visited by at most one of the three robots is in the arc clockwise between A and B.

In turn, this will give rise to five points on the circumference of the disc with each of its vertices visited twice by a robot; namely choose a point D located between A and C and a point E between B and C so that the length of arc AD is x and this is equal to the length of arc EB (the choice of x will be based on maximizing the length

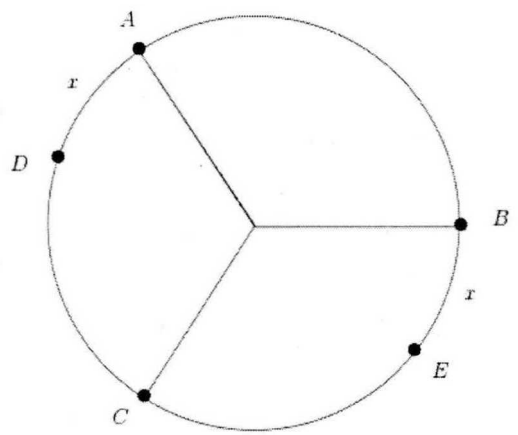

Figure 5.5: Evacuation of the second truth telling robot.

of a path visiting these vertices and will be made precise in the next paragraph). Since there are ten visitations by three robots one of the robots must have visited four consecutive points at least once.

We will show that visiting four vertices among A, B, C, D, E takes time at least $13\sqrt{3}/7 \approx 3.21$. If $x < \pi/3$ then there are 2 candidates for the shortest four-point walk, namely

$$\text{either } D \to A \to B \to E \text{ or } A \to D \to C \to E.$$

Taking into account the lengths of the corresponding chords in these two paths, it turns out that we need to maximize the function $f(x)$ defined by the equation below.

$$f(x) := \min\{\sqrt{3} + 4\sin(x/2), 2\sin(x/2) + 4\sin(\pi/3 - x/2)\}.$$

It is easily seen that the maximum of f is equal to $1 + 13\sqrt{3}/7$ and it is obtained at $x = 4/\arctan(1/(3\sqrt{3}))$. The rest of the reasoning is the same as for $T = 4$ in the first paragraph of the proof. This completes the proof of Lemma 18. □ □

Proof. (Theorem 12) Now we are ready to conclude the proofs of the two parts of

Theorem 12 on crash and Byzantine faults, respectively.

Lower Bound for Crash-Faults The proof of Part (a) follows as a corollary of Lemma 19 below.

Lemma 19. *If property $P(T)$ holds then we can achieve a lower bound of $T + \frac{\sqrt{3}}{2}$ on evacuation in the presence of a crash-faulty robot.*

Proof. (Lemma 19) Identify two points A, B at distance $\geq \sqrt{3}$ each of which was visited at most once by anyone of the three robots. Say r_1 is the robot that visited neither of those points. Set the exit to be the point farthest away from r_1's current location. Clearly, at least $\frac{\sqrt{3}}{2}$ is required for r_1 to reach the point. This proves lemma 19.

□

Lower Bound for Byzantine-Faults The proof of Part (b) follows as a corollary of Lemma 20 below.

Lemma 20. *If property $P(T)$ holds then we can achieve a lower bound of $T + \sqrt{3}$ on evacuation in the presence of a Byzantine robot.*

Proof. (Lemma 20) Identify two points A, B at distance $\geq \sqrt{3}$ each of which was visited at most once by anyone of the three robots. Assume without loss of generality that r_1 visited A. Then we have two possibilities to consider: either r_1 also visited B, or (say) r_2 visited B.

If r_1 visited both points, set r_1 to be Byzantine, then wait until either r_2 or r_3 visit either A or B. Once this first visit happens, claim that the exit is located at the other point. The robot that visited the first point will require at least $\sqrt{3}$ to reach the other point, which proves the lemma in this case.

If, say, r_2 visited point B, then have r_1 claim that the exit is located at point B, and r_2 claim that the exit is located at point A (which will happen as soon as the robots reach those points). Then r_3 will have to visit both points to find the real exit, since it has no means of distinguishing the reliable robot from the Byzantine robot. Choose the first point visited by robot r_3 not to have the exit, and set the exit at the location of the other point. Then r_3 requires at least $\sqrt{3}$ to reach the other point, which proves the lemma in this case as well.

Combining these two cases, this completes the proof of Lemma 20.

□

If we note the following approximations for the quantities arising in Lemma 18: $1 + 13\sqrt{3}/7 \approx 4.21$ and $4/\arctan(1/(3\sqrt{3})) \approx 0.76$, then the proof of Theorem 12 is complete. □ □

5.4 Additional Remarks and Conclusion

In this chapter we considered the evacuation problem on a disc for three robots exactly one of which has either crash or Byzantine faults. We analyzed the problem in both fault scenarios and gave lower bounds as well as evacuation algorithms resulting in upper bounds.

Chapter 6

Exploring Graphs with Time Constraints by Unreliable Collection of Mobile Robots

6.1 Introduction

Alice and Bob is a busy Ottawa couple with three kids Chris, Donald and Elsa. One day they need to pick up Elsa from the kindergarten, drive Donald to the wrestling practice and get Chris to the train station. They also need to get groceries, pick up wine and flowers before each store closes for a dinner party in their house. How should Alice and Bob share these tasks to minimize the effort and complete each one before its deadline?

An Ottawa School Bus Company needs to transport pupils to local schools before the start of their classes. Given the harsh Canadian climate, it is the norm rather than exception that a number of buses fail to function on any given day and an adequate replacement must be planned in advance. How should the buses allocate the tasks

so as to successfully conclude the distribution of students while respecting the time deadlines?

Throughout this chapter, the environment is modelled by a graph that must be explored by a collection of mobile robots. The graph edges are weighted by numbers, representing the time it takes to traverse them. Each graph node is assigned a deadline, representing the maximal time moment to deliver a service to this node by some mobile robot. A number of robots may crash during their work. What is the minimal time needed to service a given graph by a collection of k robots? What is such a time if we assume that up to f unknown robots may crash during their work?

6.1.1 Preliminaries and notation

We are given a weighted n-node graph $G = (V, E)$ with V its set of vertices, E its set of edges, and a set of k mobile robots initially placed at a subset of its nodes. The weight of an edge $\{v_i, v_j\}$ corresponds to the time it takes to be traversed by a robot. Each node v_i of the graph is assigned a deadline Δ_i, which is a positive real number. Robots walk along the edges of the graph with unit speed. The robots collaborate attempting to explore the entire graph. However, a subset of up to f robots may turn out to be unreliable and fail to collaborate. Unreliability refers to the robots which may be crash faulty in that they suffer from an (unspecified) passive, omission failure and then stop responding but are otherwise harmless. This subset of unreliable robots may be chosen by the adversary, which is assumed to know our algorithm beforehand. The exploration is successful if each graph node is visited before its deadline by at least one of the reliable robots.

We assume that nodes already explored "do not block passage" and can still be visited, even after their deadlines have expired, by robots on their way to reaching

unexplored parts of the graph.

We denote by $t \to r_i(t)$ the trajectory of the i-th robot as a function of the time t, where $r_i(t)$ denotes the position of the i-th robot in the graph at time t, for $i = 1, 2, \ldots, k$. Note that at a given time t, a robot may be located in the interior of an edge.

By a schedule we mean a set of functions $r_i(t), i = 1, 2, \ldots, k$ which define the motion of the robots respecting their maximum unit speed. We say that the schedule *explores* the graph if for each node v_i there exists a robot r_j such that $r_j(t^*) = v_i$, for some time $t^* \leq \Delta_i$.

Given a time Δ, we study the *decision problem* whether the graph may be successfully explored before time Δ. We also look at the *optimization problem*, that is, the problem of ensuring that the reliable robots visit every node before expiration of its deadline, and the last explored node is visited as fast as possible. If for any schedule, the adversary can find a subset of f unreliable robots, so that any of the remaining $k - f$ robots fails to visit some node before its deadline, then the instance of the problem is deemed unsolvable.

6.1.2 Related work

Searching a graph with one or more searchers has been widely studied in the mathematics literature (see, e.g. [125] for a survey). There is extensive literature on linear search (referring to searching a line in the continuous or discrete model), e.g., see [11] for optimal deterministic linear search and [81] for algorithms incorporating a *turn cost* when a robot changes direction during the search. Variants of search using collections of *collaborating* robots has also been investigated. The robots can employ either *wireless* communication (at any distance) or *face-to-face* communication, where

communication is only possible among co-located robots. For example, the problem of *evacuation* [61] is essentially a search problem where search is completed only when the target is reached by the last robot. Linear group search in the face-to-face communication model has also been studied with robots that either operate at the same speed or with a pair of robots having distinct maximal speeds [15, 44]. Linear search with multiple robots where some fraction of the robots may exhibit either *crash faults* or *Byzantine faults* is studied in [67] and [60], respectively.

The (Directed) Rural Postman Problem (DRPP) is a general case of the Chinese Postman Problem where a subset of the set of arcs of a given (directed) graph is 'required' to be traversed at minimum cost. [43] presents a branch and bound algorithm for the exact solution of the DRPP based on bounds computed from Lagrangian Relaxation. [52] studies the polyhedron associated with the Rural Postman Problem and characterizes its facial structure. [100] gives a survey of the directed and undirected rural postman problem and also discusses applications.

A scheduling problem considered by the research community concerns n jobs, each to be processed by a single machine, subject to arbitrary given precedence constraints; associated with each job j is a known processing time a_j and a monotone nondecreasing cost function $c_j(t)$, giving the cost that is incurred by the completion of that job at time t. [170] gives an efficient computational procedure for the problem of finding a sequence which will minimize the maximum of the incurred costs. Further, [170] also studies a class of time-constrained vehicle routing and scheduling problems that may be encountered in several transportation/ distribution environments. In the single-vehicle scheduling problem with time window constraints, a vehicle has to visit a set of sites on a graph, and each site must be visited after its ready time but no later than its deadline. [208] studies the problem of minimizing the total time taken to visit all sites. [135] considers the problem of determining whether there exists

a schedule on two identical processors that executes each task in the time interval between its start-time and deadline and presents an $O(n^3)$ algorithm that constructs such a schedule whenever one exists.

The author of [25] resolves the complexity status of the well-known Traveling Repairman Problem on a line (Line-TRP) with general processing times at the request locations and deadline restrictions by showing that it is strongly NP-complete. [178] considers the problem of finding a lower and an upper bound for the minimum number of vehicles needed to serve all locations of the multiple traveling salesman problem with time windows in two types of precedence graphs: the start-time precedence graph and the end-time precedence graph. [141] considers "the pinwheel", a formalization of a scheduling problem arising in satellite transmissions whereby a piece of information is transmitted for a set duration, then the satellite proceeds with another piece of information while a ground station receiving from several such satellites and wishing to avoid data loss faces a real-time scheduling problem on whether a "useful" representation of the corresponding schedule exists.

The work of [198] is very related to our work in that jobs are located on a line. Each job has an associated processing time, and whose execution has to start within a prespecified time window. The paper considers the problems of minimizing (a) the time by which all jobs are executed (traveling salesman problem), and (b) the sum of the waiting times of the jobs (traveling repairman problem). Also related is the research on Graphs with dynamically evolving links (also known as time varying graphs) which has been explored extensively in theoretical computer science (e.g., see [38, 113, 164]).

6.1.3 Outline and results

We consider first the collections of robots which are all reliable. We start in Section 6.2 with the case of a single robot on a line graph and we give an algorithm finding the shortest exploration time when the robot's starting position is given, is arbitrary, or it is arbitrary but restrained to some subset of line nodes. In Section 6.3 we study line exploration by a collection of robots at fixed or arbitrary positions on the line. We observe, that these algorithms may be extended to the ring case, although their complexity is slightly compromised.

In Section 6.4 we consider the case of unreliable robots. In one case, we show an unexpected result. If k robots are at given fixed initial positions on the line and up to f out of k robots may turn out to be crash-faulty, the problem of finding the optimal exploration time is NP-hard. This result holds even if the nodes' deadlines may be ignored (e.g. they are infinite for all nodes). For all other settings we give algorithms finding optimal exploration times. In Section 6.5 we extend our approach to the ring environment. However, the setting which was proven to be NP-hard for lines is polynomial-time decidable for the ring. Finally, we show that outside the line and ring environment the problem becomes hard. For a graph as simple as a star, already for the case of two robots, the exploration problem turns out to be NP-complete.

6.2 Single Robot on the Line

In this section, we present algorithms that allow a single robot to solve the optimization problem on the line for two cases: when the robot's initial position is given by an adversary, and when we have the possibility of choosing it ourselves.

We have a sequence of nodes $v_0 < v_1 < \cdots < v_{n-1}$ on the real line, and a robot r initially placed at initial position $r(0)$. We denote by v_s the starting node of the

robot, i.e. $r(0) = v_s$.

Observation 1. *Without loss of generality we may assume that* $\Delta_{s+1} < \Delta_{s+2} < \cdots < \Delta_{n-1}$. *Indeed, if* $\Delta_k \geq \Delta_{k+1}$ *for some* $k > s$ *we can drop node* v_k *from consideration, since visiting* v_{k+1} *before its deadline implies that* v_k *is also visited before its deadline. For the same reason, we can also assume that* $\Delta_0 > \Delta_1 > \cdots > \Delta_{s-1}$.

Observation 2. *Without loss of generality we may consider only the solutions which consist of sequences that are increasing and decreasing at alternate nodes, respectively, i.e., sequences* $r(0), r(t_1), r(t_2), \ldots, r(t_p)$ *such that* $0 \leq r(t_{2i}) < r(t_{2i+2})$, *and* $0 \geq r(t_{2i+1}) > r(t_{2i+3})$, *for all* i *in the appropriate range. Moreover, each turning node* $r(t_i)$ *is located at some node* $v_j, j = 0, 1, \ldots, n-1$.

6.2.1 The snapshot graph

With these observations in mind, we define the fundamental concept of a directed, layered *snapshot graph* S which will form the basis of all subsequent algorithms.

Every node of the snapshot graph S represents a situation when a new node of the line is visited by the robot for the first time. Consequently, each node of S is denoted by a pair $(i, \bar{j})$ or $(\bar{i}, j)$, where $i \leq j$, $[i, j]$ is the interval of nodes already explored by the robot and the node of the line marked with the bar (either $\bar{i}$ or $\bar{j}$) denotes the current position of the robot.

Observe that the robot can advance its exploration in one of two ways: either by visiting the next unexplored node to the left of the interval already explored, or by visiting the next unexplored node to its right. These two possibilities generate the directed edges between the nodes of the snapshot graph. The weight of such an edge equals the time needed by the robot to traverse the path between robot positions in both nodes. Consequently, the nodes $(i, \bar{j})$ and $(\bar{i}, j)$ are placed at layer $j - i$ and the

adjacencies in S are only between nodes of consecutive layers. Notice the following properties of the snapshot graph (see also Fig. 6.1 below):

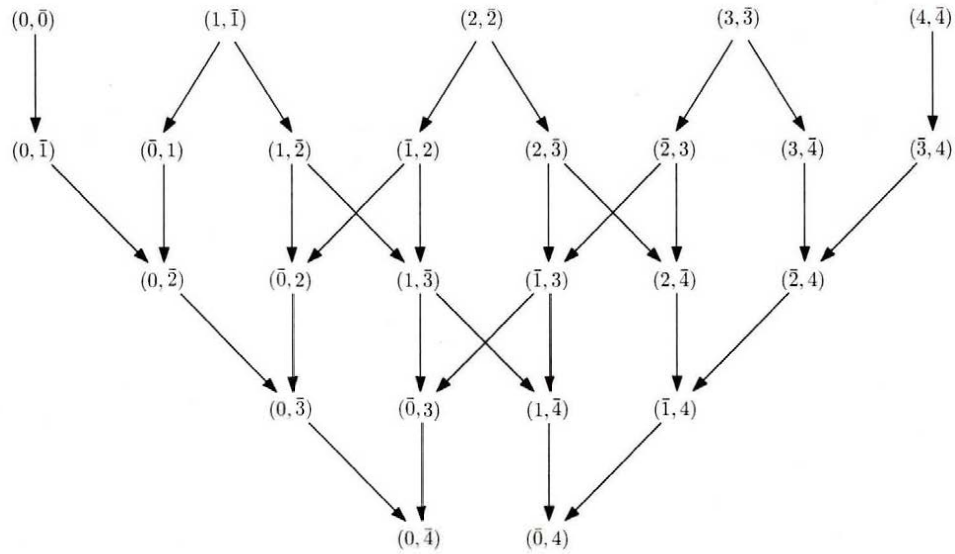

Figure 6.1: A depiction of a snapshot graph for the case of line L consisting of five nodes. For clarity, we do not show the edge weights of the snapshot graph S in Fig. 6.1. Notice that, for any line graph L, the weights of the directed edges $(i,\bar{j}) \to (i,\overline{j+1})$ and $(\bar{j},k) \to (\overline{j+1},k)$ in its snapshot graph are equal to the weight of the edge $(j,j+1)$ in the line graph L. Similarly, the weights of the directed edges $(i-1,\bar{j}) \to (\bar{i},j)$ and $(\bar{i},j-1) \to (i,\bar{j})$ in the snapshot graph S are equal to the weight of the path $i \rightsquigarrow j$ in the line graph L.

- The graph S has n layers numbered from 0 to $n-1$.
- There are n *source* nodes at the zeroth layer and $2(n-j)$ nodes at the j-th layer for each $j = 1, 2, \cdots, n-1$. Therefore, there are 2 *target* nodes (on the $(n-1)$-th target layer).
- The in-degree and the out-degree of each node is bounded by 2. Therefore, the number of nodes in the graph is bounded by snapshot graph is $O(n^2)$.

Observe that, the solution to the optimization problem for the line corresponds to

the shortest path from the source node representing the initial position of the robot to one of its target nodes, which respects the time constraints of all the nodes of L.

6.2.2 Given initial position of the robot

We first present a version of the algorithm which produces the optimal exploration path, assuming a given starting position v_s of the robot on the line. Consider the snapshot graph S described above. In order to obtain the optimal exploration path in the snapshot graph respecting the time constraints of L, we generate an all-targets shortest-time tree T whose root coincides with the node $(v_s, \bar{v_s})$ of the snapshot graph corresponding to the initial position v_s of the robot. This is done in the following way.

We add a time counter *time* to every node of S. We set to zero the time counter of the initial node $(v_s, \bar{v_s})$ and to ∞ the initial time counters of all other nodes of S. We then visit all nodes of S layer by layer. Consider a visit of any such node v, which corresponds to the first visit to node v_j of L. For each predecessor of v in S we consider the time equaling its time counter augmented by the weight of the edge joining it with v. Let Min denote the smaller of these values (we take an arbitrary one in the case of equality). If Min does not exceed the time constraint of v_j (i.e. $Min \leq \Delta_j$) we set the time constraint of v to Min and we add to T the edge from the corresponding predecessor of v. Otherwise, the time counter of v is set to ∞ and we leave v parentless.

Observe that, T is a tree, as each node has at most one parent. One of the two target nodes of the smaller time counter defines the optimal exploration time and the path to it in T corresponds to an optimal exploration path of L. Otherwise, there exists no exploration path respecting the node deadlines of the line graph.

For any node v of S we denote by $new(v)$ the index of the node of the line G which is newly explored when arriving at v. More exactly, $new(v) = j$, such that either $v = (i, \bar{j})$ or $v = (\bar{j}, k)$, for some $i \leq j \leq k \leq n-1$.

The following procedure InitStart indicates how to initialize the time counters of the nodes of S before running the main body of the algorithm. For each node i of the line L, which may be a starting position of a robot, we put a node $(i, \bar{i})$ of S to the set A. All nodes of A have their time counters initialized to 0.

Procedure InitStart(A, S) with A a subset of nodes of S at zeroth layer;

```
for every node v of V(S) \ A do
    time(v) = ∞;
for every node v of A do
    time(v) = 0;
```

Algorithm 4 describes the pseudo-code that formalizes the previously outlined construction of a shortest-time tree.

Algorithm 4: Single Robot exploration on the line with given initial position v_s;

Input: A snapshot graph S and the starting position v_s of the robot
Output: An exploration tree with optimal exploration times

```
InitStart({v_s}, S);
for layer i = 0 to n − 1 do
    for each arc v → w starting at layer i do
        t = time(v) + weight(v, w);
        if t < time(w) and t ≤ Δ_new(w) then
            time(w) = t; v = parent(w);
```

Figure 6.2 illustrates the execution of Algorithm 4. The weighted line graph containing five nodes denoted by integers from 0 to 4 is presented at the top of Fig. 6.2. The robot is initially placed at node 1. The solid directed edges depict the shortest-time tree respecting the deadlines (the remaining edges of the snapshot

graph which are not being used are dashed). Each node has been assigned the time counter computed by Algorithm 4. The path of the shortest-time tree ending in the target node represents the optimal trajectory of the robot.

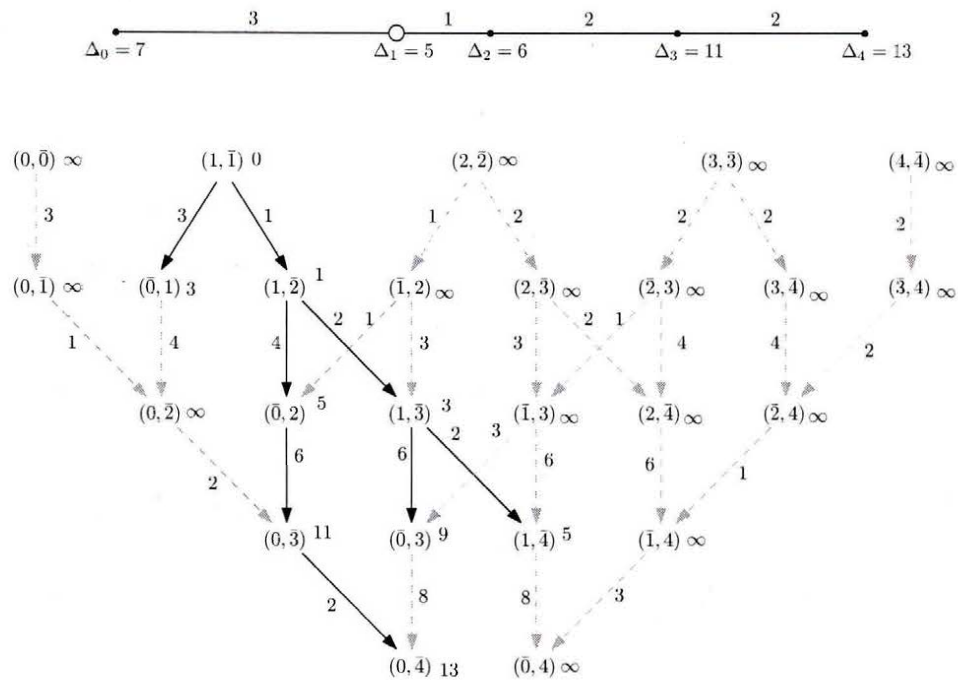

Figure 6.2: Illustration of the execution of the line-exploration algorithm starting from node 1.

Theorem 13. *Consider a line graph G and a robot placed at its starting position v_s. Algorithm 4 correctly computes an optimal exploration path which satisfies the node deadlines in $O(n^2)$ time.*

Proof. We show that for every node $v = (\bar{i}, j)$ (resp. $v = (i, \bar{j})$) of the snapshot graph the algorithm computes the shortest time $time(v)$ needed to explore the interval $[i, j]$ of the line graph.which respects the deadlines of its nodes by the robot starting at v_s, such that $i \leq s \leq j$, and ending its exploration at v_i (resp. v_j). The proof goes by induction on the layer. The claim is clearly true for any node $(i, \bar{i})$ at layer 0. Suppose

that the claim is true for all nodes at layers at most $\ell - 1$. Take any node at level ℓ, i.e., either $v = (\overline{i}, i+\ell)$ or $v = (i, \overline{i+\ell})$. Consider the shortest time exploration path ending at v. The immediate predecessors of v in this path is a node w from layer $\ell - 1$, for which the shortest-time exploration path is correctly computed by the inductive hypothesis. The time needed to reach v from w equals the time distance between $new(v)$ and $new(w)$ at the line graph. If $time(v) + weight(w \to v) \leq \Delta_{new(w)}$ then the deadline of node $new(w)$ is respected and $time(w)$ is correctly computed lines 4-6 of Algorithm 4, otherwise the exploration time of w remains at $time(w) = \infty$ as set in the InitStart procedure. The $O(n^2)$ time complexity follows directly from the properties of the snapshot graph. □

6.2.3 Arbitrary starting positions

We now consider a variation of the problem when the choice of the starting position of each robot is left to the user, or is restricted. When this choice is restricted, the user must choose the starting position of each robot within a subset of nodes of the line graph. We will show that Algorithm 4 also works in such a setting. We need, however, modify the call to procedure InitStart in line 1 of Algorithm 4, so that its first parameter equals the set of all nodes of the line at which the robot may start. An example of its execution is presented on Fig. 6.3, where a user may choose any node of the line graph as the starting position of the robot.

Observe that, for any node w of the snapshot graph, the value of $time(w)$, computed by the algorithm, represents now the shortest exploration time ending at w starting from any node of the line graph. T is now a forest with the nodes of T, whose time counter remains at ∞ isolated in T (having no children or parent in T).

Corollary 1. *Let A be the subset of nodes of the line graph which we can choose*

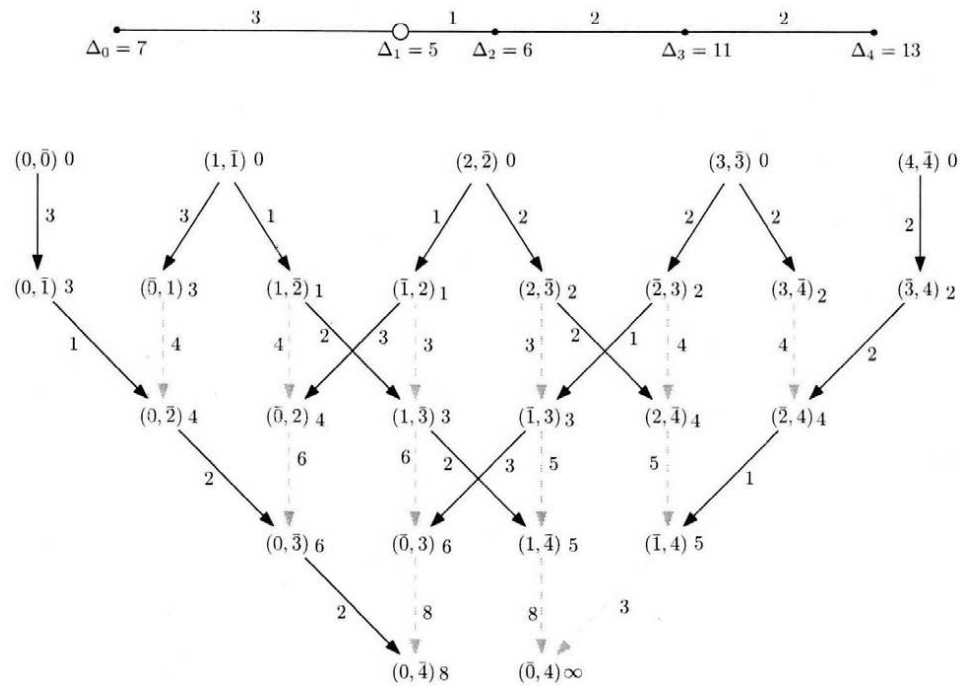

Figure 6.3: Illustration of the execution of the line-exploration algorithm in the case of arbitrary starting node. For any sub-interval $[i,j]$ of the line, the optimal robot trajectory exploring $[i,j]$ is given by the cheaper among the solid directed paths incoming to nodes $(i,\bar{j})$ and $(\bar{i},j)$.

for the starting position of the robot. Suppose that the first parameter of the call to procedure InitStart in line 1 of Algorithm 4 (A) equals the set of all nodes from zeroth level of S which correspond to the nodes of A. Such version of Algorithm 4 correctly computes in $O(n^2)$ time an optimal exploration path of the line graph, which satisfies the node deadlines. Moreover, for any sub-interval $[i,j]$ of the line, the algorithm computes an optimal robot starting position to explore $[i,j]$, the cost (time) of such exploration and the trajectory of the robot.

The proof of Corollary 1 is almost identical to the case of Theorem 13. Observe that, in the inductive step, when the parent w of any node v in T is determined, the root of the connected component of T containing v corresponds to the node of A

offering the shortest exploration path.

6.3 Multiple Robots on the Line

In this section we consider line exploration by a collection of $k < n$ mobile robots. As before we study two variants of the time optimization problem. In the first setting, the initial positions of the robots are arbitrary, i.e. the algorithm identifies the initial placement of the robots, which results in the shortest exploration time respecting the node deadlines. In the second setting, the distinct initial robot positions are given in advance. Both variants are solved using versions of dynamic programming. We start with the following observation concerning the movement of the robots.

Observation 3. *There exists an optimal exploration solution in which the robots never change their initial order along the line. Moreover, the sub-intervals of the line explored by different robots are mutually disjoint.*

Proof. This is easily proven by observing that any solution that forces two robots to cross path can be replaced by an equally efficient solution where robots turn around instead of crossing path. We also observe that there is no benefit to the exploration of a single node by multiple robots (by definition of the problem). Therefore, once two robots are located on adjacent nodes, the only efficient move for each robot is to move away from the other. □

We use the following notation. Suppose that we need to explore an interval $[i, j]$ of the line respecting the deadlines of the nodes of $[i, j]$. For the setting when the robots are placed at given initial positions, for any pair of indices i, j, such that $0 \leq i \leq j \leq n-1$, we denote by $T_{i,j}$ the optimal time of exploration of the interval $[i, j]$ using the robots placed within $[i, j]$. When the initial placement of the robots

is left to the algorithm, for any $1 \leq r \leq k$, we denote by $T_{i,j}^{(r)}$ the optimal time of exploration of the interval $[i, j]$ using r robots which may be placed at arbitrary initial positions within $[i, j]$.

6.3.1 Given initial positions

We start with the following observation

Observation 4. *Consider a line and a robot initially placed in its sub-interval $[i, j]$. Using Algorithm 4 the values $T_{i,j}$ for all $0 \leq i \leq j \leq n-1$, may be computed by the formula*

$$T_{i,j} = \min(time((i, \overline{j})), time((\overline{i}, j))) \tag{6.1}$$

Let p_i denote the initial position of robot i. We assume that we have $0 \leq p_1 < p_2 < \cdots < p_k \leq n-1$. By Observation 3 we need to partition the line into sub-intervals $[l_i, r_i]$ for $i = 1, 2, \ldots, k$ (with $l_1 = 1$ and $r_k = n$), each one explored by a different robot. The interval $[l_i, r_i]$, explored by robot i, contains its initial position p_i, but not an initial position of any other robot. Hence edges (r_i, l_{i+1}) for $i = 1, \ldots, k-1$, that we call *idle edges*, are never traversed by any robot. The following formula, is an obvious consequence of Observation 3,

$$T_{i,j} = \min_{p_q < m \leq p_{q+1}} \max(T_{i,m-1}, T_{m,j}), \tag{6.2}$$

for any $i \leq p_q, p_{q+1} < j$. Indeed, the idle edge $(m-1, m)$, separating the sub-segments of operation of robots q and $q+1$, is chosen so as to minimize the exploration time

of interval $[i, j]$.

We give first an idea of our algorithm. We generate the snapshot graph, as described in Subsection 6.2.2. Let's use the notation $p_0 = -1$ and $p_{k+1} = n$. For $m = 1, \ldots, k$ let S_m be the subgraph of S obtained by keeping the nodes $(\bar{i}, j)$ and $(i, \bar{j})$ such that $p_{m-1} < i, j < p_{m+1}$. In the first part of our algorithm, for each robot m, we run Algorithm 4 with inputs p_m and S_m, obtaining the optimal exploration time $T_{i,j}$ of each line sub-interval $[i, j]$, which contains exactly one starting position p_i, for $i = 1, 2, \ldots, k$.

In the second part of the algorithm, we combine exploration times of individual robots, in order to obtain the optimal exploration time $T_{0,j}$ using robots initially placed within $[0, j]$, subsequently for each j. Let r_j denote the number of robots initially placed in interval $[0, j]$ and suppose, that we computed the optimal exploration times of all intervals, which initially contain robots $1, 2, \ldots, r_j - 1$. When j exceeds p_{r_j} we use robot r_j and we determine the idle edges preceding the intervals of operation of r_j, resulting in the optimal exploration times of intervals, which initially contain robots $1, 2, \ldots, r_j$.

Algorithm 5: Exploration algorithm on the line with k robots at fixed initial positions

Input: Line L with starting robots' positions $p_1, p_2, \ldots p_k$

```
1 Construct the snapshot graph S from L;
2 for m = 1 to k do
3 |   Execute Algorithm 4 with inputs p_m and S_m;
4 |   for every (i, j) s.t. p_{m-1} < i ≤ j < p_{m+1} do
5 |   |   T_{i,j} := min{t_m(i, j̄), t_m(ī, j)};
6 for j = p_2 to n − 1 do
7 |   T_{0,j} := min_{p_{r_{j-1}} < m ≤ p_{r_j}} max(T_{0,m-1}, T_{m,j});
```

Theorem 14. *Algorithm 5 in $O(n^2)$ time computes the optimal exploration of the line*

by k robots initially placed at given initial positions $0 \leq p_1 < p_2 < \cdots < p_k \leq n-1$.

Proof. By Observation 3, we can assume that in an optimal solution, each robot m operates in an interval $[l_m, r_m]$, which does not contain an initial position of any other robot. Hence we have $p_{m-1} < l_m \leq p_m$ and $p_m \leq r_m < p_{m+1}$. All pairs of indices (i, j), which verify this property are considered in line 4 of Algorithm 5. By Theorem 13 and Observation 4 each such value $T_{i,j}$ is correctly computed in line 5 of Algorithm 5.

We now prove that in line 7 the Algorithm 5 correctly computes values $T_{0,j}$ for all $j = 0, 1, \ldots, n-1$. The proof goes by induction on j. For all $0 \leq j < p_2$ the interval $[0, j]$ contains a single robot, so the value $T_{i,j}$ is correctly computed in the first iteration of the for-loop in lines 2-5. Consider any $j \geq p_2$, i.e. when the interval $[0, j]$ contains more than one robot, and suppose, by the inductive hypothesis, that the values $T_{0,i}$ correspond to optimal times of exploration of segments $[0, i]$, for all $i < j$. Let T^* be the optimal time of exploration of interval $[0, j]$, which verifies the claim of Observation 3, i.e. such that there exists an idle edge $(m^* - 1, m^*)$, and $p_{r_j - 1} < m^* \leq p_{r_j}$. During such optimal exploration, robots $1, 2, \ldots, r_j - 1$ explore interval $[0, m^* - 1]$ (using some time T_1^*), and robot r_j explores interval $[m^*, j]$ (using time T_2^*). Clearly $T^* = \max(T_1^*, T_2^*)$. By the inductive hypothesis, we have $T_{0,m^*} \leq T_1^*$ and $T_{m^*+1,j} \leq T_2^*$. Consequently, we have in line 7 of Algorithm 5

$$T_{0,j} = \min_{p_{r_j-1} < m \leq p_{r_j}} \max\left(T_{0,m-1}, T_{m,j}\right) \leq \max\left(T_{0,m^*-1}, T_{m^*,j}\right) = \max(T_1^*, T_2^*) = T^*,$$

which concludes the inductive proof.

We consider now the time complexity of Algorithm 5. The snapshot graph S in line 1 is constructed in $O(n^2)$ time. Observe that since each node of S can only be in two different subgraphs S_i and S_j, we have $\sum_{i=1}^{k} |V(S_i)| \leq 2|V(S)| = O(n^2)$. Hence,

all the executions of line 3 of Algorithm 5 take $O(n^2)$ amortized time. Similarly, in line 4 of the algorithm, in all its executions, it considers $O(n^2)$ nodes of graph S. Consequently the for-loop of lines 2-5 is executed in $O(n^2)$ amortized time. As each of $O(n)$ executions of the for-loop in lines 6-7 takes $O(n)$ time we conclude the $O(n^2)$ overall time complexity of Algorithm 5. □

6.3.2 Arbitrary initial positions

This algorithm is also based on the dynamic programming approach for computing the table $T_{i,j}^{(r)}$, for all $1 \leq r \leq k$ and $0 \leq i < j \leq n-1$. The values of $T_{0,n-1}^{(k)}$ represent the optimal exploration time of the line using k robots. We use the following formula, which works for any r, r_1, r_2, where $r_1, r_2 \geq 1$, $r = r_1 + r_2$ and any $0 \leq i < j \leq n-1$.

$$T_{i,j}^{(r)} = \min_{i \leq k \leq j} \max\left(T_{i,k}^{(r_1)}, T_{k+1}^{(r_2)}\right). \tag{6.3}$$

Using Formula (6.3), the values of $T_{i,j}^{(r)}$ may be computed in a greedy manner for the increasing values of r. As Formula (6.3) may be naturally computed in $O(n)$ time, the total complexity of such a greedy approach is in $O(kn^3)$.

We give now a more efficient algorithm computing $T_{0,n-1}^{(k)}$. Observe first, that when $[i_1, j_1] \subseteq [i_2, j_2]$, then $T_{i_1,j_1}^{(r)} \leq T_{i_2,j_2}^{(r)}$. Consequently, when computing $T_{i,j}^{(r)}$, the value of index k which minimizes $\max(T_{i,k}^{(r-1)}, T_{i,k+1}^{(1)})$ may be found by a binary search (cf. function OptTime).

The following observation is easy.

Observation 5. *Consider two fixed numbers r_1, r_2 of robots. If for any interval $[i, j]$ of the line, $T_{i,j}^{(r_1)}$ and $T_{i,j}^{(r_2)}$ represent the optimal time of exploration of the interval by r_1 and r_2 robots, respectively, then function OptTime correctly computes in $O(\log n)$ time the optimal exploration time $T_{i,j}^{(r)}$ of the interval $[i, j]$ by $r = r_1 + r_2$ robots.*

Function OptTime(i, j, r_1, r_2);

```
if j − i + 1 ≤ r_1 + r_2 then
    return 0
k_low = i; k_high = j;
while k_low < k_high+1 do
    k = (k_low + k_high)/2;
    if T^(r1)_{i,k} < T^(r2)_{k+1,j} then
        k_low = k
    else
        k_high = k
return min(max(T^(r1)_{i,k_low}, T^(r2)_{k_low,j}), max(T^(r1)_{i,k_high}, T^(r2)_{k_high,j}));
```

The greedy approach would compute the values of table $T_{i,j}^{(r)}$ for any given r. Our algorithm below computes the values of $T_{i,j}^{(r)}$ when r is a power of 2 not exceeding k. Then, using formula 6.3, they are combined in $\lceil \log k \rceil$ steps, to compute the values of $T_{i,j}^{(k)}$.

Algorithm 6: ; Multiple robot line exploration with arbitrary starting positions

```
Let r_b, r_{b−1}, ..., r_0 be the consecutive bits of the binary representation of k;
Compute table T^(1)_{i,j} ;
for m = 0 to b do
    r = 2^m;
    for all pairs (i, j) such that 0 ≤ i < j ≤ n − 1 do
        T^(2r)_{i,j} = OptTime(i, j, r, r);
r = 2^b;
for m = 1 to b do
    if r_{b−m} = 1 then
        p = 2^{b−m};
        for all pairs (i, j) such that 0 ≤ i < j ≤ n − 1 do
            T^(p+r)_{i,j} = OptTime(i, j, p, r);
            r = p + r;
```

The following theorem proves the correctness and the complexity of Algorithm 6.

Theorem 15. *Algorithm 6 computes in $O(n^2 \log n \log k)$ time the optimal time needed by k robots to explore the line.*

Proof. By Corollary 1 and Formula (6.2) the usage of Algorithm 4 in line 2 of Algorithm 6 correctly computes a single robot optimal exploration time for any sub-interval of a given line. By induction on r, using Observation 5, lines 3-6 of Algorithm 6 correctly compute the optimal exploration time of any interval $[i, j]$ using 2^m robots, for any m, such that $2^m < r$.

From line 1 we have $k = r_b 2^b + r_{b-1} 2^{b-1} + \cdots + r_0 2^0$, where b is the position of the first 1-digit in the binary representation of k. We prove that, at the start of each iteration of the *for* loop from line 8, we have

1. $r = k - k \mod 2^{b+1-m}$, and
2. the table $T_{i,j}^{(r)}$ has been already computed for all $0 \leq i < j \leq n-1$.

The proof goes by induction on m. At the start of the first iteration of the loop when $m = 1$, we have $r = 2^b$. Then indeed the inductive condition is verified as

$$k - k \mod 2^{b+1-1} = (r_b 2^b + r_{b-1} 2^{b-1} + \cdots + r_0 2^0) - (r_{b-1} 2^{b-1} + r_{b-2} 2^{b-2} + \cdots + r_0 2^0) = 2^b = r$$

and the value of $T_{i,j}^{(2^b)}$ was computed previously in line 6 of the algorithm.

Suppose that the inductive condition was verified at the beginning of the m-th iteration. Suppose first that $r_{b-m} = 0$. Then the i-th iteration of the loop is empty but as $k \mod 2^{b+1-m} = k \mod 2^{b+1-(m+1)}$, so that at the beginning of the next iteration the value of r remains unchanged, it follows that the inductive condition is verified.

Consider now the case when $r_{b-m} = 1$. Then, between the start of the m-th and the $(m+1)$-st iteration of the loop in lines 10 and 13 we have $r := r + 2^{b-m}$.

Consequently, by the inductive assumption, we have at the beginning of the $(m+1)$-st iteration

$$\begin{aligned} r &= k - k \mod 2^{b+1-m} + 2^{b-m} \\ &= (r_b 2^b + r_{b-1} 2^{b-1} + \cdots + r_0 2^0) - (r_{b-m} 2^{b-m} + \cdots + r_0 2^0) + 2^{b-m} \\ &= (r_b 2^b + r_{b-1} 2^{b-1} + \cdots + r_{b-m} 2^{b-m}) = k - k \mod 2^{b+1-(m+1)} \end{aligned}$$

The value of the table $T_{i,j}^{(r)}$ is then computed in line 12 of the algorithm, which completes the induction proof.

From the inductive proof it follows that at the end of the b-th iteration of the *for* loop from line 8 (i.e. at the beginning of the non-existing $(b+1)$-st iteration) we have $r = k - k \mod 2^{b+1-(b+1)} = k$, and the table $T_{i,j}^{(r)} = T_{i,j}^{(k)}$ has been computed, which completes the proof of the correctness of the algorithm.

In line 2, the table $T_{i,j}^{(1)}$ may be computed by Algorithm 4 in $O(n^2)$ time (cf. Fig. 6.3). As $r < 2^b$, both *for* loops starting at line 3 and 8 have $O(\log k)$ iterations. Since each internal *for* loop from line 5 and 11, respectively, has $O(n^2)$ iterations calling function OptTime of complexity $O(\log n)$ we conclude that Algorithm 6 finishes in $O(n^2 \log n \log k)$ time. This algorithm produces an optimal schedule. It is not claimed that Algorithm 6 produces an optimal schedule in optimal time. □

6.4 Line Exploration with Unreliable Collections of Robots

In this section we study the exploration problem when some of the robots may be faulty, i.e., when they fail to realize their exploration tasks. In this case, other robots need to help, so that eventually every node of the line is visited by some reliable robot before its deadline. Let there be given a weighted line L, containing n nodes with given deadlines and a collection of k robots at most f of which may turn out to be faulty. Consider a schedule for k robots on the line L. We say that the schedule is f-reliable in time Δ, if for any choice of f faulty robots by an adversary, each node of the line is visited by at least one non-faulty robot before its deadline and before time Δ.

Note that, in the case when all robots are reliable, it is never useful to have more than one robot initially placed at the same position. In the case of the presence of unreliable robots, is may be required. Consequently, we will assume that it is admissible for more than one robot to start from the same node of the line.

Observation 6. *If there can be f faulty robots, then to successfully explore a node v with deadline $\Delta(v)$, node v must be visited by at least $f+1$ robots before time $\Delta(v)$.*

It is interesting to look at the *decision problem* as well as the *optimization problem* related to faulty agents. In the decision problem we look for an algorithm, which, given f and Δ, verifies whether there exists an f-reliable schedule in time Δ. In the optimization problem, we need an algorithm, which, for any given f, finds the minimal time interval Δ, which admits some f-reliable schedule in time Δ. Clearly, solving the optimization problem implies a solution to the decision problem and hardness of the decision problem implies hardness of the optimization problem. We are interested

in both settings – for fixed and for arbitrary initial positions of the robots. As the case of the arbitrary starting positions is easier we discuss this variant first.

6.4.1 Arbitrary starting positions

We prove the following theorem.

Theorem 16. *Let there be given a weighted line L, containing n nodes with given deadlines and a collection of k robots, which may be put at arbitrary starting positions on L. For any $0 < f < k$ the optimization problem involving up to f faulty robots may be solved in $O\left(n^2 \log n \log \left\lfloor \frac{k}{f+1} \right\rfloor\right)$ time.*

The idea of the proof of Theorem 16 is based on the Observation 6. To obtain an f-reliable schedule it is sufficient to partition the set of k robots into $f+1$ groups of $\left\lfloor \frac{k}{f+1} \right\rfloor$ robots, where each such group explores the segment using Algorithm 6. We show that such condition is also necessary. The time complexity follows directly from Theorem 15.

Proof. Let Δ be the time interval satisfying the claim of the theorem, in the sense that there exists an f-reliable schedule in time Δ, while for any $\Delta' < \Delta$, there does not exist an f-reliable schedule in time Δ'. We show first that the necessary and sufficient condition for the existence of such an f-reliable schedule is the following.
Condition 1: *There must exist a schedule involving $\left\lfloor \frac{k}{f+1} \right\rfloor$ robots (all reliable), starting at arbitrary initial positions on L, which solves the exploration of L in time Δ.*

Indeed, by Observation 6 each node of the line L must be explored by at least $f+1$ robots. Therefore we can partition the collection of robots into $f+1$ groups, each group entirely exploring line L. The least numerous of these groups can contain no more than $\left\lfloor \frac{k}{f+1} \right\rfloor$ robots and this group must explore L. Conversely, if $\left\lfloor \frac{k}{f+1} \right\rfloor$ robots

can explore line L in time Δ, we can form $f+1$ groups of $\lfloor \frac{k}{f+1} \rfloor$ robots each, executing the same schedule and the line is explored by each of $f+1$ independent groups.

By Theorem 15, Algorithm 6 computes the optimal time of line exploration by a collection of robots which may be placed at arbitrary initial positions. Consequently, the output of Algorithm 6 run for $r = \lfloor \frac{k}{f+1} \rfloor$ robots exactly verifies Condition 1. By Theorem 15, its time complexity is then as stated in the claim of the theorem. □

6.4.2 Given starting positions

Contrary to the case studied in the previous section, when the robots are assigned to fixed positions on the line, the existence of faulty robots leads to a problem which turns out to be NP-hard. In fact, the decision problem is hard, even in the case when all individual deadlines may be ignored (they are all larger than Δ), i.e. when the line does not have any node time constraints.

More formally we are confronted with the following problem:

Exploration of the Line with Crash Faults (ELCF) problem

Instance: A line L, a multiset P of k starting positions of robots, a number of faults f and a time interval Δ.

Question: Is there an exploration strategy for the collection of k robots, which may include up to f faulty ones, such that each node of L is visited by at least one non-faulty robot before time Δ?

Theorem 17. *The ELCF decision problem is NP-complete, and remains so even if all numerical variables are polynomially bounded. In other words, it is strongly NP-complete.*

The proof uses reduction from the Numerical 3-Dimensional Matching (N3DM), which is known to be strongly NP-complete. In the N3DM problem we have three sets, each containing the same number of q integers. We need to form q triples, each one using elements of different sets, so that the sums of all triples are the same. We construct an instance of the ELCF problem by setting $f = q - 1$ and $k = 3q$ and we put robots on the line in three groups of q initial positions. We chose the initial positions of the robots in such a way, that in order to cover the line $f+1$ times, which is needed by Observation 6, they must form triples, so that each triple of robots could explore the entire line. The choice of the initial positions is made carefully, so that the exploration is possible only if each such triple of robots corresponds to a triple of integers that is obtained from the solution of the N3DM problem.

The proof of Theorem 17 is split into two lemmas. We first show that the *ELCF* decision problem is strongly NP-hard, and then that the *ELCF* decision problem is in NP.

Lemma 21. *The ELCF decision problem is strongly NP-hard.*

Proof. We construct a polynomial-time many to one reduction from the following strongly NP-hard problem referenced as [SP16] in [147].

Numerical 3-Dimensional Matching (N3DM) problem

Instance: Three multisets of positive integers $A = \{a_1, a_2, \ldots, a_q\}, B = \{b_1, b_2, \ldots, b_q\}, C = \{c_1, c_2, \ldots, c_q\}$, and an integer S.

Question: Does there exist two permutations π_B, π_C of $[1, q]$ such that for every $1 \leq i \leq q$, $a_i + b_{\pi_B(i)} + c_{\pi_C(i)} = S$?

We construct an instance (L, P, f, Δ) of the *ELCF* problem from an instance of *N3DM* as follows. Let $a = \max_{i\in[1,q]}(a_i)$, $b = \max_{i\in[1,q]}(b_i)$ and $c = \max_{i\in[1,q]}(c_i)$. Let $I = 4S + 6a + 6b + 12c$ and $\ell = 3I - 4S - 1$. L is the line of length ℓ (with $\ell + 1$

nodes). Each edge of L has weight one. For the sake of simplicity we name i the node of L at distance i from the leftmost node. We have $3q$ robots each corresponding to an integer from one of the multisets A, B or C. For each $i = 1, 2, \ldots, q$, we put three robots: one robot $\mathcal{A}_i$ at node $\alpha_i = a_i$, one robot $\mathcal{B}_i$ at node $\beta_i = I + 2b_i$ and one robot $\mathcal{C}_i$ at node $\gamma_i = 2I + 4c_i$. The number of faults f is equal to $q - 1$ and the time interval Δ is equal to $I - 1$. The construction can be done in polynomial time. We show that the answer to the constructed instance of the $ELCF$ problem is the same as the answer to the original instance of $N3DM$.

First, assume that there exists a solution π_B, π_C to the instance of the $N3DM$ problem. We show that the robots can solve the corresponding instance of the $ELCF$ problem as follows.

Robot $\mathcal{A}_i$ will first move to the left until reaching node 0 (moving distance a_i), and then to the right until reaching node $\alpha_i^r = I - 1 - a_i$ (moving distance $I - 1 - a_i$). This can be done in time $\Delta = I - 1$ and thus robot $\mathcal{A}_i$ has visited in time all nodes in the interval $[0, \alpha_i^r]$.

Robot $\mathcal{B}_{\pi_B(i)}$ will first move to the left until reaching node $\beta_{\pi_B(i)}^l = \alpha_i^r + 1$ (moving distance $a_i + 2b_{\pi_B(i)}$) and then to the right until reaching node $\beta_{\pi_B(i)}^r = 2I - 1 - 2a_i - 2b_{\pi_B(i)}$ (moving distance $I - 1 - a_i - 2b_{\pi_B(i)}$). This can be done in time $\Delta = I - 1$ and thus it has visited in time all nodes in the interval $[\beta_{\pi_B(i)}^l, \beta_{\pi_B(i)}^r]$.

Robot $\mathcal{C}_{\pi_C(i)}$ will first move to the left until reaching node $\gamma_{\pi_C(i)}^l = \beta_{\pi_B(i)}^r + 1$ (moving distance $2a_i + 2b_{\pi_B(i)} + 4c_{\pi_C(i)}$) and then to the right until reaching node $\gamma_{\pi_C(i)}^r = 3I - 1 - 4a_i - 4b_{\pi_B(i)} - 4c_{\pi_C(i)}$ (moving distance $I - 1 - 2a_i - 2b_{\pi_B(i)} - 4c_{\pi_C(i)}$). This can be done in time $\Delta = I - 1$ and thus it has visited in time all nodes in the

interval $[\gamma^l_{\pi_C(i)}, \gamma^r_{\pi_C(i)}]$. Observe that we have :

$$3I - 1 - 4a_i - 4b_{\pi_B(i)} - 4c_{\pi_C(i)} = 3I - 4S - 1 = \ell$$

Hence $\mathcal{C}\pi_C(i)$ has visited in time all nodes in the interval $[\delta^l_{\pi_C(i)}, \ell]$.

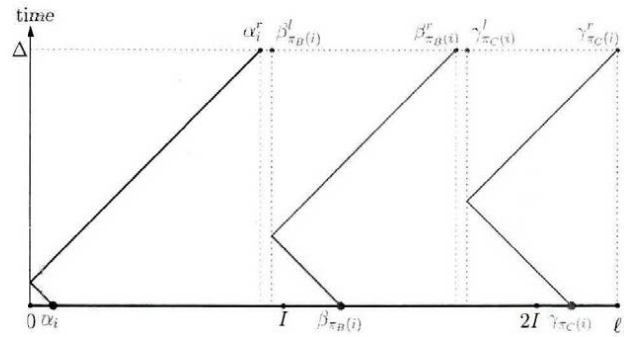

Figure 6.4: Illustration of robots $\mathcal{A}_i$, $\mathcal{B}_{\pi_B(i)}$ and $\mathcal{C}_{\pi_C(i)}$ visiting all nodes of the line.

Since $[0, \alpha^r_i] \cup [\beta^l_{\pi_B(i)}, \beta^r_{\pi_B(i)}] \cup [\gamma^l_{\pi_C(i)}, \ell] = [0, \ell]$, all nodes of the line have been visited in time by either $\mathcal{A}_i$, $\mathcal{B}_{\pi_B(i)}$ or $\mathcal{C}_{\pi_C(i)}$ for $i = 1, 2, \ldots, f+1$. It follows that every node is visited by at least one non-faulty robot and this is a solution to the $ELCF$ problem.

Now assume there is a solution to the $ELCF$ problem. Let $\mathbb{A} = \{\mathcal{A}_i \mid 1 \leq i \leq q\}$, $\mathbb{B} = \{\mathcal{B}_i \mid 1 \leq i \leq q\}$ and $\mathbb{C} = \{\mathcal{C}_i \mid 1 \leq i \leq q\}$. First, we show the following claim.

Claim 3. *The robots in $\mathbb{A}$ must visit node 0 and they are the only robots that can do it, the robots in $\mathbb{B}$ must visit node I and they are the only robots that can do it and the robots in $\mathbb{C}$ must visit node ℓ and they are the only robots that can do it.*

For $i = 1, 2, \ldots, q$, the robots in positions β_i and γ_i are too far (at distance at least I) to reach node 0 in time smaller than $\Delta = I - 1$. The robots in positions α_i are the only ones that can visit the node 0 and since this node must be visited by $f + 1 = q$ robots, they must all visit it. Hence, the robots in positions α_i cannot visit

in time node I which is at distance I of node 0. Similarly, the robots in positions β_i are the only ones that can visit the node I and since this node must be visited by $f+1=q$ robots, they must all visit it. Similarly, the robots in positions γ_i are the only ones that can visit the node $2I$ and so node ℓ (since $\ell > 2I$) and since this node must be visited by $f+1=q$ robots, they must all visit it. This ends the proof of the claim.

For $i=1,2,\ldots,q$, let $[0,\alpha_i^r]$ be the interval of nodes visited by robot $\mathcal{A}_i$, $[\beta_i^l,\beta_i^r]$ be the interval of nodes visited by robot $\mathcal{B}_i$ and $[\gamma_i^l,\gamma_i^r]$ be the interval of nodes visited by robot $\mathcal{C}_i$.

Claim 4. *There are two permutations $\pi_B(i)$ and $\pi_C(i)$ such that for $i=1,2\ldots,q$, $\beta^l_{\pi_B(i)}=\alpha_i^r+1$ and $\gamma^l_{\pi_C(i)}=\beta^r_{\pi_B(i)}+1$.*

First observe that if there is a portion of the line that is visited by more than $f+1=q$ robots, then it means that there are robots from two different sets (for example, robots from sets $\mathbb{A}$ and $\mathbb{B}$). We can then cut the trajectory of some of the robots in order to decrease the number of robots visiting the same node. So we can assume without loss of generality that each node is visited by exactly q robots. This means that there is a partition of the robots into q subsets, such that every node of the line is visited in time by exactly one robot of each subset. By the first claim, there is one robot of each set $\mathbb{A}$, $\mathbb{B}$ and $\mathbb{C}$ in each of q subsets. Hence, for $i=1,2,\ldots,q$, there is a subset of robots $\{\mathcal{A}_i,\mathcal{B}_{\pi_B(i)},\mathcal{C}_{\pi_C(i)}\}$ that must visit all the nodes of the line. Since two robots of the same subset do not visit the same node, their intervals are disjoint. This ends the proof of the claim.

By the second claim, Robot $\mathcal{A}_i$ can travel a distance Δ to search its interval $[0,\alpha_i^r]$. Observe that $\mathcal{A}_i$ starts at distance a_i from node 0. Since $\Delta > 3a \geq 3a_i$, the optimal way for robot $\mathcal{A}_i$ to search its interval is to first go to the left and then to the right.

So, we have $\alpha_i^r = I - 1 - a_i$. By Claim 4, we have $\beta^l_{\pi_B(i)} = \alpha_i^r + 1 = I - a_i$. Robot $\mathcal{B}_{\pi_B(i)}$ starts at distance $a_i + 2b_i$ from node $\beta^l_{\pi_B(i)}$. Since $\Delta > 3a + 6b \geq 3(a_i + 2b_i)$, the optimal way for robot $\mathcal{B}_i$ to search its interval is to first go to the left and then to the right. So, we have $\beta^r_{\pi_B(i)} = I - 2a_i - 2b_{\pi_B(i)} - 1$. By the last Claim, we have $\gamma^l_{\pi_C(i)} = \beta_i^r + 1 = I - 2a_i - 2b_{\pi_B(i)}$. Robot $\mathcal{C}_{\pi_C(i)}$ starts at distance $2a_i + 2b_{\pi_B(i)} + 4c_{\pi_C(i)}$ from node $\gamma^l_{\pi_C(i)}$. Since $\Delta > 6a + 6b + 12c \geq 3(2a_i + 2b_{\pi_B(i)} + 4c_{\pi_C(i)})$, the optimal way for robot $\mathcal{C}_i$ to search its interval is to first go to the left and then to the right. Observe that since robot $\mathcal{C}_{\pi_C(i)}$ must visit node ℓ, we have :

$$3I - 4a_i - 4b_{\pi_B(i)} - 4c_{\pi_C(i)} - 1 = 3I - 4S - 1 \iff a_i + b_{\pi_B(i)} + c_{\pi_C(i)} = S$$

Hence, π_B, π_C, π_D is a solution for the instance of the $N3DM$ problem. □

Lemma 22. *The ELCF decision problem is in NP.*

Proof. We consider the verifier-based definition of NP. A certificate for the instance of the $ELCF$ decision problem is simply the set of the trajectories of the k robots. Each trajectory is of length $O(n^2)$ and hence this certificate is in $O(kn^2)$ and so polynomial in the size of the instance. We can check in polynomial time (by simulating the trajectories of the robots) that every node of the line is visited before time Δ by at least $f + 1$ robots. Thus, the certificate can be verified in polynomial time. □

An interested reader may observe that, given a configuration of k robots on a line and a time Δ, finding what is the maximal number f of robot faults that will still guarantee the exploration, is also strongly NP-hard.

Interestingly, we show in the following section, that the corresponding problem for the ring environment (even its optimization version) has a polynomial time solution. More exactly, we can polynomially compute the smallest time needed to explore the

ring by k robots placed at given initial positions, when any sub-collection of up to f robots may turn out to be faulty.

6.5 The Ring Environment

In this section we show that most of the results for the line environment may be adapted to work on the ring. However, the $ELCF$ decision problem turns out to have a polynomial-time solution for the ring.

Suppose that the ring R contains nodes $0, 1, 2, \ldots, n-1$ in that counterclockwise order around R. Then every node i of the ring has a counterclockwise neighbour $(i+1) \mod n$ and a clockwise neighbour $(i-1) \mod n$. Consequently, in this section, all the ring node indices are implicitly taken modulo n. The approach used for the ring also starts by creating the snapshot graph, however slightly different from the one introduced in Section 6.2.1. The nodes of the snapshot graph are of the form $(i, \bar{j})$ and $(\bar{i}, j)$, where the node of the ring marked with the bar denotes the current position of the robot and $[i, j]$ is the segment of the ring already explored by the robot taken in the counterclockwise direction from i to j. Observe that, the terminal nodes of the snapshot graph, i.e. those which correspond to the exploration of every node of the ring, are now all nodes $(i, \bar{j})$ and $(\bar{i}, j)$, such that $(j - i) \mod n = 1$, i.e. i is the counterclockwise neighbour of j. Such snapshot graph also has $O(n^2)$ nodes of constant degree (see Fig. 6.5 below). Consequently, by using the argument from Theorem 14 we have the following Observation.

Observation 7. *All values of $T_{i,j}$ for pairs (i, j), such that each pair denotes a counterclockwise segment around the ring containing an initial position of at most one robot, may be computed in $O(n^2)$ time.*

Observe that, there exists an optimal solution for the ring with idle edges between

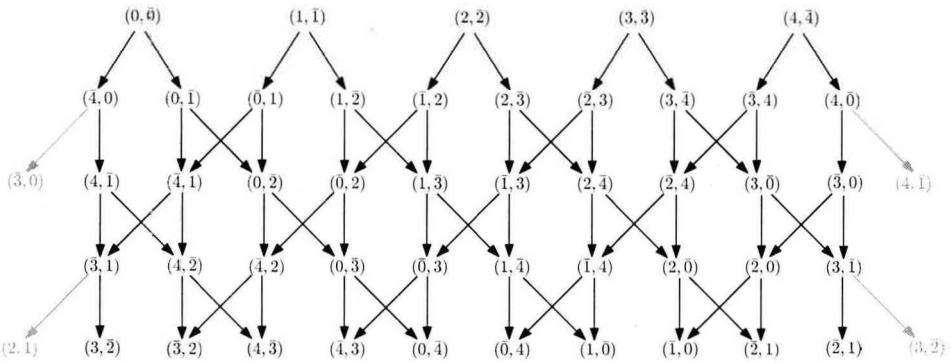

Figure 6.5: Snapshot graph for a case of ring R of five nodes. Grey nodes and edges are duplicates of other nodes at the same level (for presentation clarity). All last level nodes correspond to the ring entirely explored.

initial positions of consecutive robots. By removing one such edge the ring becomes a line-segment. Consequently, most of our observations for lines may be applied for rings. In particular, for the case of robots which may be placed at arbitrary initial positions on the ring, the following Corollary is obvious.

Corollary 2. *In $O(n^2 \log n \log k)$ time it is possible to compute the optimal time of exploration of the ring of size n by a set of k robots, which may be placed at arbitrary initial positions.*

Indeed, it is sufficient to apply Algorithm 6, in which in lines 5 and 12 we consider all pairs (i, j) (rather than pairs for which $i < j$).

In the case of robots at given initial positions, the adaptation of the line algorithm to the ring case is also relatively easy, with some compromise on its time complexity. We have the following Proposition.

Proposition 1. *There exists an $O\left(n^2 + \frac{n^2}{k} \log n\right)$ algorithm for computing an optimal exploration of the ring R of size n using k mobile robots, initially placed at fixed positions on R.*

Proof. Take a pair $i, i+1$ of successive robots around the ring R for which the distance of their initial positions is the smallest. In an optimal exploration on the segment $[p_i, p_{i+1}]$ of R, one of its edges is idle. Knowing, which such edge is idle, we might remove it from R converting the ring to a line segment. Then the line exploration Algorithm 5 may be executed for such a segment. As the segment $[p_i, p_{i+1}]$ is of size $O(n/k)$, one possible approach is to try all the possibilities of making idle every edge of $[p_i, p_{i+1}]$, each time running Algorithm 5 for the ring segment thus obtained. This would result in overall complexity $O(n^3/k)$.

Consider the following, more careful adaptation of Algorithm 5 for the ring. Its first part (lines 1-5) may be run once, computing all values $T_{i,j}$ in $O(n^2)$ time. Then the second part (lines 6-7) are repeated $O(n/k)$ times, i.e. for all segments $O(n/k)$ obtained from R by removal of each possible idle edge between p_i and p_{i+1}. Moreover, the min computation from line 7, by Observation 5, may be computed in $(\log n)$ time. This results in an $O(\frac{n^2}{k} \log n)$ complexity of lines 6-7 hence in $O\left(n^2 + \frac{n^2}{k} \log n\right)$ ring exploration algorithm. □

We now consider unreliable robots. Similarly to the line exploration case, every node of the environment must be explored $f + 1$ times by different robots before its deadline.

Consider first the case of robots which may be placed at arbitrary initial positions on the ring R. Suppose that we denote by $R^{(f+1)}$ a ring obtained in the following way. We cut R at any node v, obtaining a line segment starting and ending by a copy of v. We merge $f+1$ copies of such segment, identifying the starting and the ending nodes of consecutive copies, obtaining a segment of $n(f + 1)$ nodes. Finally, we identify both endpoints of such segment obtaining a ring $R^{(f+1)}$. Observe that, covering R by k robots' exploration trajectories, so that each node of R is visited $f + 1$ times,

is equivalent to exploring $R^{(f+1)}$ using k robots, so that each of its nodes is visited (once) before its deadline. As the size of $R^{(f+1)}$ is in $O(nf)$, from Corollary 2 we get

Corollary 3. *Suppose that in an n-node ring we can place at arbitrary initial positions k robots, which may include up to f faulty ones. In $O(n^2 f^2 \log k(\log n + \log f))$ time it is possible to compute the optimal time of exploration of the ring.*

If the initial positions of the robots on the ring are given in advance, contrary to the case of the line segment, it is possible to decide in polynomial time whether there exists an f-reliable schedule in any given time Δ.

Proposition 2. *Consider a ring R of size n and k robots placed at given initial positions at the nodes of S. For any given time Δ there is a polynomially-bounded algorithm that decides whether ring R may be explored by its robots within time Δ.*

Proof. Create ring $R^{(f+1)}$ formed of $f+1$ copies of R, thus obtaining $k(f+1)$ possible starting positions for k robots. We need to find an exploration of ring $R^{(f+1)}$ in time T using k robots, which may be placed at $k(f+1)$ starting positions. If such explorations are possible, then there exists one, for which each robot covers a disjoint segment of $R^{(f+1)}$, with idle edges separating them. Consider one such edge and remove it from $R^{(f+1)}$, obtaining a segment S of size $n(f+1)-1$. The set of k robots explore S in time T. As the chosen idle edge belongs to some copy of ring R, it is sufficient to consider n segments $S_0, S_1, \ldots, S_{n-1}$ of size $n(f+1)-1$ and check whether one of them may be explored in time T.

From the corresponding snapshot graph, we compute first for any position i on the ring $R^{(f+1)}$, the value $P(i)$ denoting the largest position j, in the counterclockwise direction around $R^{(f+1)}$, such that a robot placed at a permitted initial position can explore in time T the segment $[i,j]$ of ring $R^{(f+1)}$. Consider now an algorithm

deciding for any given segment S_m, where $m = 0, 1 \ldots, n-1$, whether S_m may be explored in time T by some set of k robots, each of which may be placed at any of the given $k(f+1)$ starting positions. Starting from the initial endpoint of S_m, for all consecutive values of $r = 1, 2, \ldots, k$, we compute the largest index $i^r_{S_m}$, such that the initial sub-segment of S_m ending at node $i^r_{S_m}$ may be explored by a set of r robots in time T. We can prove by induction on r that

$$i^{r+1}_{S_m} = P(i^r_{S_m} + 1)$$

If $i^k_{S_m}$ reaches (or exceeds) the last node of segment S_m, then S_m is explorable by k robots in time T.

We repeat the procedure for all segments S_m. As ring $R^{(f+1)}$ is possible to be explored at time T if and only if one of the segments S_m may be explored in time T this concludes the proof.

□

6.6 NP-hardness for Star Graphs

We gave exploration algorithms for lines and rings with time constraints on the nodes. It is easy to see that the exploration problem is hard for graphs, even for the case of a single robot and a graph with edges of unit length. Indeed, for a graph on n nodes, by setting all its node deadlines to $n-1$, an instance of exploration problem is equivalent to finding a Hamiltonian path. However, we show below that the exploration problem is hard for graphs as simple as stars and already for two mobile robots.

Proposition 3. *The exploration problem respecting node deadlines for given starting positions of the robots is NP-hard. This problem is also NP-hard if the starting*

positions are arbitrary.

Proof. We accomplish the reduction from the Partition Problem [136].

Partition problem

Instance: A sets of q of positive integers $A = \{a_1, a_2, \ldots, a_q\}$

Question: Does there exist a partition of set A into two subsets of equal sum.

We construct a polynomial-time reduction from the Partition problem. Consider an instance of the partition problem with the set $A = \{a_1, a_2, \ldots, a_q\}$. Let $\sum_{i=1}^{q} a_i = 2\sigma$. We design the corresponding instance of the star exploration problem. Consider a star consisting of $q+4$ edges $e_1, e_2, \ldots, e_{q+4}$. Let the weight w of each edge be such that $w(e_i) = a_i$, for $i = 1, 2, \ldots, q$, and $w(e_{q+1}) = w(e_{q+2}) = w(e_{q+3}) = w(e_{q+4}) = 4\sigma$. Take two mobile robots 1 and 2 and put them at the starting positions at the endpoints of edges e_{q+1} and e_{q+2}, different from the centre of the star. Let the deadline of each node of the star be $\Delta(e_i) = 10\sigma$, for $i = 1, 2, \ldots, q+4$. Note that the sum of the weights of all edges of the star equals 18σ. Further, observe that each robot has to end its route at one of the edges e_{q+3} and e_{q+4}. Indeed, otherwise one of the edges e_{q+3} or e_{q+4} would be traversed twice (by the same robot in both directions) and the sum of the trajectories of both robots would exceed 22σ. Hence one of the robots would arrive to its last node after time 11σ and its deadline would not be met.

Consequently, robots must traverse once both edges e_{q+1} and e_{q+2} at the beginning of their respective routes and finish the routes by traversing edges e_{q+3} and e_{q+4}. Each of the remaining edges e_i, for $i = 1, 2, \ldots, q$, must be traversed in both directions and the sum of the robot route lengths is at least $4 \cdot 4\sigma + \sum_{i=1}^{q} w(e_i) = 20\sigma$. In order for both robots to reach their last nodes within their deadline time of 10σ, each of them must traverse the subset of edges of total length σ. This requires solving the given instance of the partition problem.

It is easy to see that the above reduction works not only for the star exploration from given starting positions, but also from arbitrary ones. □

6.7 Additional Remarks and Conclusion

We studied the question of exploring graphs with time constraints by collections of unreliable robots. When all robots are reliable we used dynamic programming to give efficient exploration algorithms for line graphs and rings. We showed, however, that the problem is NP-hard for graphs as simple as stars. We showed how to extend, in most cases, our solutions to unreliable collections of robots.

One of our results is quite unexpected and important. Suppose that a collection of robots, placed on a line, may contain an unknown subset of robots (of bounded size), which turn out to be crash faulty. Verifying whether it is possible to explore the line within a given time bound is an NP-hard problem. The same problem on the ring has a polynomial-time solution.

An interested reader may observe that our positive results imply the possibility to compute the *resilience* of the configuration, i.e. given a time Δ, to find the largest value f, such that there exists a schedule assuring exploration when any set of f robots turns out to be unreliable.

In this section, we did not actually produce schedules for our robots, but we only computed the optimal times when such schedules may be completed. However, from our work it is implicitly clear how to generate such schedules. We proved the optimality of the schedules but we did not prove the optimality of our algorithms. One of the possible open problems is to attempt to design algorithms of better time complexity.

Chapter 7

Conclusion and Future Work

The purpose of our thesis was to study various exploration problems using mobile agents. These problems concern both geometric and graph environments. We were especially interested in situations where robots may experience faults, be they crash faults or Byzantine faults. We also considered cases where robots had different speeds.

Our contribution covered four specific topics. Chapter 3 focused on exploration of the two-dimensional Euclidean plane by a group of k robots. Three main variations were studied: in the first variation, robots were all reliable. In the second variation, f robots could experience crash-faults. In the third variation, f robots could experience Byzantine faults. For all variations, we introduced algorithms for two communication models: wireless and face-to-face. We also discussed upper and lower bounds: all those algorithms are asymptotically optimal, with the exception of the byzantine face-to-face model, where our algorithm is only optimal if $2f+1 < k$. A natural extension of our work is to consider collections of robots with possibly distinct visibility ranges. An interesting open question concerns exploration of polygonal environments using robots with bounded or unlimited visibility ranges.

In chapter 4, we studied the best way for one or multiple robots to intercept a

bus travelling alongside the circumference of a circle of unit length, and we provided algorithms for various circumstances: when the speed of the bus was either known or unknown, and when its direction was either known or unknown. Our work also opens several possibilities for further work, such as:

- bus with non-constant speed (i.e. known only upper and lower bound);
- bus with known movement function (e.g. $f(t)$, where $f(t)$ gives the speed of the bus at time t), but unknown initial location and direction (determined by the sign at $f(t)$);
- bus time shift (e.g. knowing that the bus moves according to $f(t+t_0)$, for some t_0, but not knowing t_0);
- agents with different and possibly non-constant speeds;
- other domains, like trees or arbitrary graphs; and
- different communication model (e.g. face-to-face, or limited visibility).

Chapter 5 studied the problem of evacuation on a disc. We studied two variations: crash-faults and Byzantine faults under the wireless communication model. We evaluated the efficiency of the algorithm by measuring the time elapsed before all reliable robots gather at the location of the exit. Our work focused on providing upper and lower bounds for the case of 3 robots, 1 of which is faulty. There are several challenging open problems, such as:

- closing the gaps between the upper and lower bounds for either robot fault (either crash or Byzantine) model with wireless communication, as presented in our work;

- exploring other types of communication models (e.g. face-to-face, or even limited visibility);
- identifying the upper and lower bounds for more than three robots, f of which may be faulty, and derive asymptotic bounds similar to the results of [56]; and
- exploring robots with different maximal speeds.

Despite the fact that obtaining tight bounds for evacuation problems are known often to lead to functions which can be a challenge to optimize, the algorithmic insights derived by this interaction between robot mobility and communication can lead to rewarding applications of distributed computing in search and evacuation.

In chapter 6, we studied exploration problems in graphs with deadlines on nodes. We studied various graphs (line, ring, star) with one, then multiple robots. We discussed the complexity of algorithms that solved this problem, and evaluated the impact of crash faults on the complexity of the problems. This leaves some open problems, such as:

- designing algorithms of better time complexity;
- studying the feasibility of the exploration of the star graph by a single robot in polynomial time;
- the case of a single robot in a tree with node deadlines; and
- identifying the smallest (or simplest) class of graphs for which the exploration by a single robot is hard.

Bibliography

[1] Y. Afek, R. Kecher, and M. Sulamy. Optimal and resilient pheromone utilization in ant foraging. *arXiv preprint arXiv:1507.00772*, 2015.

[2] N. Agmon and D. Peleg. Fault-tolerant gathering algorithms for autonomous mobile robots. *SIAM Journal on Computing*, 36(1):56–82, 2006.

[3] R. Ahlswede and I. Wegener. *Search problems*. Wiley-Interscience, 1987.

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

[5] S. Albers, K. Kursawe, and S. Schuierer. Exploring unknown environments with obstacles. *Algorithmica*, 32(1):123–143, 2002.

[6] S. Alpern, R. Fokkink, S. Gal, and M. Timmer. On search games that include ambush. *SIAM Journal on Control and Optimization*, 51(6):4544–4556, 2013.

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

[8] E. M. Arkin, M. A. Bender, S. P. Fekete, J. S. B. Mitchell, and M. Skutella. The freeze-tag problem: how to wake up a swarm of robots. In *Proceedings*

of the thirteenth annual ACM-SIAM symposium on Discrete algorithms, pages 568–577. Society for Industrial and Applied Mathematics, 2002.

[9] E. M. Arkin, M. A. Bender, and D. Ge. Improved approximation algorithms for the freeze-tag problem. In *Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures*, pages 295–303. ACM, 2003.

[10] S. Azad. *Foraging Algorithms for Robotic Swarms*. PhD thesis, Concordia University Montréal, Québec, Canada, 2015.

[11] R. Baeza Yates, J. Culberson, and G. Rawlins. Searching in the plane. *Information and Computation*, 106(2):234–252, 1993.

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

[13] B. Balamohan, S. Dobrev, P. Flocchini, and N. Santoro. Exploring an unknown dangerous graph with a constant number of tokens. *Theoretical Computer Science*, 610:169–181, 2016.

[14] R. Baldoni, F. Bonnet, A. Milani, and M. Raynal. On the solvability of anonymous partial grids exploration by mobile robots. In *International Conference On Principles Of Distributed Systems*, pages 428–445. Springer, 2008.

[15] E. Bampas, J. Czyzowicz, L. Gasieniec, D. Ilcinkas, R. Klasing, T. Kociumaka, and D. Pajak. Linear search by a pair of distinct-speed robots. In *SIROCCO*, pages 195–211. LNCS, 2016.

[16] E. Bampas, J. Czyzowicz, L. Gąsieniec, D. Ilcinkas, and A. Labourel. Almost optimal asynchronous rendezvous in infinite multidimensional grids. In *International Symposium on Distributed Computing*, pages 297–311. Springer, 2010.

[17] E. Bampas, J. Czyzowicz, D. Ilcinkas, and R. Klasing. Beachcombing on strips and islands. In *International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics*, pages 155–168. Springer, 2015.

[18] 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.

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

[20] R. Bellman. An optimal search. *Siam Review*, 5(3):274–274, 1963.

[21] M. A. Bender, A. Fernández, D. Ron, A. Sahai, and S. Vadhan. The power of a pebble: Exploring and mapping directed graphs. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 269–278. ACM, 1998.

[22] M. A Bender and D. K. Slonim. The power of team exploration: Two robots can learn unlabeled directed graphs. In *Foundations of computer science, 1994 proceedings., 35th annual symposium on*, pages 75–85. IEEE, 1994.

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

[24] S. N. Bhatt, S. Even, D. S. Greenberg, and R. Tayar. Traversing directed eulerian mazes. *J. Graph Algorithms Appl.*, 6(2):157–173, 2002.

[25] S. Bock. Solving the traveling repairman problem on a line with general processing times and deadlines. *European Journal of Operational Research*, 244(3):690–703, 2015.

[26] A. Bonato, E. Chiniforooshan, and P. Prałat. Cops and robbers from a distance. *Theoretical Computer Science*, 411(43):3834–3844, 2010.

[27] A. Bonato, P. Golovach, G. Hahn, and J. Kratochvíl. The capture time of a graph. *Discrete Mathematics*, 309(18):5588–5595, 2009.

[28] A. Bonato and R. Nowakowski. *The game of cops and robbers on graphs*. AMS, 2011.

[29] A. Bonato, P. Prałat, and C. Wang. Pursuit-evasion in models of complex networks. *Internet Mathematics*, 4(4):419–436, 2007.

[30] S. Bouchard, Y. Dieudonné, and B. Ducourthial. Byzantine gathering in networks. *Distributed Computing*, 29(6):435–457, 2016.

[31] Z. Bouzid, M. G. Potop-Butucaru, and S. Tixeuil. Optimal byzantine-resilient convergence in uni-dimensional robot networks. *Theoretical Computer Science*, 411(34-36):3154–3168, 2010.

[32] S. Brandt, F. Laufenberg, Y. Lv, D. Stolz, and R. Wattenhofer. Collaboration without communication: Evacuating two robots from a disk. In *Algorithms and Complexity - 10th International Conference, CIAC 2017, Athen, Greece, May 24-26, 2017. Proceedings*, 2017.

[33] P. Brass, F. Cabrera-Mora, A. Gasparri, and J. Xiao. Multirobot tree and graph exploration. *IEEE Transactions on Robotics*, 27(4):707–717, 2011.

[34] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun. Collaborative multi-robot exploration. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 476–481. IEEE, 2000.

[35] W. Burgard, M. Moors, and F. Schneider. Collaborative exploration of unknown environments with teams of mobile robots. In *Advances in plan-based control of robotic agents*, pages 52–70. Springer, 2002.

[36] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on robotics*, 21(3):376–386, 2005.

[37] D. Caissy and A. Pelc. Exploration of faulty hamiltonian graphs. *International Journal of Foundations of Computer Science*, 27(07):809–827, 2016.

[38] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. In *Ad-hoc, mobile, and wireless networks, LNCS*, volume 6811, pages 346–359. Springer, 2011.

[39] J. Chalopin, S. Das, and A. Kosowski. Constructing a map of an anonymous graph: Applications of universal sequences. In *International Conference On Principles Of Distributed Systems*, pages 119–134. Springer, 2010.

[40] J. Chalopin, S. Das, and N. Santoro. Rendezvous of mobile agents in unknown graphs with faulty links. In *International Symposium on Distributed Computing*, pages 108–122. Springer, 2007.

[41] J. Chalopin, Y. Dieudonné, A. Labourel, and A. Pelc. Rendezvous in networks in spite of delay faults. *Distributed Computing*, 29(3):187–205, 2016.

[42] B. Chlebus, L. Gasieniec, A. Gibbons, A. Pelc, and W. Rytter. Deterministic broadcasting in ad hoc radio networks. *Distributed computing*, 15(1):27–38, 2002.

[43] N. Christofides, V. Campos, A. Corberán, and E. Mota. An algorithm for the rural postman problem on a directed graph. *Mathematical Programming Study*, 26:155–166, 1986.

[44] M. Chrobak, L. Gasieniec, Gorry T., and R. Martin. Group search on the line. In *Proceedings of SOFSEM 2015, LNCS 8939*, pages 164–176. Springer, 2015.

[45] T. H. Chung, G. A. Hollinger, and V. Isler. Search and pursuit-evasion in mobile robotics. *Autonomous robots*, 31(4):299, 2011.

[46] M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro. Distributed computing by mobile robots: Gathering. *SIAM Journal on Computing*, 41(4):829–879, 2012.

[47] R. Cohen and D. Peleg. Convergence properties of the gravitational algorithm in asynchronous robot systems. *SIAM Journal of Computing*, 41(1):1516–1528, 2005.

[48] R. Cohen and D. Peleg. Convergence of autonomous mobile robots with inaccurate sensors and movements. *SIAM Journal on Computing*, 38(1):276–302, 2008.

[49] A. Collins, J. Czyzowicz, L. Gasieniec, A. Kosowski, E. Kranakis, D. Krizanc, R. Martin, and O. Morales Ponce. Optimal patrolling of fragmented boundaries. In *Proceedings of the twenty-fifth annual ACM symposium on Parallelism in algorithms and architectures*, pages 241–250. ACM, 2013.

[50] A. Collins, J. Czyzowicz, L. Gąsieniec, A. Kosowski, and R. Martin. Synchronous rendezvous for location-aware agents. In *International Symposium on Distributed Computing*, pages 447–459. Springer, 2011.

[51] A. Collins, J. Czyzowicz, L. Gąsieniec, and A. Labourel. Tell me where i am so i can meet you sooner. In *International Colloquium on Automata, Languages, and Programming*, pages 502–514. Springer, 2010.

[52] A. Corberán and J. M. Sanchis. A polyhedral approach to the rural postman problem. *European Journal of Operational Research*, 79(1):95–114, 1994.

[53] J. Czyzowicz, S. Dobrev, K. Georgiou, E. Kranakis, and F. MacQuarrie. Evacuating two robots from multiple unknown exits in a circle. *Theoretical Computer Science*, 2016.

[54] J. Czyzowicz, S. Dobrev, R. Královič, S. Miklík, and D. Pardubská. Black hole search in directed graphs. In *International Colloquium on Structural Information and Communication Complexity*, pages 182–194. Springer, 2009.

[55] J. Czyzowicz, L. Gąsieniec, K. Georgiou, E. Kranakis, and F. MacQuarrie. The beachcombers' problem: walking and searching with mobile robots. *Theoretical Computer Science*, 608:201–218, 2015.

[56] J. Czyzowicz, L. Gasieniec, T. Gorry, E. Kranakis, R. Martin, and D. Pajak. Evacuating robots from an unknown exit located on the perimeter of a disc. In *DISC 2014*. Springer, Austin, Texas, 2014.

[57] J. Czyzowicz, L. Gasieniec, A. Kosowski, and E. Kranakis. Boundary patrolling by mobile agents with distinct maximal speeds. In *ESA*, pages 701–712, 2011.

[58] J. Czyzowicz, L. Gasieniec, A. Kosowski, E. Kranakis, D. Krizanc, and N. Taleb. When patrolmen become corrupted: Monitoring a graph using faulty mobile robots. In *Algorithms and Computation - Proceedings of 26th International Symposium, ISAAC 2015*, pages 343–354, 2015.

[59] J. Czyzowicz, L. Gąsieniec, A. Kosowski, E. Kranakis, O. Morales-Ponce, and E. Pacheco. Position discovery for a system of bouncing robots. *Information and Computation*, 244:122–133, 2015.

[60] J. Czyzowicz, K. Georgiou, E. Kranakis, D. Krizanc, L. Narayanan, J. Opatrny, and S. Shende. Search on a line by byzantine robots. In *27th International Symposium on Algorithms and Computation, ISAAC 2016, December 12-14, 2016, Sydney, Australia*, pages 27:1–27:12, 2016.

[61] J. Czyzowicz, K. Georgiou, E. Kranakis, L. Narayanan, J. Opatrny, and B. Vogtenhuber. Evacuating robots from a disc using face to face communication. In *Proceedings of CIAC 2015, LNCS*, volume 9079, pages 140–152. Springer, Paris, France, 2015.

[62] J. Czyzowicz, K. Georgiou, E. Kranakis, L. Narayanan, J. Opatrny, and B. Vogtenhuber. Evacuating using face-to-face communication. *Proceedings CIAC 2015 (also CoRR)*, abs/1501.04985, 2015.

[63] J. Czyzowicz, D. Ilcinkas, A. Labourel, and A. Pelc. Asynchronous deterministic rendezvous in bounded terrains. In *International Colloquium on Structural Information and Communication Complexity*, pages 72–85. Springer, 2010.

[64] J. Czyzowicz, D. Ilcinkas, A. Labourel, and A. Pelc. Worst-case optimal exploration of terrains with obstacles. *Information and Computation*, 225:16–28, 2013.

[65] J. Czyzowicz, A. Kosowski, and A. Pelc. How to meet when you forget: log-space rendezvous in arbitrary graphs. *Distributed Computing*, 25(2):165–178, 2012.

[66] J. Czyzowicz, D. Kowalski, E. Markou, and A. Pelc. Searching for a black hole in synchronous tree networks. *Combinatorics, Probability and Computing*, 16(4):595–619, 2007.

[67] J. Czyzowicz, E. Kranakis, D. Krizanc, Narayanan. L., and J. Opatrny. Search on a line with faulty robots. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, Chicago, IL, USA, July 25-28, 2016*, pages 405–414, 2016.

[68] J. Czyzowicz, E. Kranakis, D. Krizanc, L. Narayanan, J. Opatrny, and S. Shende. Wireless autonomous robot evacuation from equilateral triangles and squares. In *Ad-hoc, Mobile, and Wireless Networks - 14th International Conference, ADHOC-NOW 2015, Athens, Greece, June 29 - July 1, 2015, Proceedings*, pages 181–194, 2015.

[69] J. Czyzowicz, E. Kranakis, and E. Pacheco. Localization for a system of colliding robots. *Distributed Computing*, 28(4):245–252, 2015.

[70] J. Czyzowicz, E. Kranakis, D. Pajak, and N. Taleb. Patrolling by robots equipped with visibility. In *International Colloquium on Structural Information and Communication Complexity*, pages 224–234. Springer, 2014.

[71] J. Czyzowicz, A. Pelc, and A. Labourel. How to meet asynchronously (almost) everywhere. *ACM Transactions on Algorithms (TALG)*, 8(4):37, 2012.

[72] G. D'Angelo, G. Di Stefano, and A. Navarra. How to gather asynchronous oblivious robots on anonymous rings. In *International Symposium on Distributed Computing*, pages 326–340. Springer, 2012.

[73] G. D'Angelo, A. Navarra, and N. Nisse. *Robot Searching and Gathering on Rings under Minimal Assumptions*. PhD thesis, INRIA, 2013.

[74] G. D'Angelo, A. Navarra, and N. Nisse. Gathering and exclusive searching on rings under minimal assumptions. In *International Conference on Distributed Computing and Networking*, pages 149–164. Springer, 2014.

[75] S. Das, P. Flocchini, S. Kutten, A. Nayak, and N. Santoro. Map construction of unknown graphs by multiple agents. *Theoretical Computer Science*, 385(1-3):34–48, 2007.

[76] S. Das, P. Flocchini, A. Nayak, and N. Santoro. Distributed exploration of an unknown graph. In *International Colloquium on Structural Information and Communication Complexity*, pages 99–114. Springer, 2005.

[77] S. Das, P. Flocchini, G. Prencipe, N. Santoro, and M. Yamashita. The power of lights: Synchronizing asynchronous robots using visible bits. In *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*, pages 506–515. IEEE, 2012.

[78] 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.

[79] X. Défago, M. Gradinariu, S. Messika, and P.R. Parvédy. Fault-tolerant and self-stabilizing mobile robots gathering. In *Proceedings of DISC 2006*, pages 46–60, 2006.

[80] B. Degener, B. Kempkes, T. Langner, F. Meyer auf der Heide, P. Pietrzyk, and R. Wattenhofer. A tight runtime bound for synchronous gathering of

autonomous robots with limited visibility. In *Proceedings of the twenty-third annual ACM symposium on Parallelism in algorithms and architectures*, pages 139–148. ACM, 2011.

[81] E. D. Demaine, S. P. Fekete, and S. Gal. Online searching with turn cost. *Theoretical Computer Science*, 361(2):342–355, 2006.

[82] X. Deng, T. Kameda, and C. Papadimitriou. How to learn an unknown environment. In *Proceedings of FOCS*, pages 298–303. IEEE, 1991.

[83] 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.

[84] D. Dereniowski, Y. Disser, A. Kosowski, D. Pająk, and P. Uznański. Fast collaborative graph exploration. *Information and Computation*, 243:37–49, 2015.

[85] D. Dereniowski, R. Klasing, A. Kosowski, and Ł. Kuszner. Rendezvous of heterogeneous mobile agents in edge-weighted networks. In *International Colloquium on Structural Information and Communication Complexity*, pages 311–326. Springer, 2014.

[86] D. Dereniowski and A. Pelc. Drawing maps with advice. *Journal of Parallel and Distributed Computing*, 72(2):132–143, 2012.

[87] A. Dessmark, P. Fraigniaud, D. R. Kowalski, and A. Pelc. Deterministic rendezvous in graphs. *Algorithmica*, 46(1):69–96, 2006.

[88] A. Dessmark and A. Pelc. Optimal graph exploration without good maps. *Theoretical Computer Science*, 326(1-3):343–362, 2004.

[89] G. A. Di Luna, P. Flocchini, S. G. Chaudhuri, F. Poloni, N. Santoro, and G. Viglietta. Mutual visibility by luminous robots without collisions. *Information and Computation*, 2016.

[90] Y. Dieudonné and A. Pelc. Anonymous meeting in networks. *Algorithmica*, 74(2):908–946, 2016.

[91] Y. Dieudonné, A. Pelc, and D. Peleg. Gathering despite mischief. *ACM Transactions on Algorithms (TALG)*, 11(1):1, 2014.

[92] Y. Dieudonné, A. Pelc, and V. Villain. How to meet asynchronously at polynomial cost. *SIAM Journal on Computing*, 44(3):844–867, 2015.

[93] K. Diks, P. Fraigniaud, E. Kranakis, and A. Pelc. Tree exploration with little memory. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 588–597. Society for Industrial and Applied Mathematics, 2002.

[94] S. Dobrev, P. Flocchini, R. Královič, P. Ružička, G. Prencipe, and N. Santoro. Black hole search in common interconnection networks. *Networks*, 47(2):61–71, 2006.

[95] S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro. Searching for a black hole in arbitrary networks: Optimal mobile agents protocols. *Distributed Computing*, 19(1):1–99999, 2006.

[96] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. Robotic exploration as graph construction. *IEEE transactions on robotics and automation*, 7(6):859–865, 1991.

[97] C. A Duncan, S. G. Kobourov, and V. S. Kumar. Optimal constrained graph exploration. *ACM Transactions on Algorithms (TALG)*, 2(3):380–402, 2006.

[98] C. Dutta, G. Pandurangan, R. Rajaraman, Z. Sun, and E. Viola. On the complexity of information spreading in dynamic networks. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 717–736. SIAM, 2013.

[99] M. Dynia, J. ŁopuszaŃski, and C. Schindelhauer. Why robots need maps. In *International Colloquium on Structural Information and Communication Complexity*, pages 41–50. Springer, 2007.

[100] H. A. Eiselt, M. Gendreau, and G. Laporte. Arc routing problems, part ii: The rural postman problem. *Operations research*, 43(3):399–414, 1995.

[101] Y. Elor and A. M. Bruckstein. Uniform multi-agent deployment on a ring. *Theoretical Computer Science*, 412(8-10):783–795, 2011.

[102] S. Elouasbi and A. Pelc. Deterministic rendezvous with detection using beeps. *International Journal of Foundations of Computer Science*, 28(01):77–97, 2017.

[103] Y. Emek, T. Langner, D. Stolz, J. Uitto, and Wattenhofer R. How many ants does it take to find the food? *Theor. Comput. Sci.*, 608:255–267, 2015.

[104] Y. Emek, T. Langner, D. Stolz, J. Uitto, R. Wattenhofer, and I. Technion. Towards more realistic ants.

[105] Y. Emek, T. Langner, J. Uitto, and R. Wattenhofer. Ants: Mobile finite state machines. *arXiv preprint arXiv:1311.3062*, 2013.

[106] Y. Emek, T. Langner, J. Uitto, and R. Wattenhofer. Solving the ANTS problem with asynchronous finite state machines. In *Automata, Languages, and*

Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II, pages 471–482, 2014.

[107] P. Fazli, A. Davoodi, and A. K. Mackworth. Multi-robot repeated area coverage. *Autonomous robots*, 34(4):251–276, 2013.

[108] O. Feinerman and A. Korman. Memory lower bounds for randomized collaborative search and implications for biology. In *International Symposium on Distributed Computing*, pages 61–75. Springer, 2012.

[109] O. Feinerman, A. Korman, S. Kutten, and Y. Rodeh. Fast rendezvous on a cycle by agents with different speeds. In *Distributed Computing and Networking - 15th International Conference, ICDCN 2014, Coimbatore, India, January 4-7, 2014. Proceedings*, pages 1–13, 2014.

[110] 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.

[111] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.

[112] R. Fleischer and G. Trippen. Exploring an unknown graph efficiently. In *European Symposium on Algorithms*, pages 11–22. Springer, 2005.

[113] P. Flocchini. Time-varying graphs and dynamic networks. In *2015 Summer Solstice: 7th International Conference on Discrete Models of Complex Systems*, 2015.

[114] P. Flocchini, D. Ilcinkas, A. Pelc, and N. Santoro. Computing without communicating: Ring exploration by asynchronous oblivious robots. In *International Conference On Principles Of Distributed Systems*, pages 105–118. Springer, 2007.

[115] P. Flocchini, D. Ilcinkas, A. Pelc, and N. Santoro. Remembering without memory: Tree exploration by asynchronous oblivious robots. In *International Colloquium on Structural Information and Communication Complexity*, pages 33–47. Springer, 2008.

[116] P. Flocchini, D. Ilcinkas, and N. Santoro. Ping pong in dangerous graphs: Optimal black hole search with pebbles. *Algorithmica*, 62(3-4):1006–1033, 2012.

[117] P. Flocchini, M. Kellett, P. Mason, and N. Santoro. Map construction and exploration by mobile agents scattered in a dangerous network. In *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–10. IEEE, 2009.

[118] P. Flocchini, B. Mans, and N. Santoro. Exploration of periodically varying graphs. In *International Symposium on Algorithms and Computation*, pages 534–543. Springer, 2009.

[119] P. Flocchini, B. Mans, and N. Santoro. On the exploration of time-varying networks. *Theoretical Computer Science*, 469:53–68, 2013.

[120] P. Flocchini, G. Prencipe, N. Santoro, and G. Viglietta. Distributed computing by mobile robots: uniform circle formation. *Distributed Computing*, pages 1–45, 2014.

[121] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Gathering of asynchronous robots with limited visibility. *Theoretical Computer Science*, 337(1):147 – 168, 2005.

[122] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Arbitrary pattern formation by asynchronous, anonymous, oblivious robots. *Theoretical Computer Science*, 407(1-3):412–447, 2008.

[123] P. Flocchini, N. Santoro, G. Viglietta, and M. Yamashita. Rendezvous of two robots with constant memory. In *SIROCCO*, pages 189–200. Springer, 2013.

[124] F. V. Fomin, P. A. Golovach, J. Kratochvíl, N. Nisse, and K. Suchan. Pursuing a fast robber on a graph. *Theoretical Computer Science*, 411(7):1167–1181, 2010.

[125] F. V. Fomin and D. M. Thilikos. An annotated bibliography on guaranteed graph searching. *Theoretical Computer Science*, 399(3):236–245, 2008.

[126] P. Fraigniaud, L. Gasieniec, D. R. Kowalski, and Pelc A. Collective tree exploration. *Networks*, 48(3):166–177, 2006.

[127] P. Fraigniaud and D. Ilcinkas. Digraphs exploration with little memory. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 246–257. Springer, 2004.

[128] P. Fraigniaud, D. Ilcinkas, G. Peer, A. Pelc, and D. Peleg. Graph exploration by a finite automaton. *Theoretical Computer Science*, 345(2-3):331–344, 2005.

[129] P. Fraigniaud, D. Ilcinkas, S. Rajsbaum, and S. Tixeuil. Space lower bounds for graph exploration via reduced automata. In *International Colloquium on Struc-*

tural Information and Communication Complexity, pages 140–154. Springer, 2005.

[130] P. Fraigniaud, A. Korman, and Y. Rodeh. Parallel exhaustive search without coordination. *arXiv preprint arXiv:1511.00486*, 2015.

[131] P. Fraigniaud and A. Pelc. Deterministic rendezvous in trees with little memory. In *International Symposium on Distributed Computing*, pages 242–256. Springer, 2008.

[132] P. Fraigniaud and A. Pelc. Decidability classes for mobile agents computing. In *Latin American Symposium on Theoretical Informatics*, pages 362–374. Springer, 2012.

[133] A. Frieze, M. Krivelevich, and P. Loh. Variations on cops and robbers. *Journal of Graph Theory*, 69(4):383–402, 2012.

[134] D. W. Gage. Randomized search strategies with imperfect sensors. In *Optical Tools for Manufacturing and Advanced Automation*, pages 270–279. International Society for Optics and Photonics, 1994.

[135] M. R. Garey and D. S. Johnson. Two-processor scheduling with start-times and deadlines. *SIAM Journal on Computing*, 6(3):416–426, 1977.

[136] M. R. Garey and D. S. Johnson. *Computers and intractability*, volume 29. W. H. Freeman New York, 2002.

[137] L. Gąsieniec, R. Klasing, R. Martin, A. Navarra, and X. Zhang. Fast periodic graph exploration with constant memory. *Journal of Computer and System Sciences*, 74(5):808–822, 2008.

[138] L. Gasieniec, A. Pelc, T. Radzik, and X. Zhang. Tree exploration with logarithmic memory. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 585–594. Society for Industrial and Applied Mathematics, 2007.

[139] M. Ghaffari, C. Musco, T. Radeva, and N. Lynch. Distributed house-hunting in ant colonies. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, pages 57–66. ACM, 2015.

[140] F. Hoffmann, C. Icking, R. Klein, and K. Kriegel. The polygon exploration problem. *SIAM Journal on Computing*, 31(2):577–600, 2001.

[141] R. Holte, A. Mok, L. Rosier, I. Tulchinsky, and D. Varvel. The pinwheel: A real-time scheduling problem. In *System Sciences, 1989. Vol. II: Software Track, Proceedings of the Twenty-Second Annual Hawaii International Conference on*, volume 2, pages 693–702. IEEE, 1989. Also, in Handbook of Scheduling Algorithms, Models, and Performance Analysis, CRC Press, 2004.

[142] J. Hromkovič, R. Klasing, B. Monien, and R. Peine. Dissemination of information in interconnection networks (broadcasting & gossiping). In *Combinatorial network theory*, pages 125–212. Springer, 1996.

[143] E. Huus and E. Kranakis. Rendezvous of many agents with different speeds in a cycle. In *Ad-hoc, Mobile, and Wireless Networks - 14th International Conference, ADHOC-NOW 2015, Athens, Greece, June 29 - July 1, Proceedings*, pages 195–209, 2015.

[144] T. Izumi, Y. Katayama, N. Inuzuka, and K. Wada. Gathering autonomous mobile robots with dynamic compasses: An optimal result. In *International Symposium on Distributed Computing*, pages 298–312. Springer, 2007.

[145] T. Izumi, S. Souissi, Y. Katayama, N. Inuzuka, X. Défago, K. Wada, and M. Yamashita. The gathering problem for two oblivious robots with unreliable compasses. *SIAM Journal on Computing*, 41(1):26–46, 2012.

[146] Y. Jin, Y. Liao, A. A Minai, and M. M. Polycarpou. Balancing search and target response in cooperative unmanned aerial vehicle (uav) teams. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36(3):571–587, 2005.

[147] D. S. Johnson. The NP-completeness column: an ongoing guide. *Journal of Algorithms*, 6(3):434–451, 1985.

[148] B. Kalyanasundaram and K. R. Pruhs. Constructing competitive tours from local information. *Theoretical Computer Science*, 130(1):125–138, 1994.

[149] M. Kao, J. H. Reif, and S. R. Tate. Searching in an unknown environment: An optimal randomized algorithm for the cow-path problem. *Information and Computation*, 131(1):63–79, 1996.

[150] A. Kawamura and Y. Kobayashi. Fence patrolling by mobile agents with distinct speeds. In *Distributed Computing*, volume 28:2, pages 147–154, 2015.

[151] R. Klasing, A. Kosowski, and A. Navarra. Taking advantage of symmetries: Gathering of many asynchronous oblivious robots on a ring. *Theoretical Computer Science*, 411(34-36):3235–3246, 2010.

[152] R. Klasing, E. Markou, and A. Pelc. Gathering asynchronous oblivious mobile robots in a ring. *Theoretical Computer Science*, 390(1):27–39, 2008.

[153] R. Klein. Walking an unknown street with bounded detour. *Computational Geometry*, 1(6):325–351, 1992.

[154] J. Kleinberg. On-line search in a simple polygon. In *Proceedings of SODA*, pages 8–15. SIAM, 1994.

[155] J. M. Kleinberg. The localization problem for mobile robots. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 521–531. IEEE, 1994.

[156] S. Koenig, C. Tovey, and W. Halliburton. Greedy mapping of terrain. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 4, pages 3594–3599. IEEE, 2001.

[157] A. Kolling and S. Carpin. Pursuit-evasion on trees by robot teams. *IEEE Transactions on Robotics*, 26(1):32–47, 2010.

[158] A. Korman and Y. Rodeh. Parallel linear search with no coordination for a randomly placed treasure. *arXiv preprint arXiv:1602.04952*, 2016.

[159] D. R. Kowalski and A. Malinowski. How to meet in anonymous network. *Theoretical Computer Science*, 399(1-2):141–156, 2008.

[160] E. Kranakis, D. Krizanc, F. MacQuarrie, and S. Shende. Randomized rendezvous algorithms for agents on a ring with different speeds. In *Proceedings of the 2015 International Conference on Distributed Computing and Networking, ICDCN 2015, Goa, India, January 4-7*, pages 9:1–9:10, 2015.

[161] E. Kranakis, D. Krizanc, and P. Morin. Randomized rendez-vous with limited memory. In *Latin American Symposium on Theoretical Informatics*, pages 605–616. Springer, 2008.

[162] E. Kranakis, E. Krizanc, D. Markou, A. Pagourtzis, and F. Ramirez. Two different speeds suffice for rendezvous in arbitrary graphs. In *Proceedings of*

the 43rd International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM) January 16–20, Lero–Limerick, Ireland, 2017.

[163] S. Kreutzer and S. Ordyniak. Digraph decompositions and monotonicity in digraph searching. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 336–347. Springer, 2008.

[164] F. Kuhn, N. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 513–522. ACM, 2010.

[165] A. Kumar, S. Sharma, R. Tiwari, and S. Majumdar. Area exploration by flocking of multi robot. *Procedia Engineering*, 41:377–382, 2012.

[166] S. Kwek. On a simple depth-first search strategy for exploring unknown graphs. In *Workshop on Algorithms and Data Structures*, pages 345–353. Springer, 1997.

[167] L. Lamport. The weak byzantine generals problem. *Journal of the ACM (JACM)*, 30(3):668–676, 1983.

[168] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[169] I. Lamprou, R. Martin, and S. Schewe. Fast two-robot disk evacuation with wireless communication. In *International Symposium on Distributed Computing*, pages 1–15. Springer, 2016.

[170] E. L. Lawler. Optimal sequencing of a single machine subject to precedence constraints. *Management science*, 19(5):544–546, 1973.

[171] C. Lenzen, N. Lynch, C. Newport, and T. Radeva. Trade-offs between selection complexity and performance when searching the plane without communication. In *Proceedings of the 2014 ACM symposium on Principles of distributed computing*, pages 252–261. ACM, 2014.

[172] A. Q. Li, F. Amigoni, and N. Basilico. Searching for optimal off-line exploration paths in grid environments for a robot with limited visibility. In *AAAI*, 2012.

[173] A. López-Ortiz and S. Schuierer. On-line parallel heuristics, processor scheduling and robot searching under the competitive framework. *Theoretical Computer Science*, 310(1-3):527–537, 2004.

[174] A. Lopez-Ortiz and G. Sweet. Parallel searching on a lattice. In *Proceedings of the 13th Canadian Conference on Computational Geometry*, 2001.

[175] T. Łuczak and P. Prałat. Chasing robbers on random graphs: Zigzag theorem. *Random Structures & Algorithms*, 37(4):516–524, 2010.

[176] N. A. Lynch. *Distributed algorithms*. Morgan Kaufmann, 1996.

[177] N. Megow, K. Mehlhorn, and P. Schweitzer. Online graph exploration: New results on old and new algorithms. *Theoretical Computer Science*, 463:62–72, 2012.

[178] S. Mitrovic-Minic and R. Krishnamurti. The multiple traveling salesman problem with time windows: Bounds for the minimum number of vehicles. *Simon Fraser University TR-2002-11*, 2002.

[179] C. O'Brien. *Solving ANTS with loneliness detection and constant memory*. PhD thesis, Massachusetts Institute of Technology, 2015.

[180] C. H Papadimitriou and M. Yannakakis. Shortest paths without a map. In *Proceedings of ICALP, LNCS*, volume 372, pages 610–620. Springer, 1989.

[181] A. Pelc. Deterministic rendezvous in networks: A comprehensive survey. *Networks*, 59(3):331–347, 2012.

[182] M. Potop-Butucaru, M. Raynal, and S. Tixeuil. Distributed computing with mobile robots: an introductory survey. In *Network-Based Information Systems (NBiS), 2011 14th International Conference on*, pages 318–324. IEEE, 2011.

[183] G. Prencipe. Impossibility of gathering by a set of autonomous mobile robots. *Theoretical Computer Science*, 384(2-3):222–231, 2007.

[184] G. Prencipe and N. Santoro. Distributed algorithms for autonomous mobile robots. In *Fourth IFIP International Conference on Theoretical Computer Science-TCS 2006*, pages 47–62. Springer, 2006.

[185] O. Reingold. Undirected connectivity in log-space. *Journal of the ACM (JACM)*, 55(4):17, 2008.

[186] R. Reischuk. A new solution for the byzantine generals problem. *Information and Control*, 64(1-3):23–42, 1985.

[187] I. M. Rekleitis, G. Dudek, and E. E. Milios. Multi-robot collaboration for robust exploration. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 4, pages 3164–3169. IEEE, 2000.

[188] N. Santoro, W. Quattrociocchi, P. Flocchini, A. Casteigts, and F. Amblard. Time-varying graphs and social network analysis: Temporal indicators and metrics. *arXiv preprint arXiv:1102.0629*, 2011.

[189] W. Sheng, Q. Yang, J. Tan, and N. Xi. Distributed multi-robot coordination in area exploration. *Robotics and Autonomous Systems*, 54(12):945–955, 2006.

[190] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes. Coordination for multi-robot exploration and mapping. In *AAAI/IAAI*, pages 852–858, 2000.

[191] S. Souissi, X. Défago, and M. Yamashita. Gathering asynchronous mobile robots with inaccurate compasses. *Principles of Distributed Systems*, pages 333–349, 2006.

[192] Langner T., Uitto J., Stolz D., and Wattenhofer R. Fault-tolerant ANTS. In *Distributed Computing - 28th International Symposium, DISC 2014, Austin, TX, USA, October 12-15, 2014. Proceedings*, pages 31–45, 2014.

[193] A. Ta-Shma and U. Zwick. Deterministic rendezvous, treasure hunts and strongly universal exploration sequences. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 599–608. Society for Industrial and Applied Mathematics, 2007.

[194] C. J. Taylor and D. J. Kriegman. Vision-based motion planning and exploration algorithms for mobile robots. *IEEE Transactions on robotics and Automation*, 14(3):417–426, 1998.

[195] H. Thimbleby. The directed chinese postman problem. *Software: Practice and Experience*, 33(11):1081–1096, 2003.

[196] S. Thrun. A probabilistic on-line mapping algorithm for teams of mobile robots. *The International Journal of Robotics Research*, 20(5):335–363, 2001.

[197] S. Thrun et al. Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium*, 1:1–35, 2002.

[198] J. N. Tsitsiklis. Special cases of traveling salesman and repairman problems with time windows. *Networks*, 22(3):263–282, 1992.

[199] L. A. Tychonievich and J. P. Cohoon. Coalescing swarms of limited capacity agents: Meeting and staying together(without trust). *IAENG International Journal of Computer Science*, 39(3):254–260, 2012.

[200] G. Viglietta. Rendezvous of two robots with visible bits. In *International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics*, pages 291–306. Springer, 2013.

[201] M. L. Visinsky, J. R. Cavallaro, and I. D. Walker. Robotic fault detection and fault tolerance: A survey. *Reliability Engineering & System Safety*, 46(2):139–158, 1994.

[202] I. A Wagner, Y. Altshuler, V. Yanovski, and A. M. Bruckstein. Cooperative cleaners: A study in ant robotics. *The International Journal of Robotics Research*, 27(1):127–151, 2008.

[203] G. Wang, M. J. Irwin, H. Fu, P. Berman, W. Zhang, and T. La Porta. Optimizing sensor movement planning for energy efficiency. *ACM Transactions on Sensor Networks*, 7(4):33, 2011.

[204] K. Wehmuth, A. Ziviani, and E. Fleury. A unifying model for representing time-varying graphs. *arXiv preprint arXiv:1402.3488*, 2014.

[205] M. Yamashita and I. Suzuki. Characterizing geometric patterns formable by oblivious anonymous mobile robots. *Theoretical Computer Science*, 411(26-28):2433–2453, 2010.

[206] B. Yamauchi. Frontier-based exploration using multiple robots. In *Proceedings of 2nd international conference on Autonomous agents*, pages 47–53. ACM, 1998.

[207] Y. Yang, S. Souissi, X. Défago, and M. Takizawa. Fault-tolerant flocking for a group of autonomous mobile robots. *Journal of Systems and Software*, 84(1):29–36, 2011.

[208] G. H. Young and C.-L. Chan. Single-vehicle scheduling with time window constraints. *Journal of Scheduling*, 2(4):175–187, 1999.

[209] R. Zlot, A. Stentz, M. B. Dias, and S. Thayer. Multi-robot exploration controlled by a market economy. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 3, pages 3016–3023. IEEE, 2002.