Swarms of Bouncing Robots

by

Eduardo Pacheco

A thesis submitted to the Faculty of Graduate Studies and Postdoctoral Affairs
in Partial Fullfilment of the Requirements for the Degree of

PHD IN COMPUTER SCIENCE

School Of Computer Science

at

Carleton University

Ottawa, Ontario

2

Abstract

We study models of mobile robots with limited capabilities that are deployed either on a cycle or an infinite line or on a segment. Robots start moving at the same time and when two robots collide their speeds and movement directions are instantaneously updated. Each of them has a collision detector and a clock to measure the times of its collisions. They do not have any knowledge on the total number of robots and do not have a common sense of direction. Besides, they neither have visibility nor control over their movements.

We investigate the feasibility of the localization task in the cycle and the segment by bouncing robots: every robot should figure out the starting position and initial velocity of all the other robots. We consider two different scenarios when robots have common masses and speeds and robots of arbitrary masses and speeds. We give complete characterizations of all feasible configurations for the cycle in both scenarios.

We study the survivability of bouncing robots. We say a robot survives if it never returns to its starting position. Non-surviving robots disappear from the environment. We provide sufficient and necessary conditions to have surviving robots in the cycle and in the segment. Finally we investigate communication protocols for bouncing robots that only communicate at the time of their collisions. We establish necessary and sufficient conditions for bouncing robots to perform gossiping, broadcasting and convergecast.

Contents

List of Figures

CHAPTER 1

Introduction

For a long time, robots have been seen as a potential tool that might improve our lives. The idea that some entity performs a task on our behalf has been very attractive to us. Today, the use of robots has somewhat made our lives easier by allowing us to explore unknown and hazardous environments and to perform daily activities efficiently and safely. The Curiosity project [1] and bomb disposal robots are some examples of this.

Sometimes, a task is more efficiently performed by a group of simple robots than by just one complex robot. However, the use of a set of robots for performing a single task raises some difficulties regarding coordination and communication. Overcoming such difficulties is of great importance due to the increasing use of such systems in relevant human activities. In this work, we aim to improve our understanding of *systems of mobile robots* and we also aim to contribute to overcome such difficulties. We study these systems from the algorithmic perspective of distributed computing, i.e, in this thesis we design correct and efficient algorithms for systems of mobile robots.

A Taste of History. Back in the 1970's, Distributed Artificial Intelligence (DAI) was introduced as the study, construction, and application of multi-agent systems in which *intelligent* agents collaborate to perform some set of tasks [7, 56]. Due to the emerging of new technologies together with the increasing use of complex computational systems DAI rapidly evolved and diversified giving birth to other research fields like mobile agent computing and computational organization. One of the main contributions of this field was the introduction of the concept of *agent* in computer science. An agent was conceived as an *entity that can sense its environment and* *act upon it* [56] as well as an entity that is *rational*, *deliberative*, and possesses some sort of will [7]. In this work, we consider a mobile robot in its simplest form as an agent that can sense, move within its environment, do computations and act upon its environment. We use the words agent and robot interchangeably. As we said above, our purpose is to study mobile agents from the perspective of distributed computing.

Distributed computing. Nowadays, computing systems are mainly distributed, heterogeneous, larger, and much more complex than they used to be just few decades ago. Some examples of distributed systems are multi-core machines, computer networks, swarms, and mobile sensors [28]. In all these modern computing systems, computers behave more as independent agents to fulfill their purpose [38].

Distributed computing is the field of computer science that studies the computational issues that arise in distributed systems. Hence, distributed computing encompasses the field of *mobile agent computing*; mobile agent computing studies the computational and complexity issues in distributed systems whose agents have the ability of *moving* within their environment. It is important to notice that such an ability is not restricted only to physical agents (like robots), but it also characterizes some software. However, in this proposal, we will focus only on systems of mobile agents which due to their lack of a central authority are also known as systems of autonomous agents.

Recently, mobile agent computing has been the subject of increasing interest. Researchers of different fields from artificial intelligence, software engineering, computational economics, and robotics have investigated the set of tasks that can be computed by mobile agents. Part of the reason for such an increasing interest is due to the potential advantages that mobile agents seem to have.

Some advantages of systems of mobile agents are: *efficiency*, by doing a task where a single agent solution may be expensive or even impossible; *fault tolerance*, in the presence of the failure of one agent the remaining agents still can complete their task; *flexibility*, it is usually easy for agents to adapt their behavior in the presence of changes of their environment [38]. For these reasons, mobile agents are used for electronic commerce, robotic exploration, network maintenance, etc. So mobile agents seem to be a useful tool to design, implement, and maintain distributed systems. Yet they are not a *panacea* [39].

Our Assumptions and Goals. In this thesis, we are interested in studying systems of autonomous robots from an algorithmic perspective. We think of a mobile robot as an autonomous entity that has the ability to perceive some parameters of its environment (*sensing*); the ability to receive or transmit information to other robots (*communication*); the ability to move within their environment (*mobility*); the ability to remember previously collected data (*storage*) and the ability to process such data (*computing*). We assume that robots are *deployed* in some environment with the purpose of carrying out some specific task. In order to do so, robots have to coordinate their actions with other robots. In some cases, robots will be able to communicate among themselves, if such is the case robots will have restricted communication capabilities. We will discuss more about the different mechanisms for implementing communication in a system of mobile robots in the following section.

We focus on studying how a system of autonomous robots can overcome some difficulties while performing a task. We are also interested in understanding the algorithmic limitations that their restricted capabilities impose on them. More precisely, we try to comprehend which tasks can be performed by mobile robots, under what conditions, and the cost of performing such tasks. To do so, we study models of mobile robots that assume *weak robots*, i.e, robots with extremly limited capabilities.

An algorithm for a system of robots is a distributed algorithm, such that, given a model robot, a model of environment, and a given task, it specifies the steps that should be executed by the robots to successfully complete the given task. The measure of efficiency (complexity) of an algorithm is closely related to the models of robot and environment. Bandwidth, memory, time, power consumption, and traveled distance, are examples of some measures taken into account when designing a distributed algorithm for a system of mobile robots. We will provide examples and discuss this in the following chapters.

There are many intriguing and appealing research questions about mobile robots: How should a group of robots coordinate in order to carry out a task? How simple can robots be? how much knowledge should robots have access to in order to solve a task? In this thesis, we aim to answer some of these questions. More specifically, we address the second of these questions by proposing a new model of mobile robots that mimic the behavior of gas particles moving in a one dimensional environment. They neither have visibility nor have control of their movements. We show that such robots are able to perform several tasks like self deployment, patrolling, and position discovery. As far as we know, the model of our results is novel although it has some similarities with some models of gas particles, for instance in [36]. We also address related problems that we came to encounter after we proposed our model like the *salmon problem* and study communications protocols performed by these robots.

Outline. In Chapter 2, we survey some models of mobile robots and we provide an overview of the state of the art of the field. We also motivate the models that we propose. In Chapter 3, we introduce our model of mobile robots that we call *bouncing robots* in two different flavors. We then study the task of *localization* also known as *position discovery* by bouncing robots. In our first model, robots move with the same speed while in the second one they have arbitrary speeds. We study the localization for one dimensional environments. We give algorithms to carry out localization and provide full characterizations of all feasible configurations. In Chapter 4, we study the survivability of bouncing robots that are deployed in a dangerous environment with deadly locations. When a robot visits some of such locations it is destroyed. In all those chapters we assume that robots can not communicate by any means however in Chapter 5 we allow bouncing robots to communicate and study different communication protocols carried out by them. We conclude our study in Chapter 6 by summarizing our results and stating some open problems.

CHAPTER 2

Mobile Agent Computing

In this section, we review some of the most relevant and related literature of mobile robots. We discuss different models of mobile robots, their environment, and the different timing assumptions in which robots are commonly assumed to operate. Our goal in this section is briefly describe how all these assumptions impact the efficiency of mobile robot algorithms and their feasibility. We also want to motivate the model of mobile robots that we propose which is a model that assumes robots with extremely limited capabilities. Our model turns out to be similar to some systems of particles in classical mechanics. For this reason, we describe some of the work done in physics for those systems. In the last section, we discuss some problems that have received the attention of many researchers.

1. Classical Models

Robots. Most models of mobile robots assume *homogeneity*, i.e, all robots have the same set of capabilities, and *anonymity*, meaning that robots are not labeled with unique identifiers. These two restrictions impose on *all* robots the execution of the same algorithm. On the other hand, *heterogeneity* assumes that robots do not have the same capabilities. Thus, in heterogeneous systems, not all robots participate in the same way while solving a task. Such a constraint raises a series of interesting questions related to the assignment of tasks and modeling of robots [**42**, **43**]. In this thesis, however, we are interested in homogeneous systems only. Part of the reason for this is the increasing use, due to the low cost production, of *swarms*. Swarms are large collections heterogeneous simple robots with very restricted capabilities that are able to replace a system of a few but complex robots. Swarms are becoming a very popular tool in distributed computing because they are able to show and exploit very complex behavior [7].

Usually, robots are assumed to possess a *system of coordinates* and a *compass* that allow them to navigate within their environment. A robot with such ability is said to have a *sense of direction* [28]. The weakest models of mobile robots assume no consistent sense of direction among the robots. Normally, researchers by sense of direction assume that the robots are masters of their own movements, meaning that robots have total control of their movements, so by using their compass and system of coordinates they move as they wish within their environment of deployment.

On the other hand, in a few works, it is assumed that robots have sense of direction but they do not have control of their movements. In those models, sense of direction is more like a system that allows robots to have some knowledge of their position, and robots move according to some pattern totally out of their control. This type of movement is called *passive mobility* [3, 4]. Since we are interested in studying extremely weak robots, we assume robots with passive mobility and no common system of coordinates. An interesting research question is whether robots are able to figure out the system of coordinates of all robots by performing a minimum number of operations. Clearly, in a system of robots with common sense of direction coordination is easier to obtain. Figure 2.1 depicts two robots, deployed in the plane, with different sense of direction.

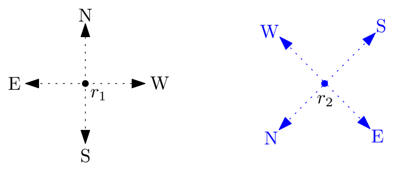

FIGURE 2.1. robots r_1 and r_2, deployed on the plane, have different sense of direction. What is north for r_1 is south east to r_2.

Interaction. Robots should be able to interact among themselves to solve any task. There are different ways to achieve this. For instance, robots can interact via their environment by shared resources (like memory) [7]. However, the most common way of getting interaction is via *sensing*. In this case, robots have sensors that allow them to sense other robots and some parameters of their environment. They do so without explicit communication. In the literature, the most common assumed sensing capability is *visibility*. Visibility might be either unlimited or limited. A robot with unlimited visibility is able to get a *snapshot* of the entire system while one with a limited visibility can only get the information within some bounded range. Sensing assumptions have relevance because they establish limits on the collaboration among robots. Visibility has been used to perform *flocking* and *pattern formation* [7, 52]. Another way of interaction among robots is via explicit communication, i.e, by exchanging messages. This type of interaction involves the design of communication protocols similar to those in computer networks. We discuss more about communication protocols later on in this section. It suffices to say that we are only interested in studying interaction that involves simple communication.

A closely related ability to visibility is the one concerning the amount of memory that robots have. If robots have memory of constant size, they only can *remember* a bounded number of past events. Such robots are called *oblivious*. So robots with limited memory are not aware of all their actions in the past. This assumption is very important when modeling the power of computing of robots. Oblivious robots may be modeled as simple finite state automata [19] while, non-oblivious robots are modeled as Turing machines. Besides this, in many applications, memory is always of great concern. Hence, algorithms that use oblivious robots are frequently sought.

Robot Representation. In most of the theoretical works, robots are commonly modeled as mere points. In such models, it is plausible that robots can gather (or pile up) at one point. Examples of this might be found extensively (see [28, 29, 30, 38, 52]). There are few works that consider *fat robots*. Fat robots robots are not dimensionless and they are frequently modeled with unit discs. In any case, there are difficulties to overcome. In the former model, a robot can not distinguish if a location is occupied by one or by many robots; in the latter one, robots can block their visibility. In Figure 2.2, fat robots r_a and r_b are not able to see each other.

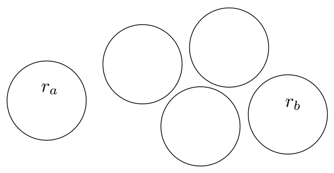

FIGURE 2.2. Fat robots r_a and r_b can not see each other

In [17], Czyzowicz et.al studied the problem of *gathering* fat robots in the plane. For this task, robots are required to get close enough to each other without colliding. Czyzowicz et.al devised algorithms to gather up to four fat robots. However, if robots are dimensionless, there exists a simple algorithm in which any number of robots can gather at one point, for instance, at the center of mass of the system [9]. No algorithm is known yet for gathering more than four fat robots. The gathering problem is an example of a task that becomes very hard when a simple variation on the representation of a robot is introduced. For simplicity, in this thesis, we model robots as points although we are aware that fat robots represent a more *realistic* setting.

Environment. Of great relevance, when defining a model of mobile robots, is the description of the environment or universe in which robots are to be deployed. We found in the literature two settings: robots operating in a *discrete universe*; in this setting, robots are deployed on a communication network which is modeled by a connected discrete graph. Agents hop between adjacent nodes of the graph collecting information in each node and being able to resume their computations in the new node. This setting is frequently used to study network maintenance, e-commerce, etc. [38]; In the *continuous* model, robots freely move on a terrain or surface, some examples that assume this model might be found in [16, 33]. The description of the world in which robots move is important since it makes a difference in the way that the efficiency of algorithms is measured. For instance, in discrete universes the number of hops that a robot performs to complete its task might be a way to measure the efficiency of the protocol while in a continuous setting the total distance traversed is of more interest. In this thesis, we focus on studying robots that move within a one dimensional and continuous universe. We try to *compensate* the simplicity of the universe by considering robots with extremely restricted capabilities.

Timing. In distributed computing, the assumptions on the timing within systems operate are very important. It is well known that the timing assumptions affect the feasibility of some problems. The most popular models are the synchronous, semi-synchronous, and asynchronous.

In synchronous systems, it is assumed the existence of a global clock that allows the agents of the system to execute their operations simultaneously in rounds. So the less rounds the better an algorithm is. On the other hand, asynchronous systems are commonly modeled by assuming an adversary that schedules the operations of the agents of the system. Agents in asynchronous systems have to deal with some grade of uncertainty which depends on the model of the adversary [5].

By describing the timing in which robots perform their operations we can extend these timing concepts to mobile robot systems. There are different alternatives to do so. To illustrate our discussion so far, we will briefly describe some popular and important models of mobile robots in the literature.

Communication for Systems of Mobile Robots. The type of communication allowed for a collection of mobile robots plays an important role in determining the way that robots can interact. Moreover, communication enhances their capabilities and effectiveness. Different models of communication for systems of mobile robots have been studied (cf. [7, 22]) and they can be classified within any of the three following categories: communication via their *environment*, for instance, via tokens or pebbles that robots are allowed to drop on the environment (e.g. [6, 13]); communication by *sensing* each other using, for instance, some visibility mechanism (e.g. [50, 9]); and communication by passing *messages*. The latter type is the most common one assumed for wireless sensor systems in which for a sensor to receive a message it has to be within the range of transmission of another sensor. The main message passing communication problems concern *broadcasting* - when the message of one robot has to reach all other ones, *convergecast* - when the initial information of all robots has to reach one of them and *gossiping* - when each robot has to inform everybody else.

A Bit of History and Examples. The first model of mobile robots that was introduced with the purpose of studying mobile robots from a theoretical point of view was the semi-synchronous model (SSYNC) due to Suzuki and Yamashita in the 1990's [52]. Since then, several variations of the this model have been proposed. Possibly, the most popular one is found in [30] due to Flochini et.al also known as Coordination and control of a set of Robots in a Distributed and Asynchronous environment (CORDA).

SSYNC and CORDA differ in the timing at which their robots operate [44]. However, both models assume that robots execute *look-compute-move* cycles. A look-compute-move cycle (lcm-cycle for short) is composed of three stages: *look*, *compute*, and *move*. A robot, during its stage *look*, observes its environment and collects data using its sensors. During the *compute* stage, a robot, based on its collected information computes a new position. Finally, during its *move* stage a robot moves to the new computed position.

In these models, it is assumed that robots have total control of their movements as well as the ability to (potentially) interact with any other robot. Robots are deployed on the plane and do not have a common sense of direction.

Regarding the timing assumptions of these models, we have three variations: the synchronous one fully-synchronous model (FSYNC) assumes that all robots run in fully synchronous rounds; all robots are activated at the beginning of each round; all of them perform their operations atomically, such that, every robot gets the same snapshot of the system. On the other hand, in the asynchronous version CORDA , robots are activated at different times and the duration of each stage is uncertain. An intermediate model is the semi-synchronous model of Suzuki et.al in which robots act in synchronized rounds, similarly as in FSYNC, but only some robots are activated in each round. There are several difficulties that arise in each of these models, for a detailed discussion about them see [28, 44].

There are some models of robots where the timing assumptions are modeled differently due to more restrictive capabilities of the robots. For instance, in [33], robots do not execute lcm-cycles since they do not have any visibility sensor but a collision sensor. Friedetzky et.al assumed that robots move within a continuous environment represented by a ring of perimeter one. In their model, robots are synchronous in the sense that they start moving at the same time at the beginning of each round, however within each round robots do not synchronize their operations. While some robots are collecting information others might be moving. The results we present in the following chapters assume a similar model.

Some Conclusions. We have seen so far some assumptions on systems of mobile robots, the way that they impact on the feasibility of some tasks and the different complexity measures that they impose. In this thesis, we propose a model where robots are totally blind, such that, they can not see each other nor can get any sort of snapshot of the system at any time. Moreover, in the model that we propose, robots do not have control of their movements, i.e, they have passive mobility. However, they are equipped with a collision sensor that allows them to detect and measure the time when a collision with another robot takes place. The environment on which these robots are deployed is one dimensional, yet we believe that such model of robots is interesting and may provide further understanding on the algorithmic limitations of mobile robots. Toward this goal, in the next section, we discuss some tasks carried out by mobile robots.

2. Tasks for Mobile Robots

Mobile robots can perform many interesting tasks. One of the most important tasks is *pattern formation* introduced by Suzuki and Yamashita. In this task, robots are to form some arbitrary pattern given in advance (for instance a circle) [2, 19, 32, 50, 52]. Other interesting task is flocking, in which robots are to follow a leader while keeping a predetermined formation [7, 34]. Another extensively studied problem is *rendezvous* [17, 31, 38, 47] in which robots are to meet in a specific place of their environment. Recently, *patrolling* [15], *spreading* [10], *self deployment* [24, 27], *motion coordination* [51, 55], and *localization* [16, 18, 33] are being investigated. Many of these problems had been studied for a long time by other communities like the *control* and engineering communities but just recently from a theoretical perspective. On the other hand, there is so much work done in this field that it would be impossible to mention here all of it. Hence, in this thesis, we focus on a simplified version of the localization problem.

While performing exploration with robots, the task of self localization is a basic one. This task consists of a robot localizing itself in its environment of deployment by performing minimum movements. Commonly it is assumed that robots know their coordinates of deployment, however, this is not always the case. Due to its importance for navigation and other tasks, the self localization problem has been of great interest among researchers [23]. The environment of deployment is modeled as a polygon of n vertices without obstacles. Once a robot knows its location, it can perform other tasks like *searching* and *patrolling*. See Figure 2.3.

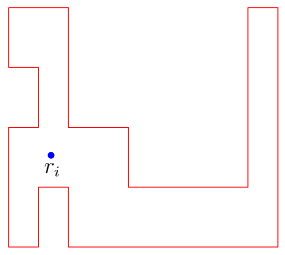

FIGURE 2.3. r_i is deployed within a polygon, its goal is to localize itself in its environment

Regarding the robot capabilities, in this version of the problem, a robot is assumed to have a compass and a range sensing device that allows it to sense the walls or boundaries of its environment as well as their orientation. Besides finding its position, a robot has to minimize its movements to figure out its location. The most important theoretical result that we are aware of is due to Dudek et.al who proved that the problem of self localization is NP-hard [23].

Other variations of the problem include environments with obstacles and probabilistic approximations. Other models, provide the robot with more capabilities like an odometer, cameras, etc. [45]. Most of the recent work that has been done on this problem comes from the robotics community.

Lawrence et.al in [25] consider an alternative model in which the deployed robot has very limited sensing capabilities. The robot only has a clock and a contact sensor in addition to the map of the environment. They experimentally show that the localization problem is feasible in such circumstances by using probabilistic methods.

Notice that this problem is intended to be solved by a single robot that is deployed in a polygon. It might be interesting to consider the self localization problem addressed by a group of robots. If robots can collaborate in order to find their initial positions in their environment, robots may later on perform several coordination tasks.

The first paper that does this is due to Friedetzky et.al in [33]. They consider a set of n anonymous robots deployed on a circle of perimeter one. Robots perform their actions in synchronized rounds. At the beginning of a round they start moving in either clockwise or anti clockwise direction. Robots are not allowed to overpass each other, i.e, they preserve their initial order at any time. When two robots collide, they bounce back. So they move in opposite direction and with the same speed that they had before they collided. Robots are not allowed to perform any sort of communication and are not allowed to leave any mark on the ring. Besides, robots are blind in the sense that they can not see each other.

Robots are to find the initial position in the ring of every other robot without performing explicit communication and only equipped with a collision sensor and a

GPS. Friedetzky et.al devised a fully randomized algorithm that solves the location discovery problem with high probability in $O(\log^2 n)$ rounds.

The models of our results in [16, 18] are partly inspired by Friedetzky's model but we restrict even more the capabilities of the robots. We assume that robots do not have control of their movements, they are only equipped with a clock to measure the time of its collisions and have no GPS available. The resulting model is quite similar to some models studied in physics. More specifically, the model of our results is similar to some models of gas particles that slide on a frictionless surface. We discuss more about these works in the following subsection. Moreover, all the algorithms that we present are deterministic.

3. Classical Mechanics and Distributed Computing

The study of the dynamics of elastic particles sliding on a one dimensional environment has been of great interest in physics for a long time. Much of the work done on this topic has been motivated in order to understand the dynamical properties of gas particles [48, 40, 53, 57]. The dynamics emerging from a collection of particles sliding on an infinite line is very rich and not well understood yet [12]. There are, however, some results concerning the total number of collisions for elastic collisions of particles of arbitrary masses that move within an infinite line. Sevryuk [48] proved that the number of collisions is upper bounded by $2\left(8n^2(n-1)m_{max}/m_{min}\right)^{n-2}$, where n is the total number of particles and m_{max} and m_{min} are the largest and smallest masses of the particles, respectively. When all particles have equal masses the number of collisions is upper bounded by n^2. Other results regarding the number of collisions for different dimensions can be found in [41].

The simplest model of a particle system, that we are aware of, is the one introduced by Jepsen in [36], where he assumes particles of equal mass and arbitrary velocity moving in a frictionless ring. In addition, [36] assumes elastic collisions, in other words, the momentum conservation and the preservation of energy principles are assumed, such that, when two particles collide they simply exchange velocities.

In order to understand some gas equilibrium properties, Jepsen studied the distribution of the initial velocities on the particles. More precisely, for a given time t and initial particle velocity v, he calculates the probability that a given particle has velocity v at time t. Jepsen remarks the simplicity of the dynamics of his system of particles. However, such a simplicity makes it attractive as a starting point in the study of more complex particle systems.

Besides the very interesting physical properties that particle systems may have, they have become attractive to researchers of other fields, among those are computer scientists.

In the distributed computing community, researchers recently studied systems of mobile robots whose dynamics are similar to those of particle systems. Susca et al. [**51**], consider a system of mobile robots that imitate the impact behavior of particles moving in a frictionless ring.

Susca et al. consider a system of n mobile robots moving at different speeds on a ring and colliding elastically. The goal of the robots is to synchronize their movements by synchronizing the times of their collisions and assigning to each of them a unique sector of the ring to traverse. By doing so, robots are able to perform perimeter surveillance. They assume that robots can communicate only with their neighbors and only when they collide, they also assume that robots have control over their speed which they modify at their times of collision, and finally that robots have an absolute position on the ring of which they are aware of. For the case when n is even, and exactly half of the robots move initially in the same direction, they provide a distributed algorithm to perform motion synchronization within a finite amount of time.

Wylie et al. in [**55**] study the motion synchronization task on a segment with relaxed assumptions on the knowledge that robots have. They assume that robots do not know the total number of robots in the ring nor the length of the segment, robots are allowed to communicate and have control over their speeds similarly as in [51]. They prove that robots can motion synchronize if they exchange information about their times of collision. So within a finite amount of time, robots reach a state in which each of them traverses a subsegment of the same length. Recently, Czyzowicz et al. [15] proved that such a strategy is optimal to perform patrolling of a segment. Similar works on motion synchronization can be found in [37].

In all these works, robots make use of the simple dynamics of the system in which they move to adapt their speeds at the times of their collisions to eventually reach a state in which they synchronize. Since communication is possible only at the times of collisions, the amount of information exchanged depends on the number of the needed collisions to reach synchronization which is not necessarily a small number. Therefore, a model that assumes that robots perform less communication or no communication at all would be of great interest. We show in chapters 2 and 3 that some information might be obtained if no communication is allowed between the robots. Another strong assumption in [55, 51] is the ability of robots to modify their speeds. We also show in those chapters how to exploit the momentum conservation and energy preservation principles to simulate an exchange of information at the time of collision between two robots without having explicit communication and without having any control of their velocities.

An interesting characteristic of particle systems is that the movements of particles are totally out of their control. There is an external factor, like heating, that causes their movements. We find in distributed computing similar assumptions on the movements of robots. As we mentioned before, population protocols are an example of similar behavior.

Another application of particle systems in computer science is described by Cooley and Newton in [11, 12]. They show how to generate pseudo random numbers efficiently by using particle systems.

4. Survivability of Mobile robots

As we said above, mobile robots have been used to perform tasks that, otherwise carried out by humans, would be dangerous, less efficient, and expensive, for instance, environment exploration, perimeter patrolling, mapping, pattern formation and localization.

Large collections of robots with limited capabilities are called *swarms*. Despite their simplicity, they are used to perform complicated tasks like surveillance and monitoring in hazardous or hard to access environments. Due to the nature of the environments on which mobile robots are frequently deployed, they may get destroyed in other words they may *die* while performing their task. For instance, while a robot is exploring a terrain, it can be destroyed by enemy forces or by stepping on a mine. Understanding their survivability will help us understand which measures could be taken in order to ensure that they fulfill their purpose.

Some researchers have studied the destruction of mobile agents while visiting some specific location of their environment. Dobrev et al. in [21] introduced the *black hole search task*. They consider a set of mobile agents moving in a ring searching for a highly harmful item called *black hole*. A black hole is a stationary process that destroys any visiting agent upon its arrival without leaving any trace of it. This task requires that at least one robot survives in order to report the location of the black hole.

A somewhat similar problem for very simple mobile robots was introduced by Moshe Rosenfeld in [46] in what he calls the *salmon problem*. The salmon problem is inspired by the life cycle of salmons: a salmon after being hatched lives in the ocean for a period of several years, then it returns to its place of birth to spawn and die. The salmon problem is stated in [46] as follows: Consider *n salmon fries* distributed on a ring, each fry moving with constant speed either clockwise or counterclockwise. When two fries collide they reverse direction and when a fry returns to its initial position, it dies. Death has priority over collisions. Is it possible that some fries live forever? Is there an efficient algorithm to decide whether all fries will die?

Rosenfeld gives an example of a configuration of five salmons of which one dies and the remaining four live forever. However, Rosenfeld's example has a flaw we show this in Chapter 4 and we give the first correct example of a swarm with survivors. Moreover we study the salmon problem in a more general setting.

CHAPTER 3

Localization

In this chapter we introduce our results on the problem of *localization* also known as *position discovery*: a collection of n anonymous mobile robots is deployed on a unit-perimeter cycle or a unit-length line segment. Every robot starts moving with constant speed at the same time, and updates its speed, according to the laws of classical mechanics for elastic collision, each time it meets any other robot or segment endpoint. The goal of each robot is to detect the presence, the initial position, and starting direction of each other robot.

The robots cannot communicate or perceive information about the environment in any way other than by bouncing. Each robot has a clock allowing it to observe the times of its bounces. The robots have no control on their walks, which are determined by their initial positions, speeds, and the starting directions. Since robots are anonymous, each robot executes the same algorithm, it receives input data in real-time about the times of the bounces, and terminates when the robot is assured about the existence and the positions of all the robots.

In Section 2 we study the localization task by robots of same masses and speeds that are deployed either on a cycle or a segment. In Section 3, we present our results concerning robots of arbitrary masses and speeds that are deployed on the cycle. In the following section we establish the common set of assumptions and notation for both Section 2 and Section 3. The results presented in this chapter were published in [16, 18].

1. Preliminaries

We consider a set of n synchronous and anonymous robots $r_0, r_1, \ldots, r_{n-1}$ deployed on a continuous, one-dimensional environment which is represented either by a unit-perimeter cycle or by a unit-length line segment. The cycle is modeled by a real interval $[0, 1)$ with 0 and 1 corresponding to the same point. The set of n robots $r_0, r_1, \ldots, r_{n-1}$ is deployed in the environment and start moving at time $t = 0$ (where the indexing of the robots is used for purposes of analysis, only). The robots are not aware of the original positions and directions of other robots or the total number of robots in the collection. The robots move at constant speed. Each robot is given an initial direction (clockwise or counterclockwise in the cycle and left-to-right or right-to-left on the segment) at which it starts its movement. Each robot knows the perimeter of the cycle (or the length of the segment) and it has a clock permitting to register the time of each of its collisions and store it in its memory.

By $r_i(t) \in [0, 1]$ we denote the position of robot r_i at time t. We suppose that originally each robot r_i occupies point $r_i(0)$ of the environment and that $0 \leq r_0(0) < r_1(0) < \ldots < r_{n-1}(0) < 1$. By dir_i we denote the starting direction of robot r_i and we set $dir_i = 1$ if r_i starts its movement in the counterclockwise direction around the cycle or the left-to-right direction along the segment. By $dir_i = -1$ we denote the clockwise starting direction (on the cycle) or right-to-left (on the segment).

Throughout this thesis, we assume the principle of momentum conservation as well as the conservation of energy and that in any collision no more than two robots participate. When two robots meet they instantaneously update their velocities according to the laws of classical mechanics for elastic collisions, i.e, if robots r_1 and r_2 of masses m_1 and m_2, and velocities u_1 and u_2 respectively, collide, after their collision they get new velocities v_1 and v_2 , respectively, where:

$$v_1 = \frac{m_1 - m_2}{m_1 + m_2} u_1 + \frac{2m_2}{m_1 + m_2} u_2, \quad v_2 = \frac{2m_1}{m_1 + m_2} u_1 + \frac{m_2 - m_1}{m_1 + m_2} u_2. \tag{1}$$

We call the trajectory of a robot a *bouncing walk*. The robots have no control on their bouncing walks, which depend only on their initial positions and directions, imposed to them by an adversary, and the bounces caused by meeting other robots. Each robot has to report the coordinates of all robots of the collection, i.e., their initial positions and their initial directions. The robots cannot communicate in any other way except for observing their meeting times. Each robot is aware of the type of the environment (cycle or segment). The only information available to each robot is the *bounce sequence*, i.e. the series of time moments $t_1, t_2, \ldots$, corresponding to its bounces resulting from the meetings with other robots.

By *localization algorithm* we mean a procedure executed by each robot, during which the robot performs its bouncing walk and uses its bounce sequence as the data of the procedure, outputting the initial positions and directions of all robots.

2. Bouncing Robots with Same Speeds

In this section we start to study the problem of localization in the cycle and in the segment by our simplest model of bouncing robots. We assume that all robots have the same masses and speeds thus when two of them collide they simply reverse directions and keep moving with the same speed (see equations 1).

Our aim in this section, is to investigate whether or not is possible for each robot to find out, after some time of its movement, what is the number of robots in the collection and their relative positions in the environment. If not, what are the configurations of robots' initial positions and directions for which a position detection algorithm exists (i.e. it is possible to report the initial configuration after a finite time). As well as the smallest amount of time after which a robot is assured to identify all other robots in the collection.

The dynamics of bouncing robots is simple and it is a good way to start their study. Despite their limited capabilities in this section we prove that they are capable to figure out the starting parameters of all the other robots in most cases. In the following section we establish the specific assumptions on this model of bouncing robots as well as some notation that we will use throughout this section.

2.1. Preliminaries. For simplicity we assume that all robots have unit-speed then distance and time traveled by the robots are commensurable, so during time t each robot travels distance t. Consequently, in this section we compare distances traveled to time intervals. When two robots meet, they *bounce back*, i.e., they reverse the directions of their movements. We call the sequence of pairs $(r_0(0), dir_0), \ldots, (r_{n-1}(0), dir_{n-1})$ the *initial configuration* of robots.

By the *cost* $C_{\mathcal{A}}(n)$ of algorithm $\mathcal{A}$ we understand the smallest value, such that for any feasible initial configuration of n robots in the environment, each robot executing $\mathcal{A}$ can report the initial configuration while performing a bouncing walk of total distance $C_{\mathcal{A}}(n)$. As in some cases the cost of the algorithm varies, depending on the robot initial directions, we denote by $C_{\mathcal{A}}(n, k)$ the cost of $\mathcal{A}$ for the class of initial configurations such that $1 \leq k \leq n/2$ robots start in one direction and $n-k$ start in the opposite one. We propose an algorithm to be executed by any robot, which computes the original positions of all other robots of the collection. We say that such an algorithm is optimal if the time interval after which the robot is assured to have the knowledge of the positions of all other robots is the smallest possible.

2.2. Results. We characterize all the feasible configurations for the cycle and the segment. For both cases we give optimal position detection algorithms for all feasible configurations. Our algorithm for the segment requires $O(n)$ robot's memory, while constant size memory is sufficient for robots bouncing on the cycle. We suppose that in one memory word we may store a real value representing the robot's position in segment $[0, 1]$.

For the case of the cycle, we show that all robot configurations with not all robots given the same initial direction are feasible. We give a position detection algorithm working for all feasible configurations. The cost of our algorithm is not constant, but it depends on the number of robots starting their movement in each direction. When $k \leq n/2$ is the number of robots starting their walks in one direction with $n-k$ given the opposite direction we prove that our algorithm has cost $\frac{1}{2}\lceil\frac{n}{k}\rceil$. We prove that this algorithm is optimal.

For the case of the segment we prove that no position detection algorithm exists for *symmetric* initial configurations. Each symmetric configuration is a configuration of a subset of robots on a subsegment, concatenated alternately with its reflected copy and itself. We give a position detection algorithm of cost 2 working for all feasible (non-symmetric) configurations on the segment. This algorithm is proven to be optimal.

In Subsection 2.3 we give the position detection algorithm for the cycle and prove its correctness for all feasible configurations. Subsection 2.4 analyses the cost of the position detection algorithm for the cycle and proves its optimality. The segment environment is addressed in Subsection2.5. The argument for the segment proceeds by reduction to that for the cycle, but the criteria for a feasible configuration on the segment take a different form, dependent on the symmetry of the configuration.

2.3. The Algorithm on the Cycle. As there is no system of coordinates on the cycle common to all robots, each robot must compute the relative positions of other robots with respect to its own starting position. We may then infer that each robot assumes that its starting position is the point 0. We then suppose that $0 = r_0(0) < r_1(0) < \ldots < r_{n-1}(0) < 1$ and it is sufficient to produce the algorithm for robot r_0.

We assume in this chapter that all robot indices are taken modulo n. When two robots meet, they reverse the directions of their movements, so the circular order of the robots around the cycle never changes. When two robots r_i and r_{i+1} meet at time t, we have $r_i(t) = r_{i+1}(t)$, and $r_i(t)$ was moving counterclockwise while $r_{i+1}(t)$ was moving clockwise just before the meeting time t.

We denote by $dist(x, y)$ the distance that x has to traverse in the counterclockwise direction around the cycle to reach the position of y (we call it the *counterclockwise distance* from x to y. Note that the *clockwise distance* from x to y equals $1-dist(x, y)$.

In order to analyze the cycle movement of the robots we consider an infinite line $L = (-\infty, \infty)$ and for each robot r_i, $0 \leq i \leq n-1$ we create an infinite number of its copies $r_i^{(j)}$, all having the same initial direction, such that their initial positions are $r_i^{(j)}(0) = j + r_i(0)$ for all integer values of $j \in \mathbb{Z}$ (see Fig. 3.1). We show that, when all copies of robots move along the infinite line while bouncing at the moments of meeting, all copies $r_i^{(j)}$ of a robot r_i bounce and reverse their movements at the same time. More precisely we prove

LEMMA 1. *For all $t \geq 0, 0 \leq i \leq n-1$ and $j \in \mathbb{Z}$ we have $r_i^{(j+1)}(t) = r_i^{(j)}(t) + 1$.*

Proof. Since the claim of the lemma holds by construction at time $t = 0$ and at any bounce moment all copies of the bouncing robots $r_b^{(j)}$ simultaneously reverse their movement, the claim of the lemma holds by induction on the number of bounces. ■

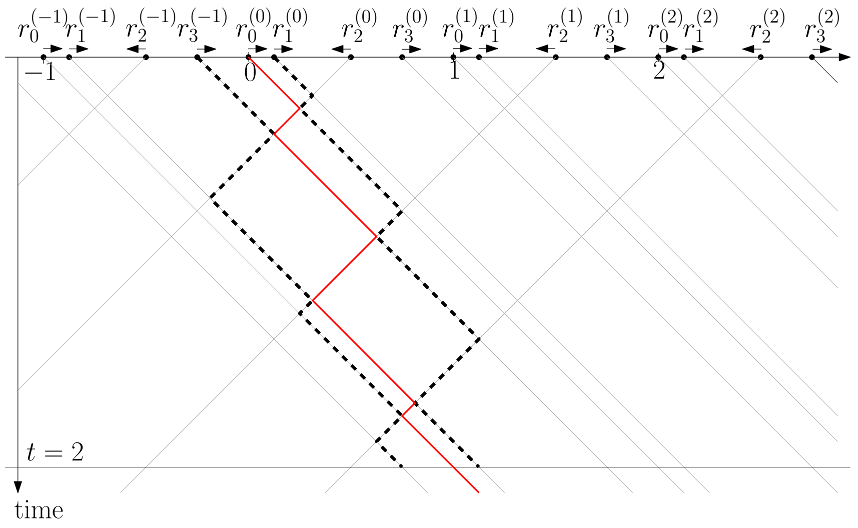

FIGURE 3.1. Example of a bouncing movement of four robots

We use the concept of a *baton*, applied recently in [33]. Suppose that each robot initially has a virtual object (baton), that the robot carries during its movement, but at the moment of meeting, two robots exchange their batons. By $b_i^{(j)}$ we denote the baton originally held by robot $r_i^{(j)}$ and by $b_i^{(j)}(t)$ we denote the position of this baton on the infinite line at time t. We can easily show the following lemma.

LEMMA 2. *For all* $t \geq 0, 0 \leq i \leq n-1$ *and* $j \in \mathbb{Z}$ *we have* $b_i^{(j)}(t) = b_i^{(j)}(0) + dir_i \cdot t = b_i^{(0)}(0) + j + dir_i \cdot t$.

Proof. Since the bouncing robots exchange their batons, the batons travel at constant speed 1 in their original directions. Therefore, at time t each baton traveled the distance t so we have $b_i^{(j)}(t) = b_i^{(j)}(0) + dir_i \cdot t$. On the other hand, by construction we have $b_i^{(j+1)}(0) = b_i^{(j)}(0) + 1$ and both batons $b_i^{(j)}, b_i^{(j+1)}$ travel at unit speed in the same direction. Hence, we have by induction on j, that $b_i^{(0)}(t) = b_i^{(j)}(t) + j$. The claim of the lemma follows. ■

In Fig. 3.1 the trajectories of all the batons held originally by the robots going in direction dir are the lines of slope dir. Each robot r_i bounces while its trajectory intersects a trajectory of some baton, since this baton is then held by one of the robots r_{i-1}, r_{i+1}. For example, the trajectory of robot $r_0^{(0)}$, is represented by a red fat polyline on Fig. 3.1, while the trajectories of its neighbor robots $r_3^{(-1)}$ and $r_1^{(0)}$ bouncing at $r_0^{(0)}$ are given by dashed polyline.

Lemma 3. *Consider robot r_a, which at the time moment t, while traveling in direction dir, meets some other robot. Suppose that, at the time of this meeting, r_a traveled the total distance d in direction dir (hence the total distance of $t - d$ in direction $-dir$). Then there exists a robot r_b, which was originally positioned at distance $(2d \mod 1)$ in direction dir on the cycle. More precisely, $(2d \mod 1) = dist(r_a, r_b)$ if $dir = 1$ and $(2d \mod 1) = dist(r_b, r_a) = 1 - dist(r_a, r_b)$ if $dir = -1$. Moreover r_b started its movement in direction $-dir$.*

Proof. Suppose that at time t robot r_a traveling in direction dir meets some other robot traveling in the opposite direction (e.g. on Fig. 3.1, see the intersection of the trajectory of $r_0^{(0)}$ with the trajectory of the baton $b_2^{(2)}$ originally held by $r_2^{(2)}$). Suppose that the baton obtained by r_a at the moment of the meeting was originally held by some robot r_b. Robot r_a traveled the total distance d in direction dir and the total distance $t - d$ in direction $-dir$, while the baton obtained by r_a at the moment of the bounce traveled distance t in direction $-dir$. Hence during time $t - d$ robot r_a and the baton stayed at the same distance and during time d they were both traveling approaching each other (i.e. jointly covering total distance $2d$ while approaching). Therefore, at time $t = 0$ the distance between robots r_a and r_b was $2d$. Since r_b may be a copy of a robot and all copies of the same robot are at integer distance, the distance of r_a to r_b on the cycle is $2d \mod 1$. The initial direction of r_b equals the direction of its original baton, i.e. $-dir$. ■

Remark 1. *The value $(2d \mod 1)$ may sometimes be equal to zero which corresponds to r_a meeting the robot currently holding the original baton of r_a (e.g. the sixth bounce of $r_0^{(0)}$ on Fig. 3.1). On the other hand, some meetings of robots may correspond to the same computed value of $(2d \mod 1)$ (e.g. all odd-numbered bounces*

```
**Algorithm** RingBounce ($dir$ : $\{-1, 1\}$);

1.    **var** $left \leftarrow 0, right \leftarrow 0$ : **real**;
2.    reset $clock$ to 0;
3.    **while** $true$ **do**
4.        **do** walk in direction $dir$ **until**
5.            $((clock - left \geq 1/2)$ **and** $(clock - right \geq 1/2))$ **or** a meeting occurs;

6.        **if** $(clock - left \geq 1/2)$ **and** $(clock - right \geq 1/2)$ **then** EXIT;
7.        **if** $dir = 1$ **then**
8.            $right \leftarrow clock - left$;
9.            **if** $(0 < right < 1/2)$ **then**
10.               OUTPUT robot at original position $2 \cdot right$ and direction $-dir$;
11.       **else**
12.           $left \leftarrow clock - right$;
13.           **if** $(0 < left < 1/2)$ **then**
14.               OUTPUT robot at original position $1 - 2 \cdot left$ and direction $dir$;
15.       $dir \leftarrow -dir$;
```

of $r_0^{(0)}$ on Fig. 3.1), so some bounces do not have a new informative value about other robot positions.

The algorithm `RingBounce` executed by a robot, which reports initial positions and directions of all other robots on the cycle, uses Lemma 3. Each bounce results in the output of information concerning one robot of the cycle. In this way, a robot running such an algorithm needs only a constant-size memory. An additional test is made in line 10 to avoid outputting the same robot position more than once.

The robot's memory consists of two real variables *right* and *left* in which the robot will store the total distance traveled, respectively, in the counterclockwise and clockwise direction. The robot also accesses its system variable *clock* which automatically increases proportionally to the time spent while traveling (i.e. to the distance traveled).

THEOREM 1. *Suppose that among all robots bouncing on the cycle there is at least one robot having initial clockwise direction and at least one robot with the initial counterclockwise direction. The algorithm* `RingBounce`, *executed by any robot of the* *collection, correctly reports the initial positions and directions of all robots on the cycle with respect to its initial position.*

Proof. Suppose without loss of generality, that the robot executing `RingBounce` is robot r_0. Since there exists at least one other robot starting in the direction different from dir_0, robot r_0 will alternately travel in both directions, indefinitely bouncing against its neighbors r_1 and r_{n-1} on the cycle.

We show by induction, that at the start of each iteration of the while loop from line 3, the variable $left$ (resp. $right$) equals to the total distance traveled by r_0 clockwise (resp. counterclockwise). Suppose, by symmetry, that r_0 walks counterclockwise in the i-th iteration and the inductive hypothesis is true at the start of this iteration. Since, by inductive hypothesis, variable $left$ keeps the correct value through i-th iteration, variable $right$ is correctly modified at line 8, as $clock$ value equals the total distance traveled in both directions. Consequently, the inductive claim is true in the $(i+1)$-th iteration.

We prove now that positions and directions of all robots are correctly reported before the algorithm ends. Take any robot r_i, $1 \leq i \leq n-1$. We consider first the case when the initial direction of r_i was clockwise. The trajectory of its original baton $b_i^{(0)}$ is then a line of slope 1 (cf. Fig. 3.1). Observe that robot r_0 stays at the same distance from baton b_i when walking in the clockwise direction and approaches it (reducing their counterclockwise distance $dist(r_0, b_i)$) when walking counterclockwise. Since $dist(r_0, b_i) \leq 1$, and r_0 and b_i walk towards each other, they approach at speed 2 during the counterclockwise movement of r_0. Consequently, the trajectories of r_0 and b_i intersect and r_0 eventually meets robot r_1 carrying baton b_i. Indeed, in line 4 of algorithm `RingBounce`, robot r_0 continues its movement as long as its total distance traveled in the counterclockwise direction is less than $1/2$, which leads to the meeting of r_0 and r_1 (carrying baton b_i), before both robots finish their executions of the algorithm. Consequently, at the moment of their meeting, r_0 outputs at line 10 the initial distance between $r_0^{(0)}$ and $r_i^{(0)}$ on line L, which equals twice the time spent while the robots were approaching each other. As r_0 may obtain a copy of the same baton more than once (cf. r_0 intersecting several trajectories of batons $b_2^{(j)}$ on Fig. 3.1), the condition $(0 < right < 1/2)$ at line 9 permits to report the position of each other robot once only. Indeed, only $r_i^{(0)}$ - the copy of r_i at the closest counterclockwise distance to r_0 verifies this condition.

Consider now the case when robot r_i, $1 \leq i \leq n-1$, starts its walk on the cycle in the counterclockwise direction. Then r_0 obtains baton b_i while walking clockwise, i.e. at the moment of some bounce at r_{n-1}, while r_{n-1} holds baton b_i. In this case, robot r_0 stays at the same distance from baton b_i when walking in counterclockwise direction and approaches it (reducing their distance of $dist(b_i, r_0) = 1 - dist(r_0, b_i)$) when walking clockwise. At the moment when r_0 meets r_{n-1} holding baton b_i (whose trajectory originates from segment $[-1, 0]$ of L) the value of variable $left$ equals half the clockwise distance from $r_0(0)$ to $r_i(0)$. Indeed, at the moment of the meeting, half of this distance was covered by r_0 walking clockwise (the value of $left$) and the other half was covered by the counterclockwise move of baton b_i. Consequently the clockwise distance from the initial position of r_0 to the initial position of r_i equals $1 - 2 \cdot left$, correctly output at line 14. ■

Observe that, once the original positions and directions of all robots are reported, it is easy to monitor all further movements of all robots of the collection, i.e. their relative positions at any moment of time. However, this would require a linear memory of the robot performing such task.

2.4. The Execution Time of Bouncing on the Cycle. As stated in the introduction, we look for the algorithm of the optimal cost, i.e. the smallest possible total distance traveled, needed to correctly report any initial configuration. We show that the algorithm `RingBounce` is the optimal one, i.e. that the time moment, at which the robot can be sure that the positions of all other robots have been reported, is the time when the robot stops executing `RingBounce`. Observe that algorithm `RingBounce` has cost at least 1, i.e. a robot executing it must travel at least distance

1. Indeed, the loop from lines 4-5 continues unless robot's walk distance in each direction totals at least half the size of the cycle. On the other hand, the example from Fig. 3.1 shows, that if the number of robots starting their walks in one direction is different from the number of robots starting walking in the opposite direction, the total cost of `RingBounce` may be higher. We have

THEOREM 2. *Consider a collection of n robots on the ring, such that k of them, $1 \leq k \leq n/2$, have one initial direction and the remaining $n-k$ robots have the other initial direction. Then the cost of* `RingBounce` *is* $C_{\mathcal{RB}}(n,k) \leq \frac{1}{2}\lceil \frac{n}{k} \rceil$.

Proof. If there are more robots starting in one direction, say positive direction dir, than in direction $-dir$ then r_i gets more frequently dir-moving batons (cf. Fig. 3.1). Since the route of r_i, intersects the trajectory of each baton only once, r_i must meet copies of batons originating from other segments than $[0,1]$ of line L. By counting we show that the last such segment is $[\lceil (n-k)/k \rceil - 1, \lceil (n-k)/k \rceil]$. Hence, in the worst case, r_i walks distance $1/2$ in direction dir and distance $\lceil (n-k)/2k \rceil$ in direction $-dir$.

By Lemma 1, we can translate the collection of robots and start their enumeration so that any of them is at the point 0 of line L and, without loss of generality, it is sufficient to consider the total walk length of r_0. By symmetry we assume that r_0 starts walking counterclockwise on the cycle.

Consider first the case when $n-k$ robots from the claim of the theorem start walking counterclockwise and k robots start walking clockwise on the cycle, with $k \leq n-k$. Note that r_0 alternately changes its direction of walk and, according to lines 4-5 of algorithm `RingBounce`, it has to travel a distance of at least $1/2$ in each direction. At the conclusion of each segment of the clockwise walk around the cycle (i.e. left walk along line L), r_0 bounces against r_{n-1}, collecting one of the $n-k$ batons traveling counterclockwise. Denote by t_{2i}, for $i = 1, \ldots, n-k$ the sequence of the consecutive time moments of all bounces of r_0 against robot r_{n-1} (recall that time equals the total distance traveled up to that moment). Suppose that r_0 starts executing algorithm `RingBounce` at time $t_0 = 0$ and denote by t_{2i+1}, for $i = 0, \dots, n-k-1$ the sequence of the consecutive time moments of all bounces of r_0 against robot r_1. At time $t_{2(n-k)}$, r_0 gets originally held baton b_0 and the total length of its clockwise travel becomes exactly $1/2$ (i.e. the value of variable $left$ becomes $1/2$). Since $t_1 < t_2 < \dots < t_{2(n-k)}$, before time $t_{2(n-k)}$ r_0 bounced also $n-k$ times against r_1, each time getting a baton, which is traveling clockwise. If $k = n/2$, there are $k = n-k$ lines of slope 1 originating from segment $[0,1)$ of L, which are trajectories of k batons traveling clockwise, see Fig. 3.2. Therefore, the loop from lines 4-5 of algorithm `RingBounce` continues until variable $right$ equals $1/2$ and algorithm finishes through the exit condition at line 6. In this case the total walk time equals to $left + right = 1$ and $\frac{1}{2}\lceil\frac{n}{k}\rceil = 1$ so the claim of the theorem holds.

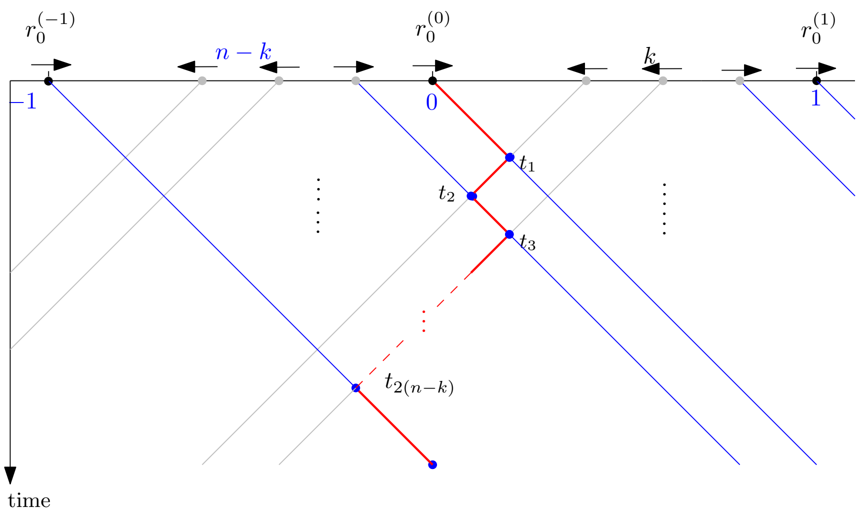

FIGURE 3.2. $n-k$ (blue) batons travel in counterclockwise direction while k (gray) batons travel in clockwise direction, the trajectory of r_0 appears in red. By time $t_{2(n-k)}$, robot r_0 has discovered all $n-k$ batons traveling in counterclockwise direction as well as those traveling in clockwise direction

In the case $k < n/2$ there are only k batons traveling clockwise ($k < n-k$), so some of them are received more than once by r_0 during the bounces at times $t_1, t_3, \dots, t_{2(n-k)-1}$. Therefore, only k copies of batons traveling clockwise originate from each integer segment $[i, i+1)$ on line L. Consequently, r_0 obtains batons whose trajectories originate from segments other than $[0,1)$ and its total traveling distance in the counterclockwise direction exceeds $1/2$. More precisely, the $(n-k)$-th consecutive copy of a baton traveling clockwise, obtained at time $t_{2(n-k)-1}$ must originate from the segment $[i^*-1, i^*]$, where $i^* = \lceil \frac{n-k}{k} \rceil$. The distance from $r_0(0)$ the initial position of r_0 and the starting position of baton met at time $t_{2(n-k)-1}$ does not exceed the value of $\lceil \frac{n-k}{k} \rceil$. Since robot r_0 has to travel counterclockwise at most half of this distance (the other half being covered by the moving baton), the total time spend by r_0 in both directions does not exceed

$$1/2 + \frac{1}{2}\left\lceil \frac{n-k}{k} \right\rceil = \frac{1}{2}\left\lceil \frac{n}{k} \right\rceil$$

Consider now the second case in which $n-k$ robots from the claim of the theorem start walking clockwise and k robots start walking counterclockwise on the cycle, with $k < n-k$. As in the previous case we can denote by $t_1 < t_2 < \ldots < t_{2(n-k)-1}$ the consecutive bounce times, where t_i for odd values of i denote the times of the bounces of r_0 against r_1 and those for even values of i denote the times of the bounces against r_{n-1}. By symmetry to the first case, the bounce at time $t_{2(n-k)-1}$, when r_0 moves counterclockwise, arises when the variable *right* does not exceed the value of $1/2$ (i.e. the total distance traveled counterclockwise by r_0) and the value of $left$ is already greater than $1/2$. At this time moment, robot r_0 goes clockwise and after the last bounce at time $t_{2(n-k)}$ continues counterclockwise and exits algorithm `RingBounce` where variable *right* becomes equal to $1/2$. Indeed, since only $n-k$ batons travel clockwise, the next bounce of robot r_0 would imply getting a baton whose trajectory originates at segment $[1, 2]$ of line L, but this would make variable *right* exceed first the value of $1/2$ and cause the exit in line 6 of `RingBounce`.

Similarly to the previous case, as $k < n-k$, at some of the bounces at times $t_2, t_4, \ldots, t_{2(n-k)}$, robot r_0 obtains the same batons. More precisely, during the bounce at time $t_{2(n-k)}$ r_0 obtains the baton whose trajectory originates at segment $[-\lceil \frac{n-k}{k} \rceil, -\lceil \frac{n-k}{k} \rceil + 1]$ of line L. Hence the total clockwise distance traveled by r_0

does not exceed $\frac{1}{2}\lceil\frac{n-k}{k}\rceil$ and the distance traveled in both directions does not exceed $1/2 + \frac{1}{2}\lceil\frac{n-k}{k}\rceil = \frac{1}{2}\lceil\frac{n}{k}\rceil$ proving the claim of the theorem. ■

From Theorem 2 we immediately have the following Corollary, which bounds the worst-case walking time for a robot.

COROLLARY 1. *Assuming that the collection of n robots admits robots starting their movements in both directions around the cycle, Then the cost of* `RingBounce` *is* $C_{\mathcal{RB}}(n) \leq \frac{n-1}{2}$.

The algorithm `RingBounce` continues until the total lengths of walks in both directions reach the values of at least 1/2, since this guarantees that the presence of each robot is eventually detected. The following theorem proves that the cost of `RingBounce` algorithm is optimal even if the (a priori) knowledge of the number of robots is assumed.

THEOREM 3. *Suppose that there is a collection of n robots on the cycle, such that k of them, $1 \leq k < n/2$, have one initial direction and the remaining $n-k$ robots have the other initial direction. Then for every $\epsilon > 0$ there exists a distribution of such robots on the cycle with their initial positions $0 \leq r_0 < r_1 < \ldots < r_{n-1} < 1$, so that a position detection algorithm terminating at time $\frac{1}{2}\lceil\frac{n}{k}\rceil - \epsilon$ cannot determine the initial positions of all robots on the cycle, even if the values of n and k are known in advance.*

Proof. Consider the following collection of n robots $r_0, r_1, \ldots, r_{k-1}, r_k, \ldots, r_{n-1}$ on the cycle, where each r_i has a counterclockwise starting direction for $0 \leq i < k$ and the clockwise starting direction for $k \leq i \leq n-1$, with the initial positions of the robots

$$r_0 = 0, r_1 = \frac{\epsilon}{2^{k-1}}, r_2 = \frac{\epsilon}{2^{k-2}}, \ldots, r_{k-1} = \frac{\epsilon}{2},$$
$$r_k = 1 - \frac{\epsilon}{2}, \ldots, r_{n-1} = 1 - \frac{\epsilon}{2^{n-k}}$$

Suppose, by contradiction, that a position detection algorithm executed by robot r_0 can determine the positions of all other robots with the total walking distance of r_0 at most $\frac{1}{2}\lceil\frac{n}{k}\rceil - \epsilon$. Robot r_0, in order to determine positions of all other robots, has to obtain each baton $b_1, \ldots, b_{n-1}$. Robot r_0 gets the clockwise-traveling batons $b_k, b_{k+1}, \ldots, b_{n-1}$ in this order at the moments of its bounces against r_1. On the other hand, the remaining batons are obtained by r_0 in order $b_{k-1}, b_{k-2}, \ldots, b_1$ at the moments of its bounces against r_{n-1}. However, since $k < n$, at least some of the batons $b_{k-1}, b_{k-2}, \ldots, b_1$ are obtained repeatedly (in the same cyclic order) because the left and right bounces are alternated. More precisely, baton b_{k-1} is obtained $\lceil(n-k)/k\rceil$ times by b_0, hence this sequence of batons is

$$\overleftarrow{b_k}, \overrightarrow{b_{k-1}}, \overleftarrow{b_{k+1}}, \overrightarrow{b_{k-2}}, \ldots, \overleftarrow{b_{2k-1}}, \overrightarrow{b_0}, \overleftarrow{b_{2k}}, \overrightarrow{b_{k-1}},$$

$$\overleftarrow{b_{2k+1}}, \overrightarrow{b_{k-2}}, \ldots, \overleftarrow{b_{3k-1}}, \overrightarrow{b_0}, \ldots, \overleftarrow{b_{n-1}}, \overrightarrow{b_f}$$

where $f = k(\lceil n/k\rceil + 1) - (n+1)$ and $\overleftarrow{b_i}$ denotes baton b_i traveling clockwise and $\overrightarrow{b_j}$ denotes baton b_j traveling counterclockwise. The copies of the last two batons of this sequence are the most distant from r_0 on line L. The trajectory of baton $\overleftarrow{b_{n-1}}$ origins in segment $[0,1)$ of line L and $dist(r_0^0(0), b_{n-1}^0(0)) = 1 - \frac{\epsilon}{2^{n-k}}$. On the other hand, the trajectory of baton $\overrightarrow{b_{k(\lceil n/k\rceil+1)-(n+1)}}$ obtained by b_0 starts in the segment $[-\lceil(n-k)/k\rceil, -\lceil(n-k)/k\rceil + 1]$ and its original distance to b_0 is

$$dist(b_f^{(-\lceil(n-k)/k\rceil)}(0), b_0^{(0)}(0)) > \lceil(n-k)/k\rceil - \frac{\epsilon}{2}$$

As in order to meet each of these batons, r_0 has to travel half of its original distance to each of them (the other half is covered by the corresponding baton itself) the total travel time by r_0 is bound by

$$\frac{1}{2}(1 - \frac{\epsilon}{2^{n-k}}) + \frac{1}{2}(\lceil(n-k)/k\rceil - \frac{\epsilon}{2}) >$$

$$\frac{1}{2}(1 + \lceil(n-k)/k\rceil - \epsilon) > \frac{1}{2}(\lceil n/k\rceil) - \epsilon$$

which contradicts the assumed claim and proves the theorem. ∎

Clearly each configuration of robots with the same initial direction of all robots is infeasible, because no robot ever bounces. Consequently from Theorem 2 and Theorem 3 follows

Corollary 2. *The family of infeasible initial configurations of robots on the cycle contains all configurations with the same initial direction of all robots.* `RingBounce` *is the optimal position detection algorithm for all feasible initial configurations of robots on the cycle. This algorithm assumes constant-size memory of the robot running it.*

Clearly, we can easily adapt algorithm `RingBounce`, so for infeasible initial configuration the algorithm stops and reports the infeasibility. It is sufficient to test whether the very first walk of the robot ends with a bounce before the robot traverses the distance of 1/2.

2.5. Bouncing on the Line Segment. In this section we show how the algorithm for bouncing robots may be used for the case of a segment. We suppose that each robot walks along the unit segment changing its direction when bouncing from another robot or from an endpoint of the segment. Robots have the same capabilities as in the case of the cycle and they cannot distinguish between bouncing from another robot and bouncing from a segment endpoint.

We consider the segment $[0,1)$ containing n robots, initially deployed at positions $0 \leq r_0(0) < r_1(0), \ldots, r_{n-1}(0) < 1$. Each robot r_i, $0 \leq i \leq n-1$ is given an initial direction dir_i, such that $dir_i = 1$ denotes the left to right initial movement and $dir_i = -1$ denotes initial movement from right to left on segment $[0,1)$. The robots start moving with unit speed at the same time moment $t = 0$ at the predefined directions and they change direction upon meeting another robot or bumping at the segment endpoint. The main difficulty of the segment case is that the robot r executing the position detection algorithm for the cycle has to report the *relative* locations of other robots, i.e. their distances to its own initial position $r(0)$, while in the segment case the *absolute* distance from $r(0)$ to the segment endpoint has to be found.

We show in this section that the bouncing problem is feasible for all initial robot configurations except a small set of symmetric ones. Intuitively, an initial configuration of robots is *symmetric* if the unit segment may be partitioned into k subsegments $S_0, S_1, \ldots, S_{k-1}$, such that the positions and directions of robots in each subsegment form a reflected copy of positions and directions of robots in a neighboring subsegment (see Fig. 3.3). More formally we have the following

Definition 1. *A configuration $C = ((r_0(0), dir_0), \ldots, (r_{n-1}(0), dir_{n-1}))$ is symmetric if there exists a positive integer $k < n$, such that $n \mod k = 0$ and the partition of segment $S = [0,1)$ into subsegments $S_0 = [0, \frac{1}{k})$, $S_1 = [\frac{1}{k}, \frac{2}{k}), \ldots, S_1 = [\frac{k-1}{k}, 1)$ with the following property. For each robot r_i, $0 \le i < n$, if $r_i(0) = \frac{p}{n} + x$, for $0 \le x < \frac{1}{k}$, (i.e. $r_i(0) \in S_p$), $0 \le p < n$, then, if $p > 0$, there exists a robot $r_{i'}$, such that $r_{i'}(0) = \frac{p}{n} - x$ and $dir_{i'} = 1 - dir_i$ and, if $p < n-1$, there exists a robot $r_{i''}$, such that $r_{i''}(0) = \frac{p+2}{n} - x$ and $dir_{i''} = 1 - dir_i$.*

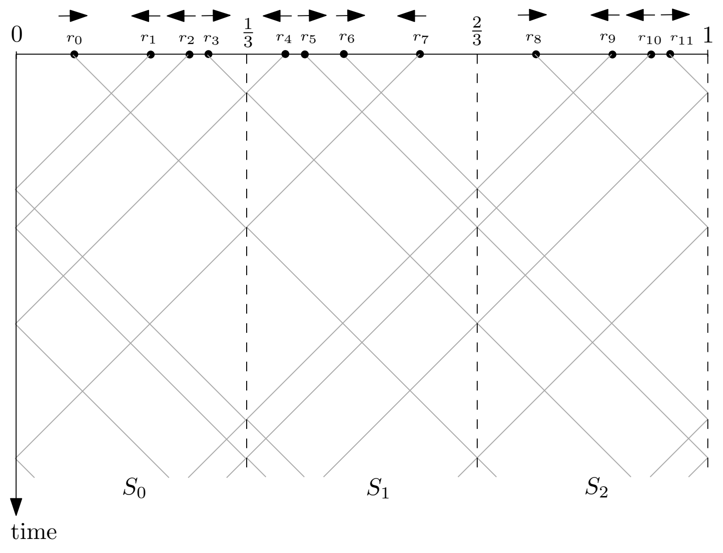

Figure 3.3. Example of a symmetric initial configuration of $n = 12$ robots containing $k = 3$ subsegments

Theorem 4. *Every symmetric initial configuration of robots is infeasible.*

Proof. Let $C_1 = ((r_0(0), dir_0), \ldots, (r_{n-1}(0), dir_{n-1}))$ be a symmetric configuration and k the number of consecutive subsegments, each one being the reflected copy of its neighbor. Construct now configuration $C_2 = ((r'_0(0), dir'_0), \ldots, (r'_{n-1}(0), dir'_{n-1}))$ of n robots considering also a sequence of n equal size intervals and swapping the roles of odd-numbered and even-numbered robots of C_1. More precisely for each robot r_i, such that $r_i(0) \in S_p = [\frac{p}{n}, \frac{p+1}{n})$ there exists a robot r'_j such that $r'_j(0) = \frac{2p+1}{n} - r_i(0)$ and $dir'_j = 1 - dir_i$. Observe that, no robot ever crosses the boundary of any subsegment S_p, i.e. $r_i(0) \in S_p$ implies $r_i(t) \in S_p$, for any $t \geq 0$. Indeed, by construction, for any robot reaching and endpoint of S_p, different from points 0 and 1, at the same time moment there is another robot approaching this endpoint from the other side within the reflected copy of S_p provoking a bounce (cf. Fig. 3.4). Therefore, within each even-numbered subsegment S_{2n} of a symmetric configuration the relative positions of robots and their directions are the same (similarly within each odd-numbered subsegment). Consequently, no robot can distinguish whether it is, say, in an even-numbered segment of C_1 or in an odd-numbered segment of C_2 so its position in segment $[0, 1)$ is unknown. ■

We show now how the position detection algorithm for the cycle may be used in the case of the segment.

Let S be a unit segment containing n robots at initial positions $r_0(0) < r_1(0) < \ldots < r_{n-1}(0)$ and the initial directions $dir_0, \ldots, dir_{n-1}$. Suppose that a segment $S^R \subset [1, 2]$ is the reflected copy of S containing n robots $r^R_n, \ldots, r^R_{2n-1}$ at the initial positions $r^R_n(0) = 2 - r_n(0) < r^R_{n-1}(0) = 2 - r_{n-1}(0) < \ldots < r^R_0(0) = 2 - r_0(0)$. The initial directions of each robot r^R_i is $1 - dir_i$ for $0 \leq i < n$. Let R_2 be the cycle of perimeter 2 composed of segment S concatenated with segment S^R, with points 0 and 2 identified.

Consider the walk of robots r_i, for $0 \leq i < n$, within segment S and cycle R_2. Let $t_0 = 0$ and $0 \leq t_1 < t_2 \ldots$ be the sequence of time moments during which some bounces occur. Each such bounce takes place either between some pair of robots or

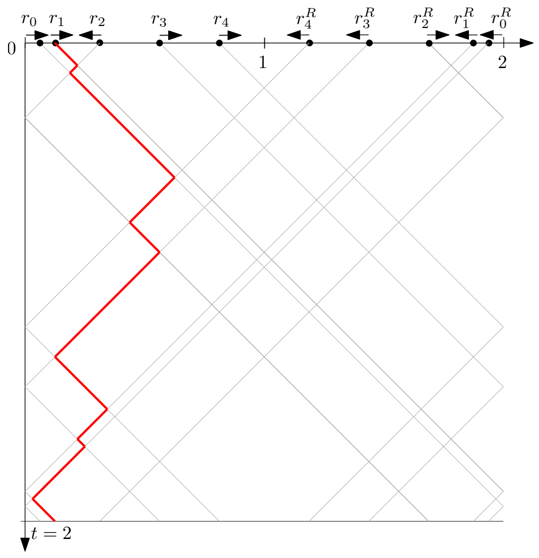

FIGURE 3.4. Five robots on a segment $[0, 1)$ and their reflected copy

when some robot bounces from an endpoint of S. It is easy to see by induction on i that at any time moment $t \in [t_i, t_{i+1}]$ each robot r_j has the same positions in S and R_2 as well as the same direction of movement and that the S^R part of R_2 is a reflected copy of S. Indeed, by construction, this condition is true for the interval $[t_0, t_1]$. If robots r_j, r_{j+1} bounce against each other in S at time t_i, at the same time robots r_j, r_{j+1} bounce in R_2, as well as, by symmetry r_j^R bounces against r_{j+1}^R. If in time t_i robot r_0 (or r_{n-1}) bounce from an endpoint of S, by inductive hypothesis r_0 bounces against r_0^R at point $0 \in R_2$ (or r_{n-1} bounces against r_{n-1}^R at point $1 \in R_2$). In each case, the inductive condition holds. We just showed

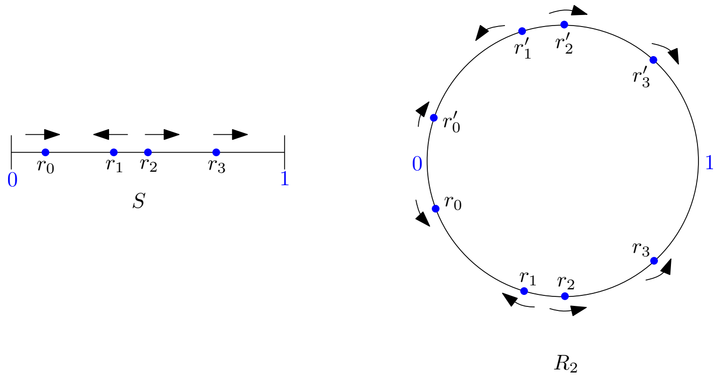

FIGURE 3.5. Example of the corresponding ring, R_2, of a segment, S, of lenght 1 wherein `RingBounce` may be used to find the initial positions of the robots on S

LEMMA 4. *The bounce sequence of any robot r_i on segment S is the same as the bounce sequence of r_i on ring R_2.*

To prove that only symmetric configurations of robots on the segment are infeasible we need the following lemma.

LEMMA 5. *Suppose that the initial configuration of robots $C = ((r_0(0), dir_0), \dots,$ $(r_{n-1}(0),\, dir_{n-1}))$ on a unit segment is not symmetric. Then no internal robot r_i, for $1 \leq i \leq n-2$, may have all its left bounces or all right bounces at the same point of the unit segment.*

Proof. Suppose, by contradiction that there exists an internal robot having all its left bounces at the same point (the proof in the case of all right bounces falling at the same point is similar, by symmetry). Let i be the smallest index $1 \leq i \leq n-2$ of a robot with this property and point x, $0 < x < 1$, be the point of all left bounces of r_i. We show first that the initial configuration of robots belonging to segment $[0, x]$ is the reflected copy of the initial configuration of robots belonging to segment $[x, 2x]$ Then robot r_{i-1} has all its right bounces also at point x. Consequently, at each moment of time after the first such bounce, the position and the direction of robot r_{i-1} is a symmetric (reflected) copy of robot r_i with respect to point x. Then, if $i \geq 2$, the trajectory of r_{i-2} is a reflected copy of the trajectory of r_{i+1}. By induction on i, for any $q \geq 0$ the trajectory of r_q is the reflected copy of the trajectory of r_{2i-q-1} and finally the trajectory of r_0 is the reflected copy of the trajectory of r_{2i-1}. Therefore, all right bounces of robot r_{2i-1} are at point $2x$ of the unit segment, so initial configuration of robots belonging to segment $[0, x]$ is the reflected copy of the initial configuration of robots belonging to segment $[x, 2x]$, as needed.

By induction on j we prove that each subsegment $[(j-1)x, jx]$ is a reflected copy of subsegment $[jx, (j+1)x]$. By minimality of x, no such subsegment contains a point which is never crossed by any robot, hence, for some value of j, we have $jx = 1$, concluding the proof. ■

We can show now, that the set of configurations on the unit segment for which no position detection algorithm exists is exactly the set of symmetric configurations. For all other configurations we propose an optimal position detection algorithm. We suppose that the robot assumes that its initial direction on the segment is positive. Otherwise, the robot needs to be *chirality aware*, i.e. capable of identifying the positive direction of the segment.

THEOREM 5. *For any collection of n robots not in a symmetric initial configuration on the unit segment there exist a position detection algorithm $\mathcal{A}$ with cost $C_{\mathcal{A}}(n) = 2$. For any $\epsilon > 0$ there exist collections of robots, such that some of them cannot terminate the execution of any position detection algorithm before time $2 - \epsilon$.*

Proof.

To construct algorithm $\mathcal{A}$ adapt algorithm `RingBounce` in the following way. The constant of 1/2 used in lines 5, 6, 9 and 13 is changed to 1. Moreover, the values of original positions output in lines 10 and 14 are multiplied by a factor of 2 and put in the list C instead of being directly output. By Lemma 4 the algorithm finds the positions of $2n$ robots of ring R_2 constructed from segment S. Note that, as cycle R_2 has size 2, we needed to scale up the time and distance constants of the algorithm by the factor of 2.

Since the robots of S are not in a symmetric initial configuration, by Lemma 5, only the endpoints of S are the points which are never crossed by any robot in S. Consequently there are only two points in R_2, which are never crossed by any robot. This unique pair of (antipodal) points split cycle R_2 into two segments, segment S, and it reflected copy S^R. Since the positions of all robots are stored in list C, it is possible to perform in algorithm $\mathcal{A}$ the generation of the bounce sequence of each robot of C, in order to find which two robots bounce in one direction against the same positions of the cycle. This way, the first and the last robot on the unit segment as well as its left and right endpoints may be identified, which permits to determine the rank and the absolute initial position of the robot running the algorithm, as well as those of all other robots.

In order to analyze the cost of such algorithm, observe that, since exactly half of $2n$ robots in R_2 have the same initial direction, by Theorem 2, robot r_i terminates its walk at time $\frac{1}{2}\lceil\frac{2n}{n}\rceil \cdot c = 2$, where $c = 2$ is the scaling factor.

We prove now the second part of the claim of the theorem. For any $\epsilon > 0$ we construct two different configurations of robots C_1, C_2 on the unit segment, such that for some robots from C_1 and C_2 the bounce sequence until time $2 - \epsilon$ is the same. Consequently, the robot observing such bounce sequence cannot unambiguously report positions of other robots.

Let C_1 be the configuration of two robots r_0, r_1, such that $dir_0 = dir_1 = 1$ and $r_0(0) = \frac{\epsilon}{5}, r_1(0) = \frac{2\epsilon}{5}$. We find below the first two values t_1, t_2 of the bounce sequence of robot r_0. Robot r_1 reaches point 1 of the segment and bounces at time $t^* = 1 - \frac{2\epsilon}{5}$, while robot r_0 is at point $1 - \frac{\epsilon}{5}$ of the segment. Since at time t^* the robots start to approach, they meet after additional time $\frac{\epsilon}{10}$, so $t_1 = 1 - \frac{2\epsilon}{5} + \frac{\epsilon}{10} = 1 - \frac{3\epsilon}{10}$ and $r_0(t_1) = 1 - \frac{\epsilon}{5} + \frac{\epsilon}{10} = 1 - \frac{\epsilon}{10}$. At time t_1 robot r_0 starts moving left on the segment until it bounces at its endpoint 0. This takes time $1 - \frac{\epsilon}{10}$, so $t_2 = t_1 + 1 - \frac{\epsilon}{10} = 2 - \frac{2\epsilon}{5} > 2 - \epsilon$.

Consider now configuration C_2, containing two robots r_0, r_1, such that $dir_0 = dir_1 = 1$ and $r_0(0) = \frac{\epsilon}{10}, r_1(0) = \frac{\epsilon}{2}$. The similar analysis reveals that $t^* = 1 - \frac{\epsilon}{2}$, $r_1(t^*) = 1$, and $r_0(t^*) = 1 - \frac{2\epsilon}{5}$. At time t^*, r_0 and r_1 start approaching and meet at time $t_1 = t^* + \frac{\epsilon}{5} = 1 - \frac{3\epsilon}{10}$, while $r_0(t_1) = 1 - \frac{\epsilon}{5}$. After the bounce at time t_1, r_0 walks left until it bounces at endpoint 0 at time $t_2 = t_1 + 1 - \frac{\epsilon}{5} = 2 - \frac{\epsilon}{2} > 2 - \epsilon$.

Since for both configurations C_1, C_2 we have $t_1 = 1 - \frac{3\epsilon}{10}$ and $t_2 > 2 - \epsilon$, hence robot r_0 cannot unambiguously output the initial robots' positions before time $2 - \epsilon$.

■

As the algorithm for the segment, presented in the proof of Theorem 5 assumes storing in robot's memory the positions of all robots, from Theorems 4 and 5 follows

Corollary 3. *The family of infeasible initial configurations of robots on the segment contains all symmetric initial configurations of robots. There exists an optimal position detection algorithm for all feasible initial configurations of robots on the segment. This algorithm assumes $O(n)$-size memory of the robot executing it.*

3. Bouncing Robots with Arbitrary Speeds

In this section we keep studying the localization task by bouncing robots. In this section we assume that robots have arbitrary masses and speeds. The difference with the model from the previous section is essentially that in this chapter we do not assume that robots move at the same speed and have knowledge of the speeds of all robots.

In this section, we show that the feasibility of any configuration and the required time for solving it under such stronger constraints depend only on the collection of velocities of the robots. More specifically, if $v_0, v_1, \ldots, v_{n-1}$ is the collection of velocities of a given robot configuration $\mathcal{C}$, we prove that $\mathcal{C}$ is a feasible robot configuration if and only if $v_i \neq \bar{v}$ for all $0 \leq i \leq n-1$, where $\bar{v} = \frac{v_0+\ldots+v_{n-1}}{n}$. To figure out the initial position of all robots no more than $\frac{2}{min_{0\leq i\leq n-1}|v_i-\bar{v}|}$ time is required.

The results of this section were published in [18].

3.1. Preliminaries.

As in the previous section we assume the principle of momentum conservation as well as the conservation of energy, so robots exchange velocities when they collide. The capabilities of each robot are limited to measuring the times of its collisions, to be aware of its velocity at any time, and to process the information that it collects. Robots do not have control of their walks nor of their velocities. Their walks depend on their initial positions, velocities, and sequence of collisions. In addition, robots are not equipped with any visibility mechanism.

In this section we only study the localization task by a collection of n robots $r_0, r_1, \ldots, r_{n-1}$ deployed on a continuous, one-dimensional cycle of perimeter one.

Each robot r_i starts moving at time $t = 0$ with velocity v_i either in counterclockwise or in clockwise direction. As usual, we assume velocities indicate direction and speed. For any non-zero velocity, v, we denote by $|v|$ its speed. As before, by dir_i we denote the starting direction of robot r_i, where $dir_i = 1$ if r_i starts its movement in counterclockwise direction (denoted for simplicity by $(+)$) and $dir_i = -1$ if it starts moving in clockwise direction (denoted by $(-)$). Accordingly, we identify positive velocity with the counterclockwise direction $(+)$ and correspondingly negative velocity with the clockwise direction $(-)$. By $|v|$ we denote the speed of velocity v_i.

Let $\mathcal{P}$ denote the sequence $p_0, p_1, \ldots, p_{n-1}$ of initial positions of the robots, meaning $p_i = r_i(0), 0 \leq i \leq n-1$. Without loss of generality, we assume $p_0 = 0$ as well as a non-decreasing order in the counterclockwise direction of the initial positions of the robots.

If robot r_i moves freely along the cycle with velocity v_i during the time interval $[t_1, t_2]$, then its position at time $t \in [t_1, t_2]$ is given by $r_i(t) = r_i(t_1) + v_i \cdot (t - t_1)$. When two robots collide they exchange velocities following the principle of momentum conservation and conservation of energy in classical mechanics for objects of equal mass [**35**]. We assume that in any collision no more than two robots are involved.

Regarding the capabilities of the robots, we assume that each of them has a clock which can measure time in a continuous way. Each robot is always aware of its clock, current velocity and the time of any of its collisions. The movement of a robot is beyond its control in that it depends solely on its initial position and velocity, as well as the collisions with other robots along the way. At the time of deployment, no robot is aware of the initial position and the velocity of any other robot nor of the total number of robots deployed in the cycle. Moreover, robots do not have a common *sense of direction.*

Let $\mathcal{S} = (\mathcal{P}, V)$ be a system of n mobile robots $r_0, r_1, \ldots, r_{n-1}$ with initial positions $\mathcal{P} = (p_0, p_1, \ldots, p_{n-1})$ and velocities $V = (v_0, v_1, \ldots, v_{n-1})$ respectively; we denote by $\bar{v}$ the average of the velocities in V. We say that the *localization problem* for $\mathcal{S}$ is feasible if there exists a finite time T, such that each robot can determine the initial positions, and the initial velocities of all robots in the system with respect to its own starting position and its own orientation of the cycle. This should be accomplished by each robot by observing the times of a sequence of collisions taking place within some time interval $[0, T]$. Note that each collision is accompanied by the measurement of collision time and a corresponding exchange of velocities.

3.2. Feasibility of the Localization Problem.

For two points p, q in the cycle, by $d^{(+)}(p, q)$ we denote the counterclockwise distance from p to q in the cycle, i.e. the distance which needs to be traveled in the counterclockwise direction in order to arrive at q starting from p. Note that for $p \neq q$ we have $0 < d^{(+)}(p, q) < 1$, and $d^{(+)}(p, q) = 1 - d^{(+)}(q, p)$.

Recall that in order to visualize the dynamics of the robots in the cycle, as we did in the previous section, we may consider an infinite line $L = (-\infty, \infty)$ and for each robot r_i we create an infinite number of its copies $r_i^{(j)}$, all having the same initial velocity, such that their initial positions in L are $r_i^{(j)}(0) = j + r_i(0)$ for all integer values of $j \in \mathbb{Z}$.

We use the idea of *baton*, applied previously in [**16**, **33**], in order to simplify our arguments and to gain intuition of the dynamics of the robots. Assume that each robot holds a *virtual object*, called baton, and when two robots collide they exchange their batons. By $b_i^{(j)}$ we denote the baton originally held by robot $r_i^{(j)}$ and by $b_i^{(j)}(t)$ we denote the position of this baton on L at time t. Notice that the velocity of baton $b_i^{(j)}$ is constant so its trajectory corresponds to the line of slope $1/v_i$.

By putting together the infinite line and the trajectories of batons, we can depict the goal of the robots up to any given time. For instance, in Fig. 3.6 the dynamics of a system of three mobile robots is depicted. The walk of robot r_0 along the cycle corresponds to the thick polyline.

When a robot moves from any given position p on line L to the position $p + 1$ (or $p - 1$) such a robot has completed a tour along the cycle in counterclockwise (or respectively. clockwise) direction. For example r_0 in Fig. 3.6 has completed two counterclockwise tours along the cycle between time t_0 and t_3. We show first, that the feasibility of the localization problem does not change when the initial speeds of all robots are increased, or decreased by the same value.

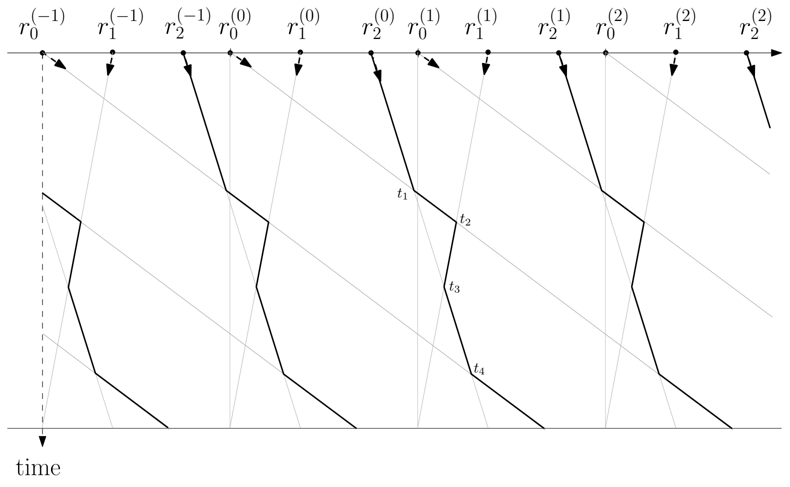

FIGURE 3.6. Trajectory of robot r_0 corresponds to the thick polyline. The times of its first four collisions are also shown

DEFINITION 1. *A translation of a system* $\mathcal{S} = (\mathcal{P}, (v_0, \ldots, v_{n-1}))$ *is a system* $\mathcal{S}_c = (\mathcal{P}, (v'_0, \ldots, v'_{n-1}))$*, where* $v'_i = v_i - c, 0 \leq i \leq n-1$*, for* $c \in \mathbb{R}$.

LEMMA 6. *Let* $\mathcal{S}$ *be a system of robots and let* $\mathcal{S}_c$ *be any of its translations. For every time* t*, velocities* v_i *and* v_j *are exchanged in* $\mathcal{S}$ *at time* t *if, and only if at time* t *velocities* $v_i - c$ *and* $v_j - c$ *are exchanged in* $\mathcal{S}_c$.

Proof. Since $\mathcal{S}$ is also a translation of $\mathcal{S}_c$, it is enough to prove the lemma in one direction. Consider any translation $\mathcal{S}_c$ of system $\mathcal{S}$ and let v'_i and v'_j be such that $v'_i = v_i - c$ and $v'_j = v_j - c$ for some $c \in \mathbb{R}$ and let b_i, b_j, b'_i, and b'_j be the corresponding batons of velocities v_i, v_j, v'_i, and v'_j respectively.

Since the times of exchange of velocities v'_i and v'_j coincide with the times of exchange of batons b'_i and b'_j, it is enough to prove that for every time t in which batons b_i and b_j are exchanged in $\mathcal{S}$, so are batons b'_i and b'_j in $\mathcal{S}_c$.

We prove first that the time of the first meeting of b_i and b_j in $\mathcal{S}$ is the same as the time of the first meeting of b'_i and b'_j in $\mathcal{S}_c$. The statement is clearly true when $v_i = v_j$, since in both systems $\mathcal{S}$ and $\mathcal{S}_c$ batons b_i and b_j stay forever at the same distance on the cycle and no meeting ever occurs. Suppose then, by symmetry, that $v_i \geq 0$ and $v_i > v_j$. Let d be the initial counterclockwise distance (i.e. at time $t = 0$) from b_i to b_j, i.e. $d = d^{(+)}(b_i(0), b_j(0))$. Observe that the batons approach at the speed equal to $|v_i - v_j|$. (Note that this holds as well when robots have different directions, i.e. $v_j < 0$: then $|v_i - v_j| = |v_i| + |v_j|$). Hence the first meeting of b_i and b_j occurs at time $t^* = \frac{d}{|v_i - v_j|}$. However in $\mathcal{S}_c$ we have $v_i' = v_i - c > v_j - c = v_j'$ and again the two batons approach reducing their original distance d with speed $|v_i' - v_j'| = |v_i - v_j|$ meeting eventually after the same time t^*.

A careful reader may observe that the above argument holds independently of the directions that the batons b_j, b_i' and b_j' may have.

Observe that the same analysis holds by induction for the k-th meeting of the robots, for $k = 2, 3, \dots$ Indeed, if $i < j$, i.e. $p_i < p_j$, and $d = p_j - p_i$, then the k-th meeting of b_i and b_j corresponds to the intersection of the trajectory of the copy $b_i^{(0)}$ of baton b_i with the copy $b_j^{(k-1)}$ of baton b_j. As their initial distance on L equals $d + k - 1$, this meeting occurs at time $t^* = \frac{d+k-1}{|v_i - v_j|}$ in both systems $\mathcal{S}$ and $\mathcal{S}_c$. If $i > j$ we have $d = 1 - (p_j - p_i)$ and the k-th meeting of b_i and b_j corresponds to the intersection of the trajectories of $b_i^{(0)}$ and $b_j^{(k)}$, which happens at time $t^* = \frac{d+k}{|v_i - v_j|}$ in both systems $\mathcal{S}$ and $\mathcal{S}_c$.

■

An example from Figure 3.7, illustrates Lemma 6. The walk of robot r_1 is represented by a thick polyline to illustrate how the walk of a robot is affected in a translation of a system.

LEMMA 7. *Let $\mathcal{S}$ be a system of robots and $\mathcal{S}_c$ any of its translations. If t_i is the time of the i-th collision of robot r_q in $\mathcal{S}$, and t_i' the time of the i-th collision of robot r_q in $\mathcal{S}_c$, then $t_i = t_i'$ for $i \geq 1$, where $t_0 = t_0' = 0$. Moreover, if $v(t_i)$ is the velocity of robot r_q at time t_i, then the velocity of r_q at time t_i' is $v(t_i) - c$.*

Proof. Assume the lemma holds for i, so at time t_i robot r_q obtains some baton b_j in $\mathcal{S}$, while at the same time r_q obtains the corresponding baton b_j' in $\mathcal{S}_c$. Let t_{i+1} denote the first time moment after t_i when baton b_j meets another baton in $\mathcal{S}$, say

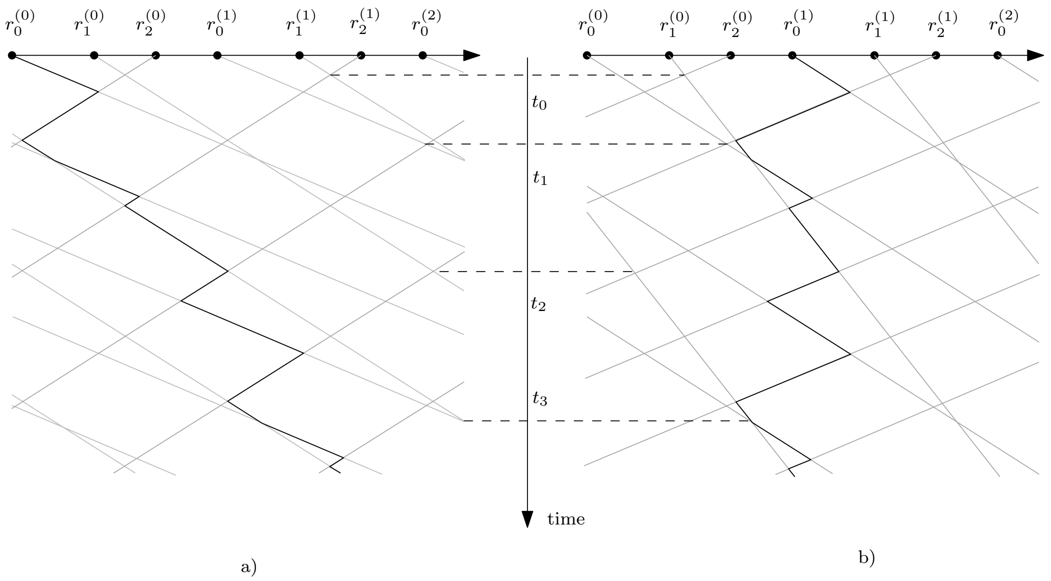

FIGURE 3.7. *a*) depicts a system of three robots r_0, r_1, r_2 whose velocities are $3, 2, -2$ respectively, *b*) depicts a translation of *a*) with new velocities $2, 1, -3$ (after subtracting 1 to each original velocity). Notice that the time at which each collision takes place does not get affected.

b_k. By Lemma 6 at the same time t_{i+1} baton b'_j meets b'_k in $\mathcal{S}_c$. As $v'_k = v_k - c$ the claim follows.

■

We show below, that every robot, by each of its collisions, acquires information about the initial position (relative to its own initial position) and initial velocity of some other robot of the system. We show later, that if $\mathcal{S}$ is feasible, at some time moment the collision revealing the position and velocity of any other robot will eventually arise. However it is worth noting that up to that time moment, some collisions revealing the positions of the same robot may arise several times. We assume that, at time $t = 0$ each robot learns about its initial velocity.

LEMMA 8. *Consider the collisions obtained by robot r_q deployed at its initial position p_q in the cycle. Suppose that the last collision of robot r_q, at time t_i revealed some robot r_s, of initial velocity v_s and initial position at counterclockwise distance d_s from p_q, i.e. $d_s = d^{(+)}(p_q, p_s)$. Further assume that, at time t_{i+1}, robot r_q collides* *obtaining velocity v_t. Then there exists in $\mathcal{S}$ a robot r_t with initial velocity v_t such that $d^{(+)}(p_q, p_t) = ((v_s - v_t)t_{i+1} + d_s) \mod 1$.*

Proof. Between time t_i and t_{i+1} robot r_q moves with velocity v_s so we may assume that it holds baton b_s. The time of collision t_{i+1} corresponds to the time of intersection of the trajectory of some copy $b_s^{(j)}$ of this baton with the trajectory of some copy $b_t^{(k)}$ of the baton moving with velocity v_t. The absolute distance in L between the starting positions $b_s^{(j)}(0)$ and $b_t^{(k)}(0)$ equals $|v_s - v_t|t_{i+1}$. Therefore $d^{(+)}(p_s, p_t) = (v_s - v_t)t_{i+1} \mod 1$. Since by the assumption of the Lemma $d^{(+)}(p_q, p_s) = d_s$, we have $d^{(+)}(p_q, p_t) = (d^{(+)}(p_q, p_s) + d^{(+)}(p_s, p_t)) \mod 1 = ((v_s - v_t)t_{i+1} + d_s) \mod 1$. ∎

Figure 3.8 illustrates Lemma 8 in which a robot r_q during its times of collisions localizes the starting configurations of other robots.

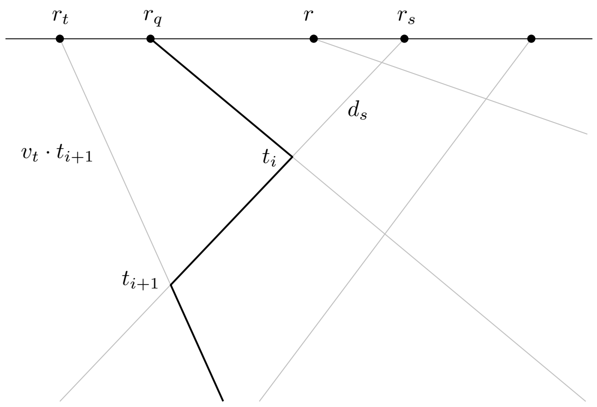

FIGURE 3.8. Trajectory of robot r_q corresponds to the thick polyline

It follows from Lemma 8 that for a robot to figure out the starting position of every other robot it should acquire every velocity of the system in a finite amount of time. Lemma 8 provides the core of an algorithm for robots to report the starting position of every robot. We describe such an algorithm later on. The next lemma is an immediate consequence of Lemma 3.7 and Lemma 8.

LEMMA 9. *For any system $\mathcal{S}$ and its translation $\mathcal{S}_c$, the position discovery problem is solvable for $\mathcal{S}$, if and only if it is solvable for $\mathcal{S}_c$.*

Given a fixed point ρ in the cycle, which we call the *reference point* and $\mathcal{S}$ a system of robots, we associate with each robot r_i an integer counter c_i that we call *cycle counter*. A cycle counter c_i increases its value by one each time robot r_i traverses the reference point ρ in the counterclockwise direction and decreases by one when traversing ρ in clockwise direction. We denote by $c_i(t)$ the value of cycle counter c_i at time t. The initial value of c_i is set to 0, meaning $c_i(0) = 0$.

Let $D_i^{(+)}(t)$ denote the total distance that robot r_i traveled until time t in the counterclockwise direction, and $D_i^{(-)}(t)$ - the total distance traveled by r_i in the clockwise direction. Denote $D_i(t) = D_i^{(+)}(t) - D_i^{(+)}(t)$. The following observation is the immediate consequence of $\sum_{i=0}^{n-1} v_i = 0$ for system $\mathcal{S}_{\bar{v}}$:

Observation 1. *For any system $\mathcal{S}_{\bar{v}}$ at any time moment t we have $\sum_{i=0}^{n-1} D_i(t) = 0$.*

Lemma 10. *Consider the translation $\mathcal{S}_{\bar{v}}$ of any system $\mathcal{S}$. At any time t, no two cycle counters differ by more than 1, i.e $|c_i(t) - c_j(t)| \leq 1, 0 \leq i, j \leq n-1$. Moreover, there should be a cycle counter$c_{k(t)}$ such that $c_{k(t)}(t) = 0$ for some $0 \leq k(t) \leq n-1$.*

Proof. Let us observe that since robots can not overpass each other they always keep their initial cyclic order. Therefore, we can simulate the traversals on ρ by the robots by assuming that robots remain static while ρ is moving in one of the two directions along the cycle; when ρ traverses a robot r_i in clockwise direction, counter c_i increases by one and decreases by one if ρ traverses r_i in counterclockwise direction.

We prove first that $|c_i(t)| \leq 1$, for each $0 \leq i \leq n-1$. Indeed, suppose to the contrary, that $|c_i(t)| \geq 2$. Consider first the case when $c_i(t) \geq 2$. In such a case, r_i must have traversed point ρ at least two more times in the counterclockwise direction than in the clockwise one. Since the robots do not change their relative order around the cycle, each other robot r_j must have traversed ρ at least once more in the counterclockwise direction than in the clockwise one. Hence $D_i^{(+)}(t) > D_i^{(-)}(t)$ for each $i = 0, \ldots, n-1$. This contradicts $\sum_{i=0}^{n-1} D_i(t) = 0$. The argument for $c_i(t) \leq -2$ is symmetric.

It is easy to see that there are no two robots r_i, r_j, such that $c_i(t) = 1$ and $c_j(t) = -1$. Indeed in such a case these robots must have traversed point ρ in opposite directions which would have forced them to overpass - a contradiction.

Hence the values of all cycle counters at time t belong to the set $\{0, 1\}$ or to the set $\{0, -1\}$. However $c_i(t) \neq 0$, for all $i = 0, \ldots, n-1$ would imply $D_i(t)$ be all positive or all negative, contradicting $\sum_{i=0}^{n-1} D_i(t) = 0$, which concludes the proof. ∎

We can conclude with the following Corollary:

COROLLARY 4. *For each robot r_i of $\mathcal{S}_{\bar{v}}$ and any time t we have $|D_i(t)| < 1$.*

Proof. Suppose to the contrary that $|D_i(t)| \geq 1$ or, by symmetry, that $D_i^{(+)}(t) - D_i^{(-)}(t) \geq 1$. In such a case, r_i at time t made a full counterclockwise tour around the cycle. By putting the reference point $\rho = r_i(0)$, we notice that this forces each other robot r_j to have $c_j(t) \geq 1$, which contradicts Observation 1. ∎

Fig. 3.9 depicts a system of mobile robots, where the average of the velocities is equal to 0. Notice that every robot in the picture never completes more than one round along the cycle in any direction. In the picture the movements of r_0 are shown with a thick polyline to illustrate this.

THEOREM 6. *For any system of n mobile robots $\mathcal{S} = (\mathcal{P}, V)$ the localization problem is feasible if and only if $v_i \neq \bar{v}$, for every $v_i \in V$, Moreover, if the problem is feasible, then each robot knows the positions and the velocities of other robots before time $T = \frac{2}{min_{0 \leq i \leq n-1} |v_i - \bar{v}|}$.*

Proof. By Lemma 9, it is sufficient to prove the theorem for $\mathcal{S}_{\bar{v}} = (\mathcal{P}, V_{\bar{v}})$.

We prove first, that if some robot r_i has the initial velocity $v_i = \bar{v} = 0$, then the system is not feasible. For the localization problem to be feasible in $\mathcal{S}_{\bar{v}}$, each robot must hold every baton at some time within some finite time interval $[0, T]$. We prove

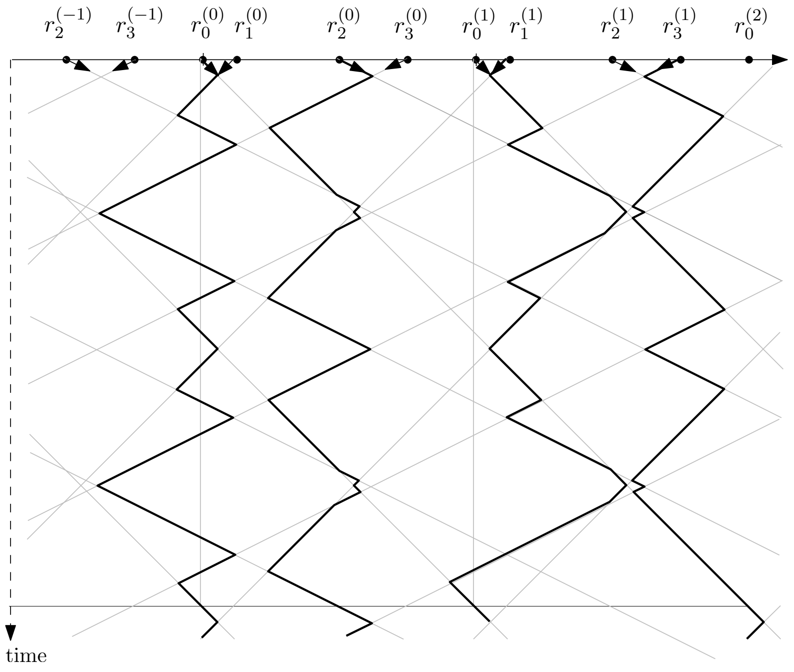

FIGURE 3.9. An example of a system of robots where the average of the velocities is equal to 0. Notice that no robot completes more than one round in any direction.

by contradiction that, if there is a baton b_q of velocity 0, then there exists a robot whose trajectory will not intersect the trajectory of b_q. Thus, such a robot would not obtain the information about the velocity and the position of robot r_q.

Consider cycle counters $c_j(t)$ for each robot r_j, $0 \leq j \leq n-1$, where the reference point is set to $\rho = r_q(0)$. Because $v_q = 0$, there is always a robot of $\mathcal{S}_{\bar{v}}$ that remains motionless at point ρ. In other words, each robot of $\mathcal{S}_{\bar{v}}$, in order to hold baton b_q has to move to position ρ and collide with the current robot at that position. Observe that it is not possible that all robots arrive at point ρ from the same direction around the cycle. Indeed, in such a case the robot velocities would be all positive or all negative implying $\bar{v} \neq 0$. Consequently, observe that there must exist two time moments t_1, t_2 and two consecutive robots r_i and r_{i+1} (where index $i+1$ is taken modulo n) such that one of these robots visited ρ at time t_1 while walking in one direction and the other robot visited ρ at time t_2 while walking in the opposite direction. Notice that $t_1 \neq t_2$, since we supposed no three robots meeting simultaneously, and ρ coincides with a stationary robot.

Suppose, that r_i arrived at ρ at time t_1 while walking clockwise and r_{i+1} arrived at ρ at time t_2 while walking counterclockwise. As robots are arranged in the counterclockwise order around the cycle it follows that within the time interval $[t_1, t_2]$ each other robot has to walk counterclockwise through ρ (or walk more times counterclockwise than clockwise) increasing its cycle counter.

Let $\mathcal{S}' = (\mathcal{P}', V_{\bar{v}})$, where $\mathcal{P}' = (r'_0(t_1), \ldots, r'_{n-1}(t_1))$ and let c'_j be the respective cycle counter of robot r'_j for every $0 \leq j \leq n-1$. Notice that during the interval of time $[0, t_2 - t_1]$ every robot of $\mathcal{S}'$ behaves exactly the same way as does every robot in $\mathcal{S}_{\bar{v}}$ in the interval of time $[t_1, t_2]$. Thus, at time $t^* = t_2 - t_1$ we have $c'_j(t^*) > 0$ for all $0 \leq j \leq n-1$ which contradiction Lemma 10. This implies that there is at least one robot that does not learn the initial position of all robots.

The cases where r_{i+1}, rather than r_i, arrived at ρ at time t_1 and when the directions of r_i and r_{i+1} while walking through ρ are reversed, are symmetric.

Suppose now that no robot has the initial velocity $\bar{v} = 0$. Consider any robot r_i and the interval $I_2 = [r_i(0) - 1, r_i(0) + 1]$ of the infinite line L. By Corollary 4 robot r_i never leaves interval I_2 during its movement, hence its trajectory is bound to the vertical strip of width 2 (cf. Fig. 3.9). Consider any baton b_j. Suppose, by symmetry, that $v_j > 0$. Take the trajectory of a copy of baton b_j which origins from the left half of I_2, i.e. from the segment $[r_i(0) - 1, r_i(0]$. This trajectory will go across the vertical strip of width 2 enclosing I_2 and leave it before time $\frac{2}{|v_j|}$, forcing the meeting of robot r_i and baton b_j. If $v_j < 0$ we need to take a copy of b_j starting at the right half of I_2, i.e. in $[r_i(0), r_i(0) + 1]$ and the argument is the same. The time of $\frac{2}{|v_j|}$ is maximized for j minimizing $|v_j|$. ∎

An example of an infeasible robot configuration is shown in Fig. 3.10, in which robot r_0 never learns the initial position of robot r_1 since it never acquires velocity $\bar{v}$.

3.3. The Localization Algorithm. In this section we present an algorithm to solve the localization problem. The algorithm is based on Lemma 8. According to Theorem 6, if robots have knowledge of the velocities of other robots on the cycle,

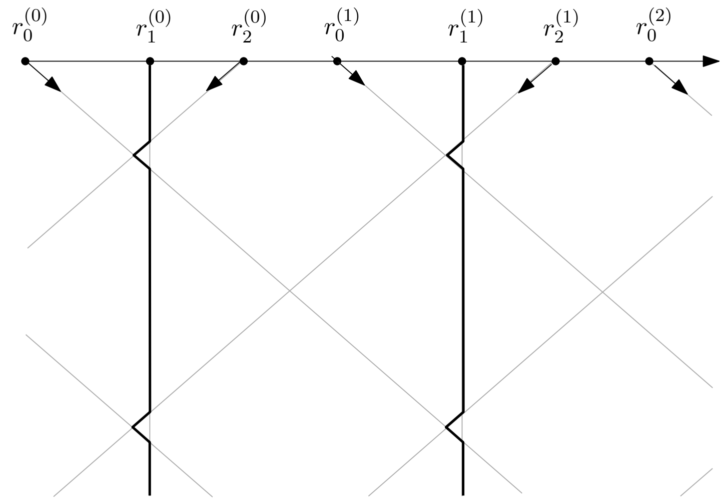

FIGURE 3.10. Example of an infeasible configuration. Robot r_0 never learns about r_1.

it is possible for them to detect the in-feasibility of the system without even starting to move, or stop it when the variable *clock* reaches the value of $\frac{2}{min_{0\leq i\leq n-1}|v_i-\bar{v}|}$. Otherwise, the algorithm is designed to run indefinitely. However, we can also assume that a central authority, perhaps having the knowledge of the velocities of the robots in the system, may modify robot's signal variable *move* to halt the execution. The present algorithm may report the position of the same robot more than once; this may be clearly avoided providing robots with linear-size memory to recall all previously output robots.

The main theorem ensures that at this time all robots have discovered all the initial positions if the system is feasible.

In algorithm `RingLocalization` we assume that a robot has at any time immediate access to its clock as well as to the information of its current velocity through the variables *clock* and *velocity*, respectively. So the value of these variables can not be modified by the robot and they get updated instantaneously as a collision happens. We can assume that the values of these variables correspond to the readings of robots' sensors. A robot uses auxiliary variables, namely *old_velocity* and *pos* for recalling the position and the velocity of the robot detected through its last collision.

```
Algorithm RingLocalization
1.   var $pos \leftarrow 0, old\_velocity \leftarrow velocity$: real;
     $move \leftarrow$ true : boolean;
2.    reset $clock$ to 0;
3.    while $move$ do
4.       walk until collision;
5.       $pos \leftarrow ((velocity - old\_velocity) \cdot clock + pos) \mod 1$;
6.       output("Robot of velocity" $velocity$
                          "detected at position" $pos$);
7.       $old\_velocity \leftarrow velocity$;
```

A robot r_q executing Algorithm `RingLocalization` moves until a collision with another robot takes place. At the presence of a collision it acquires a new velocity v_s, using this velocity (and its previous velocity stored in *old_velocity*) it computes the starting position p_s (with respect to r_q's starting position) of the corresponding robot q_s. Notice that variable *pos* clearly keeps track of $d_s = d^{(+)}(p_q, p_s)$ of Lemma 8. Because of this and Theorem 6 the next results follows immediately.

Corollary 5. *Let $\mathcal{S} = (\mathcal{P}, V)$ be a system of robots. Suppose that no robot has initial velocity $\bar{v}$, meaning $v_i \neq \bar{v}$ for all $v_i \in V$, and that the algorithm* `RingLocalization` *is executed by each robot for time $\frac{2}{min_{0\leq i\leq n-1}|v_i-\bar{v}|}$. Then, every robot correctly reports the initial positions and directions of all robots on the cycle with respect to its initial position.*

For two robots at small distance ϵ, starting in opposite directions with small velocities v_1 and $v_2 = -v_1$ it takes time $\frac{1-\epsilon}{2v_1}$ to get the first collision, so the worst-case time of localization algorithm proportional to $\frac{1}{min_{0\leq i\leq n-1}|v_i-\bar{v}|}$ is unavoidable. Observe that without the knowledge of velocities, even if the number of robots in the system is known, it is impossible for a robot to decide at any time if the system is infeasible.

Corollary 6. *For any robot r in any system system, there is no time bound for r to decide whether it has localized all the starting positions of all the robots.*

Proof. By Theorem 6 to any system $\mathcal{S}$ it is possible to add a new robot of velocity equal to the average $\bar{v}$, making $\mathcal{S}$ infeasible for at least some robot r_i of $\mathcal{S}$. Consequently, given arbitrarily large time T^* it is also possible to add to $\mathcal{S}$ a robot of velocity close to $\bar{v}$, so the system stays feasible but not within the time bound of T^*. ∎

4. Conclusions

Notice that the evenly spread problem of n robots on a cycle can be performed using bouncing robots for all cases in which there is a majority on one of the two starting directions of the robots, i.e, for all cases in which solving the localization problem takes more than a constant time. After executing `RingBounce`, robots figure out the system of coordinates of all robots, hence they simply agree upon an orientation of the cycle. For instance, if the majority starts moving in counter clockwise direction they agree upon such a direction as the clockwise direction of the cycle, and then they execute the algorithm described in [27] to evenly spread themselves in the cycle. Notice that we would also have to allow robots to have control of their movements.

An open problem is investigating the localization problem by bouncing robots of different masses this scenario seems to be challenging since the resulting dynamics of the system of robots is much more complex. In this case, robots do not exchange velocities any more so probably robots would need to be enhanced with new capabilities for them to solve the task.

CHAPTER 4

Survivability of Bouncing Robots

In this section we study the survivability of swarms of bouncing robots. To do so, we mark the starting position of every robot as *deadly* in the sense that if a robot ever returns to its starting position it dies. A robot *survives* if it never returns to its starting position. We investigate the necessary conditions for swarms of bouncing robots to have surviving subsets in the cycle and in the segment. Since bouncing robots do not have any control over their movements, it might seem that in most configurations all the robots must die. We prove that this is not always the case, thus answering an open question first posed in [46] for the case of robots of equal masses and speeds.

1. Results

In Section 3, we prove some general properties regarding the dynamics of bouncing robots that are crucial for understanding their survivability. In Section 4, we study the survivability of swarms of robots of equal masses and speeds. We prove that if the robots are deployed in a segment they all must die. In contrast, in the cycle we show the existence of swarms with two survivors for a swarm of $n \geq 4$ robots and we prove that the smallest swarm with survivors has four robots with exactly two survivors. We also prove a lower bound on the number of robots that die if not all robots have the same initial direction. For the case of robots of arbitrary masses and velocities, in Section 5, we prove that, if robots are deployed in a segment, the survivors, if any, must become indefinitely immobile. However, this is not the case when the robots are deployed in a cycle. We show that in the cycle at least one robot dies and that the maximum number of robots that can survive either in the cycle or in the segment is $n-1$. We conclude by listing some open problems in Section 5.

The results of this chapter were published in [14].

2. Preliminaries

Let $\mathcal{S}_n = (\mathcal{M}, \mathcal{H}, \mathcal{U})$ be a swarm of n bouncing robots $r_0, r_1, \ldots, r_{n-1}$ with masses $\mathcal{M} = (m_1, m_1, \ldots, m_n)$, starting positions or home bases $\mathcal{H} = (h_1, h_1, \ldots, h_n)$, and non-zero initial velocities $\mathcal{U} = (u_1, u_2, \ldots, u_n)$, respectively. We call the *size* of a swarm the total number of robots.

We study the question concerning robots moving in a segment and in a cycle. We assume that each robot starts moving at the same time with constant speed until a collision takes place. When two robots collide, they update their velocities following the conservation of momentum and conservation of energy principles. For the case of the segment, its end points model walls in which robots can collide with. When a robot collides with a wall it reverses direction but keeps moving with the same speed. Recall that throughout this thesis, we assume that collisions are elastic and that in any collision no more than two robots participate. Thus, if two robots of equal masses collide, they simply exchange velocities. If robots r_1 and r_2 of masses m_1 and m_2, and velocities u_1 and u_2 respectively, collide, after their collision they get new velocities v_1 and v_2 , respectively following equations 1.

A cycle of perimeter l is modeled by the real interval $[0, l]$, with 0 and l corresponding to the same point. By $r_i(t) \in [0, l]$ we denote the position of robot r_i at time t. We suppose that originally each robot r_i occupies the position $r_i(0) = h_i$ of its environment and that $0 \leq r_1(0) < r_2(0) < \cdots < r_n(0)$. Each robot is given an initial direction, clockwise (CW) or counterclockwise (CCW) in the cycle and left-to-right or right-to-left on the segment, according to which it starts its movement. By dir_i we denote the starting direction of robot r_i and we set $dir_i = 1$ if r_i starts its movement in the counterclockwise direction around the cycle or the left-to-right direction along the segment. By $dir_i = -1$ we denote the clockwise starting direction (on the cycle) or right-to-left (on the segment).

By $\mathcal{S}^+$ we denote the number of robots in the swarm $\mathcal{S}$ whose initial direction is counterclockwise and by $\mathcal{S}$ the number of robots in the swarm $\mathcal{S}^-$ whose initial direction is clockwise. We identify the counterclockwise and left-to-right directions as positive.

A robot dies if at some time it returns to its home base. If a robot r_a dies, it disappears from the environment and does not interact with any other robot any more. On the other hand, we say that a robot survives if it does not die. If D is a subset of robots of a swarm $\mathcal{S}_n$ that die at some time, $\mathcal{S}_n \setminus D$ denotes the resulting swarm of survivors after the death of the robots in D.

The death of a robot takes priority over collisions, i.e, if two robots collide at the home base of one of them, the death of the robot returning to its home base takes place first and the other robot keeps moving without updating any of its parameters.

If the collisions of a swarm are not concurrent (no two different collisions take place at the same time), we can enumerate the collisions. Notice that this is not possible in most cases since it is plausible that two different pairs of robots may collide at the same time. If collisions can be enumerated, we denote the i-th collision of a given swarm by $C_i = (a^{(dir)}, b^{(dir')})$, where a and b are the robots involved in the i-th collision, and $dir, dir' \in \{+, -\}$, denote the directions that the robots had before colliding. We use $(+)$ to denote the CCW direction and $(-)$ to denote the CW direction. We use the notation $u_i^{(j)}$ to indicate the velocity of robot r_i resulting after the j-th collision of the swarm, where $u_i^{(0)} = u_i$ denotes the initial velocity of robot r_i.

Since we use some properties of classical mechanics to prove our results, we define some useful concepts that extend from the properties of systems of particles. Suppose we have a swarm $\mathcal{S}_n$ of bouncing robots $r_1, r_2, \ldots, r_n$, of initial velocities $u_1, u_2, \ldots, u_n$ and masses $m_1, m_2, \ldots, m_n$, respectively. Analogously to systems of particles we define the momentum of the swarm as $P(\mathcal{S}_n) = \sum_{i=1}^{n} m_i u_i$, where $m_i u_i$ is the linear momentum of robot r_i. The velocity of the swarm is defined as $U(\mathcal{S}_n) = \frac{P(\mathcal{S}_n)}{M}$, where $M = \sum_i^n m_i$. Finally, the kinetic energy of $\mathcal{S}_n$ is given by $KE(\mathcal{S}_n) = \sum_{i=1}^{n} m_i u_i^2$.

3. General Behavior of Swarms of Bouncing Robots

In this section we focus on studying the dynamics of bouncing robots as they are deployed either in a segment or a cycle. In all these results, we do not assume that robots die since we focus only on studying their dynamics. We denote the infinite line by $\mathcal{L} = (-\infty, \infty)$, and the half positive semi-line by $\mathcal{L}^+ = (0, \infty)$, where 0 represents the wall on which the leftmost robot may collide. We say that a swarm $\mathcal{S}_n$ deployed on $\mathcal{L}$, *expands to the right* (respectively to the left) if and only if:

(1) there exists $t_0 \geq 0$, such that for every $t \geq t_0$, no more collision takes place, and

(2) for any $b > 0$, there exists some robot r_m and time t_m such that $r_m(t_m) > b$ (respectively $r_m(t_m) < a$, for arbitrary $a < 0$).

We say that $\mathcal{S}_n$ expands in both directions, if and only if $\mathcal{S}_n$ expands to the right and to the left.

Theorem 1. *Let $\mathcal{S}_n$ be any swarm of bouncing robots deployed on $\mathcal{L}$, such that $KE(\mathcal{S}_n) \neq 0$. Then, for any finite segment $[a, b] \subset \mathcal{L}$, there exists a time $t^\star$, such that for any $t' > t^\star$ some robot $r_m(t') \notin [a, b]$. Moreover, if the swarm has either positive or negative or zero momentum, the swarm expands to the right or to the left or in both directions, respectively.*

Proof. Let $\mathcal{S}_n$ be a swarm of robots, such that $KE(\mathcal{S}_n) \neq 0$, and let $[a, b] \subset \mathcal{L}$ be any finite segment. We assume that $h_i \in [a, b]$, for $i \geq 1$. Since the number of elastic collisions in $\mathcal{L}$ for any system of particles is finite [48], and the robots of $\mathcal{S}_n$ behave exactly as particles, the number of collisions in $\mathcal{S}_n$ is also finite. More precisely, there exists some $t_0 \geq 0$, such that for any $t \geq t_0$ no more collisions take place among the robots of the swarm. Because of the principle of conservation of kinetic energy,

$KE(\mathcal{S}_n) \neq 0$ at any time, meaning that at any time there exists one robot r_m that is still moving on the line. Let us assume that $r_m(t_0) \in [a, b]$ and that r_m is moving to the right with velocity v, then $r_m(t') > b$ for any time $t' > t^\star$, where $t^\star = \frac{|b - r_m(t_0)|}{|v|}$, for arbitrary $b \in \mathcal{L}$. Analogously, if r_m is moving to the left, $t^\star = \frac{|a - r_m(t_0)|}{|v|}$. This finishes the proof of the first part of the theorem. Notice that since we are assuming the principle of conservation of momentum, at any time bigger than t_0 the momentum of the system remains the same. Thus, if $P(\mathcal{S}_n) < 0$, not all robots might have positive velocities. Thus, $\mathcal{S}_n$ expands to the left. Analogously, if $P(\mathcal{S}_n) > 0$, there must exist some robot moving with positive velocity. If $P(\mathcal{S}_n) = 0$, the swarm must expand in both directions. ∎

In the next corollary, we assume that robots are deployed on the half infinite line. The origin models a wall, if the leftmost robot collides with the wall, it bounces back, i.e, the robot reverses direction but keeps moving with the same speed.

Lemma 11. *Let $\mathcal{S}_n$ be any swarm of bouncing robots deployed on $\mathcal{L}^+$. There exists a swarm $\mathcal{S}_{2n}$ of bouncing robots deployed on $\mathcal{L}$, such that $P(\mathcal{S}_{2n}) = 0$ and half of its robots mimic the dynamics of the robots in $\mathcal{S}_n$.*

Proof. We define $\mathcal{S}_{2n}$ in the following way: for each $r_i \in \mathcal{S}_n$ with initial position $r_i(0) \in \mathcal{L}^+$ and initial direction dir_i, we define robots $r_{\hat{i}}$ and r_{-i} in $\mathcal{S}_{2n}$, such that $r_{\hat{i}}(0) = r_i(0)$, $r_{-i}(0) = -r_i(0)$, $dir_{\hat{i}} = dir_i$ and $dir_{-i} = -dir_i$. Consider the walk of robots r_i, for $0 \leq i \leq n$, within the line $\mathcal{L}^+$. Let $t_1 < t_2 < \cdots$ be the sequence of times when a collision takes place in $\mathcal{S}_n$. It is easy to see by induction on i that at any time t, $r(t)_{\hat{i}} = r_i(t)$ and that $r(t)_{-i} = -r_i(t)$ in $\mathcal{L}^+$. ∎

The following corollary follows from the Theorem 1 and Lemma 11.

Corollary 1. *Any swarm $\mathcal{S}_n$ of bouncing robots of arbitrary masses and velocities that are deployed on $\mathcal{L}^+$, such that $KE(\mathcal{S}_n) \neq 0$, expands to the right.*

Proof. Take swarm $\mathcal{S}_{2n}$ of Lemma 11. Since $P(\mathcal{S}_{2n}) = 0$ and $KE(\mathcal{S}_{2n}) \neq 0$, Theorem 1 implies that $\mathcal{S}_{2n}$ expands in both directions. It follows that $\mathcal{S}_n$ expands to the right. ■

Let $D_i^{(+)}(t)$ denote the total distance that robot r_i traveled until time t in the CCW direction, and $D_i^{(-)}(t)$ - the total distance traveled by r_i in the CW direction. Denote $D_i(t) = D_i^{(+)}(t) - D_i^{(-)}(t)$. The next theorem establishes a relationship between the momentum of a swarm and the total distance that any robot traverses.

THEOREM 2. *If $\mathcal{S}_n$ is a swarm of bouncing robots of same masses but arbitrary speeds, such that $P(\mathcal{S}_n) \neq 0$, then all robots eventually complete a full cycle.*

Proof. Without loss of generality assume $P(\mathcal{S}_n) > 0$, then we have that $U(\mathcal{S}_n) > 0$ which implies that $t \cdot (\sum_{i=1}^{n} v_i) > 0$, for any $t > 0$. Therefore, there exists a big enough $t^\star > 0$ such that $D_i(t^\star) \geq 1$ for any $i \geq 1$. ■

4. Robots of equal masses and equal speeds

In this section, we study the survivability of bouncing robots of equal masses and speeds that are deployed either in a segment or a cycle. The following result shows that there are no swarms of equal masses and speeds in the segment that contain surviving robots.

THEOREM 3. *All bouncing robots die of any swarm deployed in the segment.*

Proof. Let $\mathcal{S}_n$ be a swarm of bouncing robots of same speeds and masses. Notice that in $\mathcal{S}_n$ during a collision either with a robot or with a wall, robots simply reverse direction, so if $KE(\mathcal{S}_n) \neq 0$, no robot can remain static at any time even in the presence of the death of a robot. We prove this theorem by induction on the size of the swarm, so we assume that in any swarm of size $n-1$ all robots die. Let $\mathcal{S}_n$ be a swarm of size n. It is enough to prove that one of the extreme robots of $\mathcal{S}_n$ dies. Let r_1 and r_n be the leftmost robot and the rightmost robot, respectively. Notice that if $dir_1 = 1$, r_1 will die after reversing direction. Without loss of generality, let us assume that $dir_1 = -1$. If no robot dies, robot r_1 has to be bumping against the wall and its neighbor r_2 so that it never returns to its home base, for the same reason r_2 is bumping against r_1 and r_3 without reaching its home base and so on. This can only happen if robots $r_1, \ldots, r_{n-1}$ are at the left of their home bases indefinitely, however this can not be true for r_n which after colliding with r_{n-1} reverses direction and dies. The remaining system has $n-1$ robots and because of the induction hypothesis all of them die. ∎

Observation 2. *An interested reader may notice that Theorem 3 also holds for robots of equal masses but arbitrary non-zero speeds.*

Theorem 4. *There exist swarms of size four in a cycle containing two survivors.*

Proof. Let $\mathcal{S}_4$ be a swarm of size four deployed in a cycle of perimeter 80. Let $h_1 = 10, h_2 = 25, h_3 = 30, h_4 = 80$ and $u_1 = u_2 = u_3 = 1, u_4 = -1$ be the corresponding home bases and initial velocities of the robots, respectively. We assume that all the robots have unitary masses. It is easy to check that robots r_1 and r_2 survive while the other two robots die. ∎

The configuration from Theorem 4 is the first correct example of surviving subset of robots. We argue below that the example from [46], represented in Table 1, which supposedly contains a swarm of five robots with four survivors is not correct. The table describes the positions (in degrees) of the robots (in the cycle) at the given time, where + and − indicate the current directions of the robots (+ for CCW and − for CW). The time is given in seconds and it takes 128 seconds for a robot to complete a full cycle. For instance, r_1 has starting position at 120 degrees and CW initial direction and it dies after 110 seconds while the remaining robots live eternally.

Although it is correct that the first robot to die is r_1, it is easy to check that robot r_3 or robot r_2 must die as well. This is because both of them have to reverse direction and inevitably one of them must return to its home base. The following

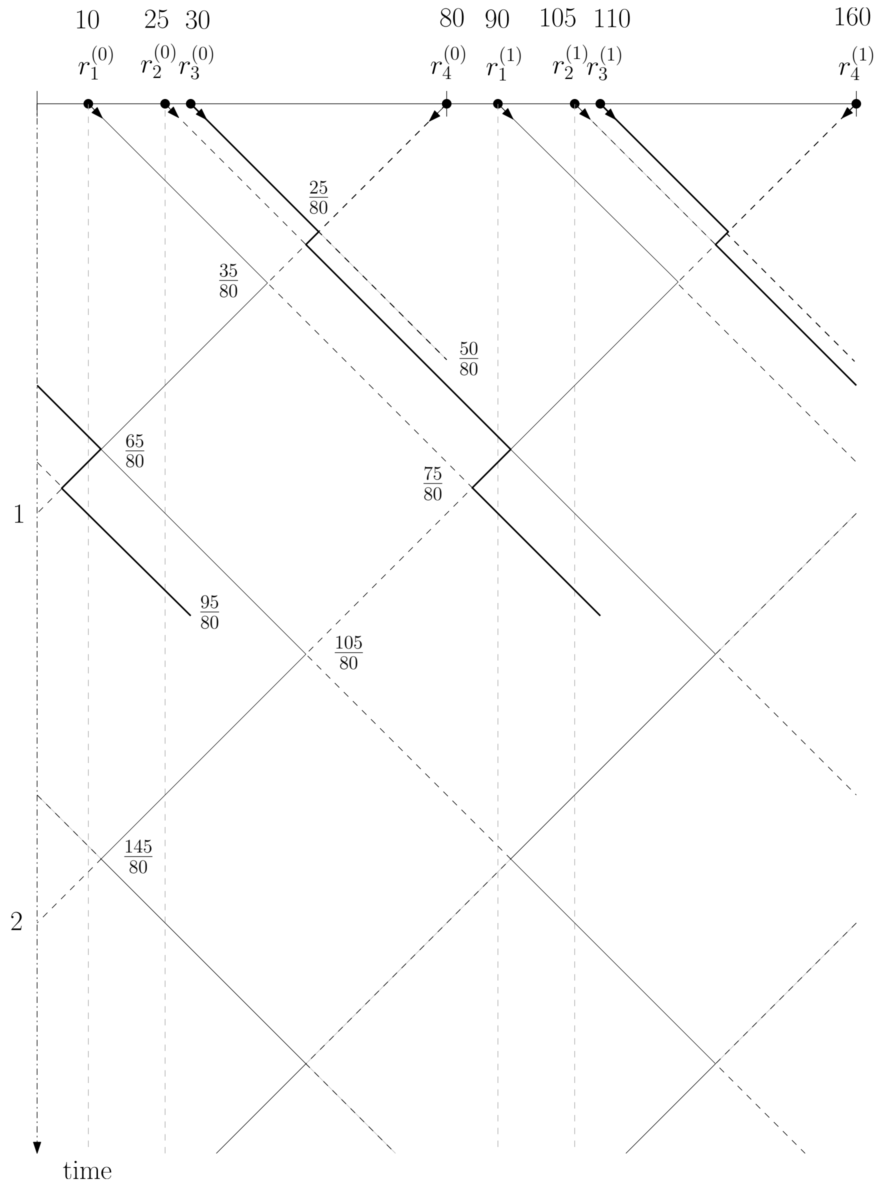

FIGURE 4.1. This figure depicts the walks, times of collision and the death of the robots of Theorem 4. Robots r_3 and r_4 die while r_1 and r_2 live eternally.

time	r_1	r_2	r_3	r_4	r_5
0	−120	−92	+55	+41	+51
110	die	−102	+100	+0	−74
123		+115	−87	−115	+87

TABLE 1. Example given in [46]

theorem states that there are no swarms of smaller size than the swarm of Theorem 4 with survivors.

THEOREM 5. *In the cycle any swarm of size less than four has no survivors.*

Proof. The Theorem is clearly true for swarms of size less than three. Without loss of generality, assume that $1, -1$, and 1 are the initial velocities of robots r_1, r_2 and r_3, respectively. Let us assume that $d^{(+)}(h_3, h_2) = x$ and $d^{(+)}(h_1, h_3) = y$ as Fig 4.2 depicts. Notice that $C_1 = (r_2^{(-)}, r_3^{(+)})$, so let b be the point of the first collision that takes place at time $t_1 = \frac{x}{2}$. Let a be the position of r_1 at such a time. Following Equation 1, r_3 reverses direction after colliding with r_2. In the best scenario, r_3 reaches its home base and dies together with r_2 at time x. Thus, r_1 completes a full cycle and dies. If r_3 does not die, collision $C_2 = (r_2^{(-)}, r_3^{(+)})$ takes place at some point c at time $t_2 = t_1 + \frac{y}{2}$ before r_3 reaches its home base. Notice that $d^{(+)}(a, c) = d^{(+)}(c, b) = \frac{y}{2}$ (see Fig 4.2). It is clear that at time $2t_2$, robot r_1 reaches its home base, at the same time r_3 is at the home base of r_2 which has already died. Even if $h_2 = h_1$, robot r_1 dies since the death of robots take priority over collisions. Therefore, the theorem holds. ■

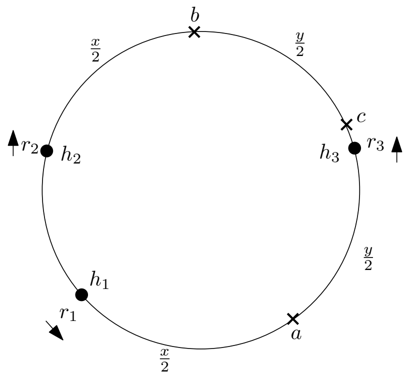

FIGURE 4.2. This figure illustrate theorem 5. No robot survives

For two points p, q in the cycle, by $d^{(+)}(p, q)$ we denote the counterclockwise distance from p to q in the cycle, i.e. the distance which needs to be traveled in the counterclockwise direction in order to arrive at q starting from p. We denote the time of the j-th collision of robot r_i by $t_i^{(j)}$. For simplicity, we scale the perimeter of the cycle of Theorem 4 so as to have a unitary cycle.

Observation 3. *The following observations are crucial to understand the survivability of the survivors of the swarm of Theorem 4.*

(1) *For $i = 1, 2, 3$, we have that:*
   (a) $t_i^{(1)} = \frac{d^{(+)}(h_i,h_4)}{2}$. *Thus for r_i not to die after reversing direction in its first collision, it should get its second hit before time $2t_i^{(1)}$. This can only happen if, there exists some robot r_c such that $dir_c = 1$ and $d^{(+)}(h_c, h_i) < 2t_i^{(1)}$.*
   (b) *robot r_i after its first collision every time that it moves in* CW *direction it does so by exactly $\frac{1}{2}$ time.*

(2) *For r_1 and r_2 to survive, the second hit of r_1 must take place within the interval (h_1, h_2). To guarantee so, for each robot r_l, such that $dir_l = 1$, $2\left(t_1^{(1)} - d^{(+)}(h_1, h_2)\right) < d^{(+)}(h_l, h_2)$, and*

(3) *If robot r_3 participates in a fifth collision, its total distance traversed would be $1 + t_3^{(1)} - \frac{d^{(+)}(h_2,h_3)+d^{(+)}(h_1,h_2)}{2}$. So it must hold that $d^{(+)}(h_3, h_4) - \left(d^{(+)}(h_2, h_3) + d^{(+)}(h_1, h_2)\right) > 0$, since robot r_3 dies after its fourth collision.*

The next theorem is an extension of Theorem 4.

Theorem 6. *There exists a swarm in the cycle of size n of two survivors, for any $n \geq 4$.*

Proof. The idea is to simply extend the swarm of Theorem 4 by inserting an arbitrary number of robots r_a that copy the behavior of r_3, i.e, that die after their fourth collision without disturbing the survivability of r_1 and r_2. Notice that if we add a new robot r_a between robots r_2 and r_3, we have that its first collision takes place at time $t_a^{(1)} = \frac{d^{(+)}(h_a,h_4)}{2} > t_3^{(1)}$. Further, because of Observation 2, it holds that $t_a^{(1)} - \left(d^{(+)}(r_2, r_a) + d^{(+)}(r_1, r_2)\right) > 0$. Thus, at time $t_a^{(5)}$, the total distance covered by r_a would be $1 + t_a^{(1)} - \frac{d^{(+)}(h_2,h_a)+d^{(+)}(h_1,h_2)}{2}$ which is also bigger than one and therefore r_a would die after its fourth collision. Moreover, after inserting r_a, robot r_3 still dies since the total distance covered in *ccw* direction until its new fourth collision would be $d^{(+)}(h_a, h_3) + d^{(+)}(h_2, h_a) < d^{(+)}(h_2, h_3)$. Finally, notice that the addition of r_a would not make the second hit of r_1 to happen outside of $\frac{d^{(+)}(h_a,h_3)}{2}$ since Observation 2 holds for r_3. It is easy to check that Observation 1 holds after the insertion of r_a. We can repeat this procedure as many times as we want by adding new robots and still having r_1 and r_2 as survivors. Therefore, the theorem holds. ■

The following corollary is an immediate consequence of Theorem 2.

COROLLARY 2. *For any swarm $\mathcal{S}$ in the cycle at least $|\mathcal{S}^+ - \mathcal{S}^-|$ robots die.*

Proof. Let $\mathcal{S}$ be a swarm of bouncing robots of equal masses and speeds, we have that $P(\mathcal{S}) = ms(\mathcal{S}^+ - \mathcal{S}^-)$, where s and m denote the values of the common speed and mass of the robots, respectively. Theorem 2 guarantees that unless $\mathcal{S}^+ - \mathcal{S}^- = 0$, the swarm moves forward in the direction of the majority and thus there exists some robot that returns to its home base and dies. It follows that in $\mathcal{S}$ at least $|\mathcal{S}^+ - \mathcal{S}^-|$ robots die. ■

5. Robots of Arbitrary Masses and Velocities

Recall that If the collisions of a swarm are not concurrent, $C_i = (a^{(dir)}, b^{(dir')})$ denotes the i-th collision in the swarm, where a and b are the robots involved in such collision and $dir, dir' \in \{+, -\}$, denote the directions that the robots had before they collide.

THEOREM 7. *For any $n \geq 2$ there exists a swarm $\mathcal{S}_n$ of bouncing robots of size n in the segment, such that:*

(1) *$u_1 > 0$ and $u_i < 0$, for all $2 \leq i \leq n$.*
(2) *$C_i = (r_i^{(+)}, r_{i+1}^{(-)})$, for all $1 \leq i \leq n-1$, such that:*
    (a) *$u_i^{(i)} = 0$, robot r_i stops.*
    (b) *$u_{i+1}^{(i)} > 0$, robot r_{i+1} reverses direction.*
(3) *only r_n dies.*

(4) $u_{i+1} = \frac{m_{i+1}-m_i}{2m_{i+1}} u_1^0 \prod_{j=2}^{i} \left(\frac{2m_{j-1}}{m_{j-1}+m_j} + \frac{(m_j-m_{j-1})^2}{(m_{j-1}+m_j)2m_j} \right)$.

Proof. Let us assume that robots are deployed on the infinite line $\mathcal{L}$, and let d_i be the distance between robot r_i and robot r_{i+1}. Also, let X_i be the distance that robot r_i traverses until its first collision (with r_{i-1}). We first define d_n, the distance between r_n and r_{n+1}, such that C_n takes place. Property 3 ensures that r_n dies after colliding with r_{n-1}, so we set the C_n collision such that it happens before r_n returns to its home base. Also C_n should not interfere with the order of the previous collisions, so it should take place after the time of the C_{n-1} collision. Therefore, for arbitrary given initial velocity u_{n+1} of robot r_{n+1}:

$$\frac{X_n}{u_n} u_{n+1} - X_n < d_n < \left(\frac{X_n}{u_n} + \frac{X_n}{u_n^1} \right) u_{n+1}, n \geq 2.$$

$$X_{n+1} = \left(\frac{X_n}{u_n} + \frac{X_n}{u_n^1} \right) u_{n+1}, n \geq 2.$$

$$X_1 = u_1 \frac{d_1}{|u_1| + |u_2|}, d_1 > 0.$$

Notice that after C_n takes place it is possible that r_n keeps moving and that r_{n+1} does not reverse direction. So we set the value of u_{n+1} equal to:

$$u_{n+1} = \frac{m_{n+1} - m_n}{2m_{n+1}} u_n^{n-1}, \ n \geq 1.$$

where $u_n^{(n-1)}$ is the velocity that robot r_n acquires at its first collision (the $n-1$ collision of the entire swarm). Because of property 2b for $\mathcal{S}_n$, $u_n^{(n-1)} > 0$, it holds that $m_{n+1} - m_n < 0$. It is easy to verify in equations 1 and 1 that $u_n^{(n)} = 0$ and that $u_{n+1}^{(n)} > 0$ and therefore robot r_{n+1} returns to its home base, thus properties 1-3 hold. Notice that we can set the perimeter of the cycle equal to: $d_a + d_b + \sum_{i=1}^{n} d_i$, where d_a is the distance between the left wall and robot r_1 and d_b the distance between r_{n+1} and the right wall.

Finally, to show how to compute the velocities of the robots (property 4), observe that:

$$u_i = a_{i-1}u_{i-1}^{(i-2)}, \text{ for } i \geq 2, \tag{2}$$

$$u_i^{(i-1)} = b_i u_{i-1}^{(i-2)} + c_i u_i, \text{ for } i \geq 2. \tag{3}$$

where we used the abbreviations:

$$a_i := \frac{m_{i+1} - m_i}{2m_{i+1}}, \quad b_i := \frac{2m_{i-1}}{m_{i-1} + m_i}, \quad c_i := \frac{m_i - m_{i-1}}{m_{i-1} + m_i}.$$

Equation (3) is Equation 1 for the new velocity of r_i after collision $C_{i-1} = (r_{i-1}^{(+)}, r_i^{(-)})$. Recurring over Equation (3) and recalling that $u_1^{(0)}$ is an arbitrary positive velocity we see that

$$u_i^{(i-1)} = u_1^0 \prod_{i=2}^{n} (b_i + c_i a_{i-1})$$

Finally we have

$$u_{i+1} = a_i u_1^{(0)} \prod_{j=2}^{i} (b_j + c_j a_{j-1})$$

■

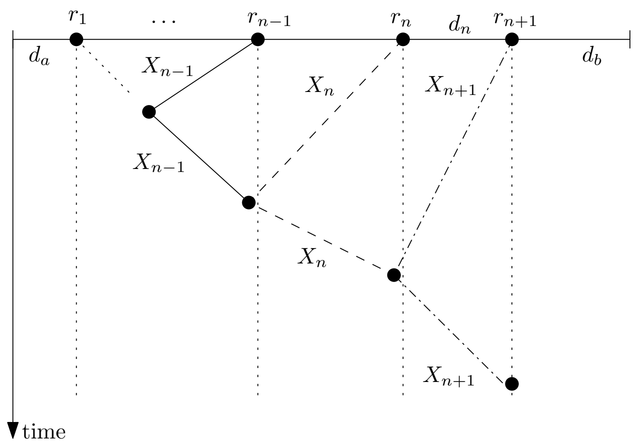

FIGURE 4.3. Illustration of Theorem 7, where exactly $n-1$ robots survive

Notice that Theorem 7 can be extended to the cycle since the construction is independent of the environment.

Corollary 3. *For any $n \geq 2$ there exists a swarm $\mathcal{S}_n$ of bouncing robots of size n in the cycle in which exactly $n-1$ robots survive, such that properties 1 − 4 in Theorem 7 are valid.*

Theorem 3 shows that in the segment all robots die if they have all the same masses and speeds. This is because, the laws of classical mechanics for particles of equal masses and speeds force each robot to have at any time a fraction of the total kinetic energy of the swarm so they can never be static. However, if robots have arbitrary masses and velocities, it is possible that some robot carrying all the kinetic energy of the swarm dies, then all the remaining static robots would survive.

Theorem 8. *Let $\mathcal{S}_n$ be any swarm of bouncing robots of different masses and arbitrary velocities deployed in the segment and let D be the subset of robots of $\mathcal{S}_n$ that do not survive. Therefore, the robots of the resulting swarm after the death of the robots in D must be static, i.e, $KE(S_n \setminus D) = 0$.*

Proof. Let D be the set of robots of the swarm that have died so far and let us assume that $KE(S_n \setminus D) \neq 0$. We can relabel the robots in such a way that the robots at the extremes of the segment are r_1 and r_m for some $m \leq n$. Without loss of generality assume that r_m is static if not, we can consider the equivalent system in which we substract u_b (the velocity of r_m) from all the velocities.

Notice that if robot r_m is at the left of its home base (respectively r_1 at the right of its home base), it is enough to prove that it will get hit by r_{m-1}) (respectively by r_2), Equation 1 guarantees that r_m must move left-to-right direction after a hit by r_{m-1} and hence return to its home base. Corollary 1 guarantees that the hit exists. So we assume that r_n remains still at the right of its home base. Again Corollary 1 guarantees that r_m will get hit by r_{m-1}, however, it is possible that after colliding with the right wall and before it returns to its home base it collides with r_{m-1} so it would not die. Notice that for r_{m-1} not to die, it has to collide with r_{m-2} before it reaches its home base, and so does r_{m-3} and so on. In general, robots $r_1, \ldots, r_{n-1}$

should be at the right of their home bases and r_i must collide r_{i+1} before r_{i+1} reaches its home base, for $1 \leq i \leq n$. It follows that r_1 must die after colliding with r_2. Notice that we can apply this argument as long as $|D| < n$ or $KE(\mathcal{S}_n \setminus D) \neq 0$. In both cases the lemma holds. ∎

In contrast to the case of the segment in which all survivors must be static, in the cycle there are swarms of non-static survivors.

THEOREM 9. *In the cycle there exist a swarm $\mathcal{S}_3$ of size three in which only one robot dies and the kinetic energy of the survivors is different from zero.*

Proof. Let $h_a = 0, h_b = 20, h_c = 16$ be the home bases of robots r_a, r_b and r_c, respectively, and let $m_a = 1, m_b = 3, m_c = 3$ and $u_a = 3, u_b = 1, u_c = -1$ be the values of their respective masses and velocities. The resulting swarm satisfies the properties of the theorem. ∎

The next theorem states that the number of survivors can never be larger than $n-1$ in any swarm of arbitrary masses and speeds deployed in the cycle.

THEOREM 10. *Any swarm $\mathcal{S}_n$ of bouncing robots of arbitrary masses and speeds in the segment or a cycle has some robot that dies.*

Proof. Consider first a segment environment. As the initial kinetic energy of the swarm is positive and it stays the same after any bounce it must stay positive until some robot dies. By Theorem 8, at least one robot has to die. Consider now a cycle environment. Take the the segment representation of the cycle. It is enough to prove that one of the two robots at the end points of the segment dies. Theorem 1 guarantees that if no robot dies, all the robots can not remain colliding among themselves in the interval (h_1, h_n) for an indefinite period of time but that there exists a time $t^\star$ in which no more collisions take place and that some robot would eventually leave the interval $[h_1, h_n]$. Notice that the first robots that can leave the interval $[h_1, h_n]$ can only be r_1 or r_n. Thus, whatever the direction of the expansion of the swarm is, robot r_1 or robot r_n will die. ∎

6. Conclusions

One open problem is to study the survivability of robots when the deadly zones are not just points but rather regions of the environment. It also remains open to investigate the largest number of survivors for robots of equal masses and speeds. Towards this goal, we wrote a program in Java to simulate the survivability of bouncing robots in the cycle. Our program randomly generated positions and initial directions for n robots of unitary masses and speeds and tested their survivability. Our program could only find examples with two survivors for $1 \leq n \leq 10$. It also remains open to figure out the largest number of non-static survivors for swarms of different masses and velocities. We only considered elastic collisions between robots, so we think that it would be interesting to study the case of inelastic collisions and different physical interactions among the robots.

CHAPTER 5

Communication with Bouncing Robots

In this Chapter we investigate communication protocols in a modified version of the model of *bouncing robots*. As before each robot is given an initial direction at which it starts its movement, the interaction among robots is limited to elastically colliding among themselves. Each bouncing robot has no knowledge about its environment and about any other robot. When two of them collide, they instantaneously update their velocities according to the laws of classical mechanics for elastic collisions. They neither have control on their movements nor have any visibility mechanism. However, we allow robots to communicate only when they come into contact with each other.

We study collections of bouncing robots that are able to communicate in a limited way. Each bouncing robot initially possesses a piece of data that is intended to be transferred to other robots. During a collision, robots exchange all the data they have collected until that moment. Our model of communication is *similar* to the ones used for studying the spread of some diseases in the sense that certain information is transmitted as long as an *infected* robot touches a non-infected one.

It was proven in [48] that the number of collisions of elastic particles sliding on an infinite line is finite. Thus, there is a time after which bouncing robots stop colliding among themselves, implying that the spreading of information can not last forever. This raises some fundamental questions which we address in this thesis: For a robot r transmitting some information, what are the robots that get the initial information of r? Are bouncing robots able to perform broadcasting, convergecast, and gossiping? One may relate our information spreading to *infection propagation* - when one robot is infected and infection is transmitted by contact, we may be interested what is the portion of the population infected at time t. The results presented in this Chapter have been submitted to a conference.

1. Preliminaries

Let $\mathcal{S}_n = (\mathcal{H}, \mathcal{V})$ be a collection of n bouncing robots $r_1, \ldots, r_n$, starting positions $\mathcal{H} = (h_1, h_1, \ldots, h_n)$, and non-zero initial velocities $\mathcal{V} = (v_1, v_2, \ldots, v_n)$, respectively. We assume that $h_1 < h_2 < \cdots < h_{n-1} < h_n$ and that robots are deployed on an infinite line,
$glsli = (-\infty, \infty)$. Bouncing robots behave like elastic particles updating their velocities at the times of their collisions according to the laws of classical mechanics [35]. Similarly as in [48, 12, 40], we assume that no more than two robots may collide at the same position and at the same time. We assume that all robots have equal masses such that when two of them collide they simply exchange velocities. We denote by $\nu_i(t)$, the velocity of robot r_i at time t, thus $\nu_i(0) = v_i$.

Each robot r_i initially holds some piece of data (or information) d_i which is intended to be transmitted to other robots. When two robots collide, they exchange *all the data* that each of them has collected up to that moment. So we assume that every robot has enough memory to store all the information it collects. Robots do not overpass each other thus preserving at all times their initial order.

We denote by $\rho(t, i)$ the index of the rightmost robot holding d_i at time t, analogously, $\lambda(t, i)$ denotes the index of the leftmost robot carrying d_i at time t. For every robot r_i we define $M_i = \max\{v_j \in \mathcal{V} |\, j \le i\}$ analogously, $m_i = \min\{v_k \in \mathcal{V} |\, i \le k\}$. Finally, $R_i = |\{v_j < M_i |\, i < j\}|$, and $L_i = |\{v_j > m_i |\, i < j\}|$.

The transmission range of any robot r_i is an interval of robot indices $[a, b] \subseteq [1, n]$ such that for every $j \in [a, b]$, robot r_j receives d_i. A robot r_i *broadcasts* d_i if and only if its transmission range is $[1, n]$. We denote the set of robots of $\mathcal{S}_n$ that perform broadcasting by $\mathcal{B}(\mathcal{S}_n)$. A *convergecast robot* r is a robot that receives every d_i for $1 \le i \le n$. *Gossiping* in $\mathcal{S}_n$ takes place if and only if $\mathcal{B}(\mathcal{S}_n) = \{r_1, \ldots, r_n\}$.

2. Transmission Range of Bouncing Robots

Sevryuk [48] proved that the number of collisions in a system of elastic particles is finite. For any collection of robots $\mathcal{S}_n$ there exists a minimal time moment $t^\star$ such that for any $t > t^\star$ no more collisions take place among the robots of $\mathcal{S}_n$. We call $t^\star$ the *expansion time* of $\mathcal{S}_n$. The following lemma follows immediately.

Lemma 12. *After the time of expansion of any collection of bouncing robots all the robots are sorted by their velocities.*

Since after the time of expansion no more collision can take place, any transmission of information among robots must happen before the system expands. The following lemma is a consequence of Lemma 12.

Lemma 13. *For any collection of bouncing robots $\mathcal{S}_n$, if robot r_i acquires initial velocity v_j for $i \geq j$ ($j > i$), then r_i receives all information d_k for $j \leq k \leq i$ ($i \leq k \leq j$).*

Proof. For r_i to get velocity v_j there was a sequence of collisions among robots $s = r_j, r_{j+1}, \dots, r_{i-1}, r_i$ in which v_j was transferred from r_j to r_i. Notice that together with v_j also d_j was transferred. As v_j is transferred from r_j to r_i, every robot r_k also transmits d_k together with v_j, for all $j \leq k \leq i$. Therefore, at the time r_i received d_j it also received every d_k. Analogously, if $j > i$. ∎

The initial information d_i is spread through the bounces to successive robots left and right to r_i. In particular, at any time t, we may be interested what are the leftmost and the rightmost robots (and their current speeds) which acquired d_i.

Lemma 14. *The speeds of the rightmost robots $r_{\rho(t,i)}$ holding d_i never decrease in time, i.e. $\nu_{\rho(t_1,i)}(t_1) \leq \nu_{\rho(t_2,i)}(t_2)$ for $t_1 \leq t_2$. Moreover, eventually it acquires velocity M_i, i.e, there exists a time t' such that $\nu_{\rho(t',i)}(t') = M_i$.*

Proof. It is enough to look at the velocity of the rightmost robot holding d_i at the times of its collisions. Let us assume that the next collision of $r_{\rho(t,i)}$ is with its neighbor r which has velocity u before such a collision, say at time t''. If $u > \nu_{\rho(t,i)}(t)$, clearly $\rho(t,i) = \rho(t'',i)$ and after the collision $r_{\rho(t'',i)}$ moves with velocity u. In case of $u < \nu_{\rho(t,i)}(t)$ this can only happen if r is to the right of $r_{\rho(t,i)}$. Thus, at the time of collision r gets d_i and starts moving with velocity $\nu_{\rho(t,i)}(t)$, thus $r_{\rho(t'',i)} = r$. Therefore, the first part of the lemma holds. For the second part of the lemma, let us assume that $\nu_{\rho(m,i)}(m) \neq M_i$ for all m, in particular for $m = t^\star$, the time of expansion of the system. Hence, there exists some robot r_j such that $\nu_j(t^\star) = M_i$, where $j < \rho(t^\star, i)$. This contradicts Lemma 12, since $M_i \geq \nu_{\rho(t^\star,i)}(t^\star)$. Fig. 5.1 illustrates this lemma. ■

Analogously for the leftmost robot we have that the following lemma holds

Lemma 15. *The speeds of the leftmost robots $r_{\rho(t,i)}$ holding d_i never increase in time, i.e. $\nu_{\rho(t_1,i)}(t_1) \geq \nu_{\rho(t_2,i)}(t_2)$ for $t_1 \leq t_2$. Moreover, eventually it acquires velocity m_i, i.e, there exists a time t' such that $\nu_{\rho(t',i)}(t') = m_i$.*

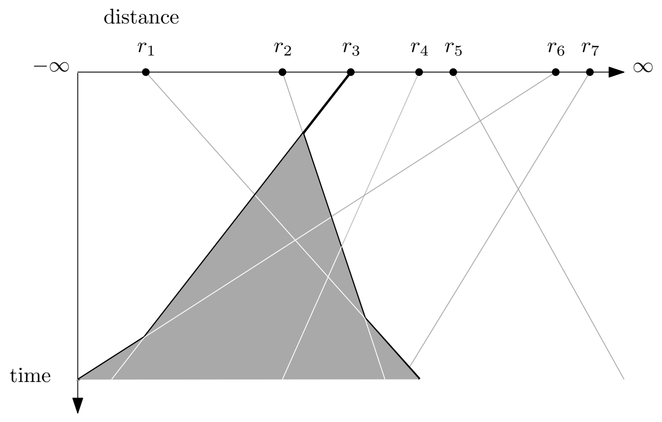

Figure 5.1. The bold polylines depict the spread of d_3 among robots.

Notice that $\rho(t^\star, i)$ and $\lambda(t^\star, i)$ will determine the transmission range of robot r_i since at the time of the expansion of the system no more collisions take place. We denote by $Max_i(t)$ the $\max\{\nu_j(t) |\, j \leq i\}$, i.e., the maximum of the velocities of the robots to the left (including $\nu_i(t)$) of r_i at time t. Analogously, $Min_i(t)$ denotes $\min\{\nu_j(t) |\, j > i\}$.

LEMMA 16. *$r_{\rho(t,j)}$ (respectively $r_{\lambda(t,j)}$) transfers d_j to its neighbor $r_{\rho(t,j)+1}$ (respectively $r_{\lambda(t,j)-1}$) if and only if $Min_{\rho(t,j)}(t) \leq Max_{\rho(t,j)}(t)$ (respectively $Max_{\lambda(t,i)}(t) \geq Min_{\lambda(t,i)}(t)$).*

Proof. Lemma 14 ensures that the speed $\nu_{\rho(t_1,i)}$ never decreases. It is enough to prove that $r_{\rho(t,j)}$ will collide with $r_{\rho(t,j)+1}$ eventually. To do so, it is enough to prove that $r_{\rho(t,j)+1}$ eventually moves at some velocity slow enough to be reached by $r_{\rho(t,j)}$. So we assume that no collision takes place between $r_{\rho(t,j)}$ and $r_{\rho(t,j)+1}$. Lemma 14 ensures that $\nu_{\rho(t,i)}(t_0) = Max_i(t)$ and that $\nu_{\rho(t,i)+1}(t_1) = Min_i(t)$ for some $t_0, t_1 \geq t$. By hypothesis we know that $Max_i(t) \geq Min_i(t)$, therefore $r_{\rho(t,j)}$ transfers d_j to $r_{\rho(t,j)+1}$ eventually. Clearly if d_j is transferred by $r_{\rho(t,j)}$ to $r_{\rho(t,j)+1}$ the lemma holds. Analogously for $r_{\lambda(t,j)}$. ■

Recall that $R_i = |\{v_j < M_i | \, i < j\}|$ and $L_i = |\{v_j > m_i | \, i < j\}|$. The following theorem establishes the transmission range of a robot.

LEMMA 17. *Information d_i is transferred only to robots $r_{i-L_i}, \ldots, r_{i+R_i}$.*

Proof. Notice that $r_i = r_{\rho(0,i)}$ and let us consider first the transmissions of d_i to successive robots to the right of r_i. Lemma 16 and Lemma 14 guarantee that these changes of successive robots happen exactly R_i times. At time t of the R_ith transmission of d_i to some robot to the right of r_i (which corresponds to the R_ith update of the rightmost robot), all robots r_j such that $j > i + R_i$, it holds that $\nu_j(t) \geq M_i$, thus no more transmission of d_i can take place. Moreover $\rho(t,i) = i + R_i$. Analogously, for the transmission of d_i to the left of r_i. ■

Lemma 17, establishes the transmission range of r_i as $[i - L_i, i + R_i]$. The following corollary establishes the necessary and sufficient conditions for a set of communication primitives to take place in a collection of bouncing robots. Notice that they follow immediately from Lemma 17.

COROLLARY 7 (Communication primitives). *For any robot $r_i \in \mathcal{S}_n$:*

(1) *$r_i \in \mathcal{B}(\mathcal{S}_n)$ if and only if $M_i > v_j$ and $m_i < v_k$ for all $j \geq i$ and $k \leq i$;*

(2) *r_i is a convergecast robot if and only if $R_1 \geq i$ and $L_n \geq n - i$ and*

(3) *Gossiping takes place in $\mathcal{S}_n$ if and only if $v_1 > v_j$ and $v_n < v_k$, for all $j > 1$, and $k < n$.*

3. Deciding the Feasibility of Communication

Notice that the communication primitives in a collection of bouncing robots depend on the order of their speeds but not on their initial positions on $\mathcal{L}$. In this section, we show what preprocessing, if any, should be done on the set of velocities of a collection of robots and how to store such information so that we can efficiently decide whether the aforementioned communication primitives happen.

Given any collection of bouncing robots $\mathcal{S}_n$, we store the velocities of all the robots into a table $\mathtt{V}$, such that $\mathtt{V}[\mathtt{i}] = v_i$, and for every robot r_i, we define $M_i' = \max\{v_j > m_i | \, i < j\}$, $m_i' = \min\{v_k \in \mathcal{V} | \, i \leq k\}$. Note that $M_{i+1} \geq M_i$, $m_{i+1} \leq m_i$, $M_{i+1}' \geq M_i'$ and $m_{i+1}' \leq m_i'$. The next lemma follows immediately from the last observation.

Lemma 18. *In $O(n)$ time we can build tables $\mathtt{M}, \mathtt{m}, \mathtt{M}'$, and $\mathtt{m}'$ such that $\mathtt{M}[\mathtt{i}] = M_i, \mathtt{m}[\mathtt{i}] = m_i, \mathtt{M}'[\mathtt{i}] = M_i'$, and $\mathtt{m}'[\mathtt{i}] = m_i'$.*

After constructing these tables in linear time, we can use them in order to decide whether or not any robot can perform broadcasting. The following lemma states so.

Lemma 19. *For any collection $\mathcal{S}_n$ of bouncing robots, there are data structures that allow us to decide whether $r_i \in \mathcal{B}(\mathcal{S}_n)$ in $O(1)$ time.*

Proof. It suffices to check the tables of Lemma 18, more specifically we have to check that $\mathtt{M}[\mathtt{i}] > \mathtt{M}'[\mathtt{i}]$ and $\mathtt{m}[\mathtt{i}] < \mathtt{m}'[\mathtt{i}]$ hold. Theorem 7 guarantees that if this is the case then the transmission range of r_i is $[1, n]$. ■

The following lemma is an immediate consequence of Lemma 19.

Lemma 20. *For any collection of bouncing robots $\mathcal{S}_n$, we can decide in $O(n)$ time whether gossiping takes place.*

The following lemma establishes that the set of robots that are able to perform broadcasting have consecutive indices.

LEMMA 21. *Let a and b be the minimum and maximum indices, respectively, of all the robots in $\mathcal{B}(\mathcal{S}_n)$. Then $r_j \in \mathcal{B}(\mathcal{S}_n)$ for every $a \leq j \leq b$.*

Proof. Notice that $a+R_a \leq j+R_j$ and $j-L_j \leq b-L_b$, therefore the transmission range of r_j is $[1,n]$. ■

The next corollary is an immediate consequence of Lemma 19.

COROLLARY 8. *For any collection of bouncing robots $\mathcal{S}_n$, we can decide in $O(n)$ time whether broadcast is possible. Moreover, in $O(n)$ time we can build a data structure containing the range of indices $[a,b]$ of the robots in $\mathcal{B}(\mathcal{S}_n)$.*

The following lemma establishes an interesting result about convergecast in a collection of bouncing networks.

THEOREM 7. *For any collection of bouncing robots $\mathcal{S}_n$, we can decide in $O(1)$ time whether there is a convergecast robot.*

Proof. If $v_1 > v_n$, at the time of expansion then v_1 and v_n must be held by robots r_b and r_a, respectively such that $b > a$. Thus there exists a robot $r_j, a \leq j \leq b$ that acquired velocities v_1 and v_n. Because of Lemma 13, r_j acquired every $d_i, i \leq j$ at the time of getting v_1; analogously, at the time of getting v_n, r_j got every $d_k, k \geq j$. Therefore r_j is a convergecast robot and it suffices to check that $v_1 > v_n$. ■

4. Time of Transmission

In this section we explore the necessary time for the robots to carry out the communication primitives we have studied so far. It turns out that computing the necessary time for the robots to complete their transmission of information is closely related to a well known geometric problem, namely, computing the upper envelope of an arrangement of lines.

The concept of kth level is related to the concepts of k-set and Davenport-Schinzel sequences [8, 49]. The kth level problem stated as in [8] follows: given n univariate linear functions $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}, f_i : \mathbb{R} \to \mathbb{R}$ and a number $k \in \{1, \ldots, n\}$ construct $G : \mathbb{R} \to \mathbb{R}$, where $G(x) =$ the k-th smallest of the numbers $f_1(x), \ldots, f_n(x)$. The k-level forms an x-monotone polygonal chain.The complexity of the k-level corresponds to the number of vertices of such a polygonal chain.

When $k = 1, n$, the function G corresponds to the lower envelope and the upper envelope of $\mathcal{F}$, respectively. When the functions define lines in the plane it is known that the upper envelope of $\mathcal{F}$ can be optimally computed in $O(n \log n)$ time and in $O(n)$ time if the lines are sorted (this is because of its duality with computing the convex hull of a set of points).

It is easy to see that our diagram of *time* $\times$ *distance* depicting the trajectories of robots gives us an immediate way to relate the trajectories of bouncing robots with the kth level problem. Recall that in this diagram, when a robot moves with velocity v its trajectory corresponds to a line of slope $1/v$. Thus, we can define $L = \{l_1, \ldots, l_n\}$ such that l_i corresponds to the equation of the line passing through $(0, h_i)$ with slope $1/v_i$ (recall h_i stands for the initial position of r_i). Fig. 5.1 depicts the trajectory of r_1 which corresponds to the lower envelope while the trajectory of r_4 corresponds to the upper envelope of L. We denote by $lEnv(L)$ and $uEnv(L)$ the lower (left) and the upper (right) envelopes of L, respectively. Notice that the rightmost copy of d_i at time t is carried by $r_{\rho(t,i)}$. We denote by $traj_i^+(t)$ the trajectory of the rightmost copy of d_i up to time t and by $traj_i^-(t)$ the trajectory of the leftmost copy of d_i up to time t.

Observation 4. *Notice that:*

(1) $uEnv\left(\{l_1, \ldots, l_i\}\right) = traj^+i(t)$
(2) $lEnv\left(\{l_i, \ldots, l_n\}\right) = traj^-i(t)$
(3) r_j *gets* d_i *when* $r_{\rho(i,t)} = r_j$ *for* $j \geq i$ *and some* t
(4) r_j *gets* d_i *when* $r_{\lambda(i,t)} = r_j$ *for* $j \leq i$ *and some* t

LEMMA 22. *For any collection of bouncing robots $\mathcal{S}_n$, we can compute in $O(n \log n)$ time the moment at which d_i is transferred to any robot r_j in the transmission range of r_i.*

Proof. Recall that the number of robots to the right of r_i whose initial velocity is less than M_i is exactly R_i. Thus there are associated to them R_i functions describing lines in the plane *time* $\times$ *distance*. Let $l^{(1)} \ldots, l^{(R_i)}$ be such a set of lines sorted in increasing order by the initial position of their associated robots. Let us assume that $j \in [i+1, i+R_i]$. Because of Observation 4, there is a time t at which $r_{\rho(i,t)} = r_j$. This takes place exactly at the intersection of $uEnv(\{l_1, \ldots, l_i\})$ with line $l^{(j-i)}$. Clearly, the time to compute this intersection is dominated by the required time to compute $uEnv(\{l_1, \ldots, l_i\})$. Analogously, if $j \in [i - L_i, i]$, we define $l^{(1)}, \ldots, l^{(L_i)}$ the L_i lines whose slope is bigger than m_i and sorted in decreasing order by the initial position of their corresponding robots and we compute the intersection of line $l^{(i-j)}$ with $uEnv(\{l_i, \ldots, l_n\})$ ■

Fig. 5.2 illustrates Lemma 22 showing how to compute the time transmission of d_3 to some robot.

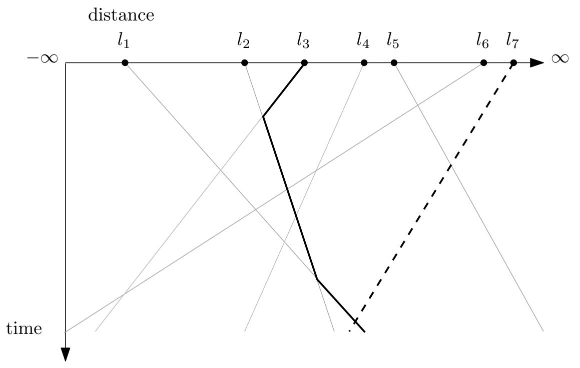

FIGURE 5.2. The fat polyline is the upper envelope of lines l_1, l_2 and l_3. The transmission of d_3 to r_6 takes place at the intersection of $uEnv(\{l_1, l_2, l_3\})$ and l_7.

It turns out that the time when convergecast takes place can be easily computed. The next theorem states so.

Theorem 8. *For any collection of bouncing robots $\mathcal{S}_n$, in $O(1)$ time is possible to determine the earliest time when convergecast is completed.*

Proof. By Lemma 7 in $O(1)$ time we can decide if convergecast takes place in $\mathcal{S}_n$. If this is the case, it means that $\texttt{V[1]} > \texttt{V[n]}$. Assume that $dist(h_1, h_n) = d$ (the distance between the initial positions of robots r_1 and r_n). Then according to Lemma 7, convergecast takes place at time $\frac{d}{|v_1|+|v_n|}$ the intersection point between lines l_1 and l_n. ∎

The next result follows immediately from Lemma 22.

Theorem 9. *For any collection of bouncing robots, there are $O(n \log n)$ algorithms computing the earliest time t at which:*

(1) *the broadcast from robot r_i is completed*
(2) *gossiping in $\mathcal{S}_n$ is carried out*

Proof. Because of Lemma 19 in $O(1)$ time we decide whether $r_i \in \mathcal{B}(\mathcal{S}_n)$ (after $O(n)$ preprocessing). Lemma 22 guarantees that in $O(n \log n)$ time we can compute the times t_1 and t_2 at which r_1 and r_n received d_i, respectively. Thus the completion of the broadcasting of d_i by r_i takes place at time $\max\{t_1, t_2\}$. Notice that gossiping in $\mathcal{S}_n$ is completed at the time that r_1 has received d_n and r_1 has received d_1. To compute these times we use Lemma 22 again. ∎

Computing the trajectory of the kth robot is equivalent to computing the kth level of L, and the number of collisions of r_k corresponds to the complexity of the kth level of L. The next result follows immediately from the results for the k-level problem (see [54, 20] for further details).

Corollary 9. *In any system of bouncing robots $\mathcal{S}_n$, the number of collisions of r_k does not exceed $nk^{\frac{1}{3}}$ and it is at least $n2^{\Omega(\sqrt{\log k})}$. Moreover, the trajectory of r_k can be computed in $O(nk^{\frac{1}{3}} log^c n)$ expected time, for some constant c.*

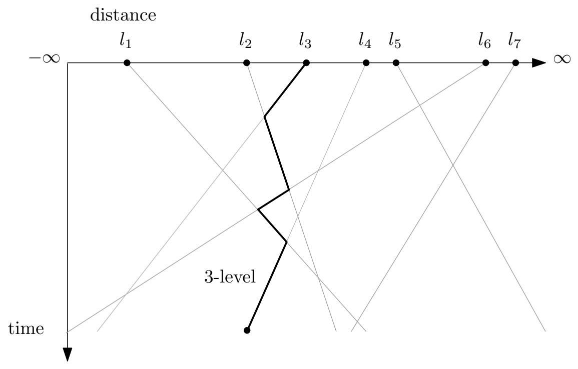

FIGURE 5.3. The 3-level of $\{l_1, \ldots, l_7\}$ corresponds to the trajectory of r_3.

Constructing the trajectories of the first k robots correspond to the construction of all ith levels for $1 \leq i \leq k$. Everett et.al [26] proved that this can be done in $O(n \log n + nk)$ time and that this is optimal.

5. Conclusions

Communication multiplies the capabilities of mobile robots, in this paper we studied a simple model of limited communication for bouncing robots. We proved that robots enabled with such mechanism non-trivial protocols of communication can be carried out by bouncing robots. However the type of communication we assumed allows robots to perform these protocols just once. An open question is to investigate long term communication mechanisms for bouncing robots as well as specific tasks than can be carried out while information is being exchanged.

CHAPTER 6

Conclusions and Future Research

The area of mobile robot computing is an interesting, challenging, and relatively new domain of distributed computing. Although some theoretical research has been done in this area many questions remain open.

In this thesis, we gave a brief description of the most common models that have been proposed to study systems of mobile robots. We also proposed models that assume extremely weak robots, i.e, robots that have very restricted capabilities. In all these models, robots do not have common sense of direction and do not have control over their movements. In addition, they are only equipped with a collision sensor and a clock that allow them to measure the times when they collide with each other. The environments that we considered are one dimensional. We notice that our model resembles some models of gas particles that have been previously studied in the area of classical mechanics.

In these models, we studied the problem of localization: a collection of n anonymous mobile robots are deployed in a continuous cycle of perimeter one. All robots start moving at the same time along the cycle. The task of each robot is to localize the initial position and starting direction of every deployed robot. All this should be done within a finite amount of time. This problem had been previously addressed from a non-deterministic point of view in [**33**].

We addressed the localization problem in Chapter 3, from a deterministic point of view. We assumed that robots have equal speeds. When two robots collide they *bounce back* (maintaining their speed but moving in opposite direction). We proved that all robot configurations in which not all the robots have the same initial direction are feasible and we also provided a localization algorithm working for all feasible configurations. The time complexity of our algorithm depends on the number of robots starting their movement in each direction. If the less frequently used initial direction is given to $k \leq n/2$ robots, the time until completion of the algorithm by the last robot is $\frac{1}{2}\lceil\frac{n}{k}\rceil$, this time is optimal. We considered as well the case when robots are deployed on a segment of length one. In this case, the necessary time to solve the problem is $O(1)$ for all non-symmetric cases.

In Chapter 3, we also considered a different situation in which robots have arbitrary velocity and they are not aware of the velocities of the other robots. We assumed the conservation of momentum and the conservation of energy principles, so robots exchange velocities when they collide. The resulting dynamics of the robots is quite similar to simple gas particle systems. The capabilities of each robot are limited to measuring the times of its collisions, to being aware of its velocity at any time, and to processing the information that it collects. Similarly as in Chapter 2, robots neither have control of their walks nor of their velocities. Their walks depend on their initial positions, velocities, and sequence of collisions while their velocities at any time depend only on their sequence of collisions.

We show that the feasibility of any configuration and the required time for solving the localization problem under such stronger constraints depend only on the collection of velocities of the robots. More specifically, if $v_0, v_1, \ldots, v_{n-1}$ is the collection of velocities of a given robot configuration $\mathcal{S}$, we prove that $\mathcal{S}$ is a feasible robot configuration if and only if $v_i \neq \bar{v}$ for all $0 \leq i \leq n-1$, where $\bar{v} = \frac{v_0+\ldots+v_{n-1}}{n}$. To figure out the initial position of all robots no more than $\frac{2}{min_{0\leq i\leq n-1}|v_i-\bar{v}|}$ time is required.

In Chapter 4, we study the survivability of swarms of bouncing robots that are deployed on environments with deadly points. When a robot reaches such a point it dies. We prove some general properties regarding the dynamics of bouncing robots that are crucial for their survivability and that can be extended to systems of elastic particles. We studied this problem for different one dimensional environments as well as for different models of robots. We show the existence of set of no dying robots for all the environments.

Finally in Chapter 5, we studied a modified version of bouncing robots in which each of them are capable to exchange messages during their times of collision. This model of communication is similar to the spreading of some diseases. We establish sufficient and necessary conditions for bouncing robots to perform several communication protocols like gossiping, broadcasting, and convergecast. We also established a interesting connection with our model of bouncing robots with an old problem in computational geometry, computing the kth level of a set of functions.

Computing the kth level of a set of functions is an extensively studied problem in differential equations and computational geometry and has a close relationship with solving the k-set problem and Davenport-Schinzel sequences [8, 49]. In this Chapter, we show that computing the trajectory of the kth robot moving on an infinite line is equivalent to computing the kth level of an arrangement of lines. The number of collisions that a robot experiences and the time that a robot requires for its information to be spread among other robots is related to bounding the complexity of a level and computing the upper envelope of an arrangement of lines, respectively.

1. Future Research

There remain many open problems in this new area of bouncing robots. Many of the open questions are closely related to challenging problems in physics that have been studied for a long time. For instance, it is unknown the precise number of collisions in a system of bouncing robots of arbitrary masses and speeds living in an infinite line. As far as we are aware there are no examples that matches the upper bound in [48].

Understanding the dynamics of a system of particles of arbitrary masses that slide on a cycle would be crucial for robots to do localization in the cycle. For the case of three robots of arbitrary masses and speeds the dynamics of the corresponding system can be studied using the dynamics of billiards [12]. Yet the resulting dynamics is intricate and little is known about it.

Another open question is to investigate the use of bouncing robots in different environments like planar graphs and two-dimensional environments. Further, it would be interesting to study different synchronization models of bouncing robots. For instance, asynchronous bouncing robots. Assuming non-frictionless environments would also be attractive.

Regarding the survivability of bouncing robots it remains open to find, for the case of equal masses, configurations in which there are more than two survivors.

Different models of communication for bouncing robots may be studied. In particular one that allows them to communicate for longer periods of time.

Bibliography

[1] Curiosity project. http://www.nasa.gov/mission_pages/msl/index.html.

[2] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita. Distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Transactions on Robotics and Automation*, 15(5):818–828, 1999.

[3] D. Angluin, J. Aspnes, Z. Diamadi, M.J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18(4):235–253, 2006.

[4] D. Angluin, J. Aspnes, and D. Eisenstat. Stably computable predicates are semilinear. In *Proceedings of Principles of Distributed Computing*, pages 292–299. ACM, 2006.

[5] H. Attiya, J. Welch, and J. Wiley. *Distributed computing: fundamentals, simulations and advanced topics*, volume 53. Wiley, 2004.

[6] M. Bender, A. Fernández, D. Ron, A. Sahai, and S. Vadhan. The power of a pebble: Exploring and mapping directed graphs. In 30*th Symposium on Theory of Computing (STOC)*, pages 269–278, 1998.

[7] Y.U. Cao, A.S. Fukunaga, and A. Kahng. Cooperative mobile robotics: antecedents and directions. *Autonomous robots*, 4(1):7–27, 1997.

[8] T. Chan. Remarks on k-level algorithms in the plane. *Manuscript, University of Waterloo*, 1999.

[9] R. Cohen and D. Peleg. Convergence properties of the gravitational algorithm in asynchronous robot systems. *SIAM Journal on Computing*, 34(6):1516–1528, 2005.

[10] R. Cohen and D. Peleg. Local spreading algorithms for autonomous robot systems. *Theoretical Computer Science*, 399(1):71–82, 2008.

[11] B. Cooley and P.K. Newton. Random number generation from chaotic impact collisions. *Regular and Chaotic Dynamics*, 9(3):199–212, 2004.

[12] B. Cooley and P.K. Newton. Iterated impact dynamics of n-beads on a ring. *Society for Industrial and Applied Mathematics Review*, 47(2):273–300, 2005.

[13] J. Czyzowicz, S. Dobrev, Kranakis E., and Krizanc D. The power of tokens: Rendezvous and symmetry detection for two mobile agents in a ring. In 34*th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, pages 234–246, 2008.

[14] J. Czyzowicz, S. Dobrev, E. Kranakis, and E. Pacheco. Survivability of swarms of bouncing robots. In *The* 11*th Latin American Theoretical INformatics Symposium*, pages 622–633. Springer, 2014.

[15] J. Czyzowicz, L. Gąsieniec, A. Kosowski, and E. Kranakis. Boundary patrolling by mobile agents with distinct maximal speeds. *Proceedings of European Symposia on Algorithms (ESA)*, pages 701–712, 2011.

[16] J. Czyzowicz, L. Gasieniec, A. Kosowski, E. Kranakis, O. Morales-Ponce, and E. Pacheco. Position discovery for a system of bouncing robots. In *The* 26*-th International Symposium on Distributed Computing (DISC)*, pages 341–355, 2012.

[17] J. Czyzowicz, L. Gąsieniec, and A. Pelc. Gathering few fat mobile robots in the plane. *Principles of Distributed Systems*, pages 350–364, 2006.

[18] J. Czyzowicz, E. Kranakis, and E. Pacheco. Localization for a system of bouncing robots. In 40*th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 7966, 2013.

[19] S. Das, P. Flocchini, N. Santoro, and M. Yamashita. On the computational power of oblivious robots: Forming a series of geometric patterns. In *Proceedings of Symposium on Princples of Distributed Computing (PODC)*, pages 267–276. ACM, 2010.

[20] T. Dey. Improved bounds for planar k-sets and related problems. *Discrete & Computational Geometry*, 19(3):373–382, 1998.

[21] S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro. Mobile search for a black hole in an anonymous ring. In *Distributed Computing*, pages 166–179. Springer, 2001.

[22] G. Dudek and M. Jenkin. *Computational principles of mobile robotics*. Cambridge university press, 2010.

[23] G. Dudek, K. Romanik, and S. Whitesides. Localizing a robot with minimum travel. *SIAM Journal on Computing*, 27(2):583–604, 1998.

[24] Y. Elor and A.M. Bruckstein. Multi-agent deployment and patrolling on a ring graph. Technical report, Israel Institute of Technology, 2009.

[25] L. Erickson, J. Knuth, J. O'Kane, and S. LaValle. Probabilistic localization with a blind robot. *IEEE International Conference on Robotics and Automation, ICRA 2008.*, pages 1821–1827, 2008.

[26] H. Everett, J. Robert, and M. Van Kreveld. An optimal algorithm for the ($\leq k$)-levels, with applications to separation and transversal problems. In *Proceedings of the 9th annual symposium on Computational geometry*, pages 38–46. ACM, 1993.

[27] P. Flocchini, G. Prencipe, and N. Santoro. Self-deployment algorithms for mobile sensors on a ring. *Algorithmic Aspects of Wireless Sensor Networks*, pages 59–70, 2006.

[28] P. Flocchini, G. Prencipe, and N. Santoro. Computing by mobile robotic sensors. *Theoretical Aspects of Distributed Computing in Sensor Networks*, pages 655–693, 2011.

[29] P. Flocchini, G. Prencipe, and N. Santoro. Distributed computing by oblivious mobile robots. *Synthesis Lectures on Distributed Computing Theory*, 3(2):1–185, 2012.

[30] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Hard tasks for weak robots: The role of common knowledge in pattern formation by autonomous mobile robots. *Algorithms and Computation*, pages 93–102, 1999.

[31] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Gathering of asynchronous robots with limited visibility. *Theoretical Computer Science*, 337(1):147–168, 2005.

[32] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Arbitrary pattern formation by asynchronous, anonymous, oblivious robots. *Theoretical Computer Science*, 407(1):412–447, 2008.

[33] T. Friedetzky, L. Gasieniec, T. Gorry, and Russell M. Observe and remain silent (communication-less agent location discovery). In *The 37th International Symposium On Mathematical Foundations of Computer Science (MFCS)*, pages 407–418, 2012.

[34] V. Gervasi and G Prencipe. Flocking by a set of autonomous mobile robots. Technical report, University of Pisa, 2001.

[35] R.D. Gregory. *Classical mechanics*, volume 1. Cambridge University Press, 2006.

[36] D.W. Jepsen. Dynamics of a simple many-body system of hard rods. *Journal of Mathematical Physics*, 6:405, 1965.

[37] D. Kingston, R.W. Beard, and R.S. Holt. Decentralized perimeter surveillance using a team of uavs. *IEEE Transactions on Robotics*, 24(6):1394–1404, 2008.

[38] E. Kranakis, D. Krizanc, and E. Markou. The mobile agent rendezvous problem in the ring. *Synthesis Lectures on Distributed Computing Theory*, 1(1):1–122, 2010.

[39] D.B. Lange and M. Oshima. Seven good reasons for mobile agents. *Communications of the ACM*, 42(3):88–89, 1999.

[40] T. Murphy. Dynamics of hard rods in one dimension. *Journal of statistical physics*, 74(3):889–901, 1994.

[41] TJ Murphy and EGD Cohen. Maximum number of collisions among identical hard spheres. *Journal of statistical physics*, 71(5-6):1063–1080, 1993.

[42] L.E. Parker. Heterogeneous multi-robot cooperation. Technical report, DTIC Document, 1994.

[43] L.E. Parker. Alliance: An architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240, 1998.

[44] G. Prencipe. The effect of synchronicity on the behavior of autonomous mobile robots. *Theory of Computing Systems*, 38(5):539–558, 2005.

[45] K. Romanik and S. Schuierer. Optimal robot localization in trees. *Proceedings of the twelfth annual symposium on Computational geometry*, pages 264–273, 1996.

[46] M. Rosenfeld. Some of my favorite "lesser known" problems. *Ars Mathematica Contemporanea*, 1(2):137–143, 2008.

[47] N. Roy and G. Dudek. Collaborative robot exploration and rendezvous: Algorithms, performance bounds and observations. *Autonomous Robots*, 11(2):117–136, 2001.

[48] M.B. Sevryuk. Estimate of the number of collisions of n elastic particles on a line. *Theoretical and Mathematical Physics*, 96(1):818–826, 1993.

[49] M. Sharir and P. Agarwal. *Davenport-Schinzel sequences and their geometric applications*. Cambridge university press, 1995.

[50] K. Sugihara and I. Suzuki. Distributed algorithms for formation of geometric patterns with many mobile robots. *Journal of Robotic Systems*, 13(3):127–139, 1996.

[51] S. Susca and F. Bullo. Synchronization of beads on a ring. In *46th IEEE Conference on Decision and Control*, pages 4845–4850, 2007.

[52] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing*, 28(4):1347–1363, 1999.

[53] L. Tonks. The complete equation of state of one, two and three-dimensional gases of hard elastic spheres. *Physical Review*, 50(10):955, 1936.

[54] G. Tóth. Point sets with many k-sets. In *Proceedings of the sixteenth annual symposium on Computational geometry*, pages 37–42. ACM, 2000.

[55] H. Wang and Y. Guo. Synchronization on a segment without localization: algorithm and applications. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 3441–3446, 2009.

[56] Gerhard Weiss. *Multiagent Systems, a Modern Approach to Distributed Artificial Intelligence*. MIT press, 2000.

[57] J.J. Wylie, R. Yang, and Q. Zhang. Periodic orbits of inelastic particles on a ring. *Physical Review E*, 86(2):026601, 2012.