Local Protocols for Contracting and Expanding Robot Formation Problems

Dissertation

In partial fulfillment of the requirements for the degree of

Doctor rerum naturalium (Dr. rer. nat.)

at the Faculty of Computer Science,
Electrical Engineering and Mathematics
at Paderborn University

submitted by

JANNIK CASTENOW

Reviewers

- Prof. Dr. Friedhelm Meyer auf der Heide,
  Paderborn University

- Prof. Dr. Christian Scheideler,
  Paderborn University

Zusammenfassung

Inspiriert von der Vision von Roboterschwärmen, die gemeinsam Gebiete wie die Oberfläche entfernter Planeten erkunden (z.B. *Marsbees* [75]), untersuchen wir die theoretischen Grundlagen von mobilen Roboterschwärmen. Die Roboter werden als Punkte in einem d-dimensionalen euklidischen Raum modelliert. Jeder Roboter hat nur sehr begrenzte Fähigkeiten und *lokale* Informationen über seine Umgebung (er kann andere Roboter nur bis zu seiner *Sichtweite* wahrnehmen). In der Regel sind die Roboter *orientierungslos*, d. h. sie haben kein gemeinsames Koordinaten- oder Positionierungssystem. Aufgrund der begrenzten Fähigkeiten der Roboter werden bereits grundlegende Aufgaben für die Roboter zu einer algorithmischen Herausforderung. In dieser Arbeit untersuchen wir vier Formationsprobleme, bei denen sich die Roboter so bewegen sollen, dass ihre Positionen ein vorgegebenes Muster bilden. Unser Fokus liegt auf der folgenden Frage: Wie schnell können die Roboter ein bestimmtes Formationsproblem mit gegebenen Roboterfähigkeiten lösen?

Die Formationsprobleme können in zwei Klassen eingeteilt werden: *Kontrahierende* und *expandierende* Formationsprobleme. Das Ziel von kontrahierenden Formationsproblemen ist es, die Roboter näher zusammenzubringen oder eine Zielstruktur zu minimieren. Auf der anderen Seite gibt es expandierende Formationsprobleme, bei denen sich die Roboter ausbreiten müssen, um eine möglichst große Fläche abzudecken oder eine Zielstruktur zu maximieren. Kontrahierende Formationsprobleme sind in der Literatur gut bekannt, aber über expandierende Formationsprobleme ist, insbesondere bei Robotern mit nur lokalen Informationen, noch viel Forschungsbedarf.

Konkret untersuchen wir die kontrahierenden Formationsprobleme GATHERING und CHAIN-FORMATION. Beim GATHERING müssen sich die Roboter zu einem einzigen, nicht vordefinierten Punkt bewegen. Unsere Forschung identifiziert strukturelle Eigenschaften von Protokollen zur effizienten Lösung des GATHERING-Problems in verschiedenen Zeitmodellen. Insbesondere entwickeln wir die ersten lokalen GATHERING-Protokolle für Roboter in einem euklidischen Raum der Dimension drei oder höher. Das CHAIN-FORMATION-Problem untersucht eine Kette von Robotern. Genauer gesagt, sind die Roboter in einer Kettentopologie (jeder innere Roboter hat zwei Nachbarn) zwischen zwei stationären äußeren Robotern verbunden. Das Ziel der Roboter ist es, die Länge der Kette zu minimieren, d. h. sich auf einer geraden Linie zwischen den äußeren Robotern zu positionieren. Wir zeigen, wie die Fähigkeit, Informationen konstanter Größe zu kommunizieren (die Roboter haben *Lichter*), dazu beiträgt, schnelle $(1+\varepsilon)$-Approximationsprotokolle für das CHAIN-FORMATION-Problem zu entwickeln.

Außerdem führen wir die expandierenden Formationsprobleme MAX-CHAIN-FORMATION und MAX-LINE-FORMATION neu ein. Beide zielen darauf ab, die Roboter in einer geraden Linie mit maximaler Länge (in Bezug auf die Sichtweite) anzuordnen. Das Problem MAX-CHAIN-FORMATION untersucht eine Kette von Robotern, bei der die äußeren Roboter die Fähigkeit haben, sich zu bewegen. Beim MAX-LINE-FORMATION-Problem haben die Roboter keine vordefinierte

Kettenstruktur, können aber alle anderen Roboter in unmittelbarer Nähe sehen. Wir stellen mehrere Protokolle vor, die von unterschiedlichen Roboterfähigkeiten ausgehen, um beide Formationsprobleme zu lösen. Interessanterweise zeigen wir, dass viele Ideen von Protokollen für kontrahierende Formationsprobleme wiederverwendet werden können. Allerdings zeigen unsere Protokolle auch ein komplexeres Konvergenzverhalten im Vergleich zu kontrahierenden Formationsprotokollen.

Abstract

Inspired by the vision of robot swarms collectively exploring hazardous areas such as the surface of distant planets (e.g., *Marsbees* [75]), we study the theoretical foundations of mobile robot swarms. The robots are modeled as points in a d-dimensional Euclidean space. Each robot on its own has very limited capabilities and only *local* information about its environment (it can only observe other robots up to its *viewing range*). Mostly, the robots are *disoriented*, i.e., they do not agree on any common coordinate or positioning system. Due to the limited capabilities of the robots, even basic tasks for the robots become algorithmically challenging. In this thesis, we study four robot *formation problems* in which the robots are to move such that their positions fulfill a predefined pattern. Our focus lies on the following question: How fast can the robots solve a particular formation problem under given robot capabilities?

The formation problems can be categorized into two classes: *contracting* and *expanding* formation problems. The goal of contracting formation problems is to move the robots closer together or to minimize a target structure. On the opposite side, we have expanding formation problems, where the robots have to spread out to cover as much area as possible or to maximize a target structure. Contracting formation problems are well-established in the literature, but much less is known about expanding formation problems, especially for robots with only local information.

Concretely, we study the two contracting formation problems GATHERING and CHAIN-FORMATION. GATHERING demands to move the robots to a single, not predefined point. Our research identifies structural properties of protocols to solve GATHERING efficiently in different time models. Especially, we develop the first local gathering protocols for robots in a Euclidean space of dimension three or higher. The CHAIN-FORMATION problem studies a chain of robots. Precisely, the robots are connected in a chain topology (each inner robot has two neighbors) between two stationary outer robots. The goal of the robots is to reduce the length of the chain, i.e., to position themselves on a straight line between the outer robots. We show how the ability to communicate constant-sized information (the robots have *lights*) helps to derive fast $(1+\varepsilon)$-approximation protocols for the CHAIN-FORMATION problem.

Furthermore, we newly introduce the expanding formation problems MAX-CHAIN-FORMATION and MAX-LINE-FORMATION. Both aim to arrange the robots in a straight line of maximum length (with respect to the viewing range). MAX-CHAIN-FORMATION studies a chain of robots in which the outer robots can move. In the MAX-LINE-FORMATION problem, the robots do not have a predefined chain structure but can observe all other robots in close vicinity. We introduce several protocols assuming different robot capabilities to solve both formation problems. Interestingly, we show that many ideas of protocols for contracting formation problems can be reused. However, our protocols also exhibit a more complex convergence behavior compared to contracting formation protocols.

Preface

As I write these words, a long, very exciting journey is coming to an end. Many people have made this journey possible in the first place or have accompanied me on it. Following, I would like to thank these people. First of all, I would like to thank my supervisor Friedhelm Meyer auf der Heide, who gave me the opportunity to do research at his chair. Even in difficult times, I felt great confidence that my approaches were the right ones and would ultimately lead to success. This trust in combination with the freedom you gave me to find my own research profile was an important key to the completion of this thesis.

Next, I would like to thank Till Knollmann. We have been friends since the beginning of our studies and have also shared the time of the Ph.D. as friends, colleagues, and co-authors. You were a great anchor and support during all this time. Dear Till, I do not know if I would have gotten this far without you.

Apart from Till and Friedhelm, I had several other fantastic co-authors: Michael Braun, Thorsten Götte, Jonas Harbig, Peter Kling and Daniel Jung. I would especially like to thank Peter and Daniel, with whom I always had productive discussions despite the physical distance. Thank you for your time, your commitment, and your constant belief in our joint research. I also thank Jonas, who participated in my research as an undergraduate and eventually joined the research group as a Ph.D. student.

Besides the co-authors, I would like to thank my former office colleagues Johannes and Malex. The time with you in the office, especially the rounds together in the game World of Padman, will remain unforgotten. Representing all other colleagues, I would like to thank Petra, Heinz-Georg, and Matthias, who always helped me in every matter.

Furthermore, I would like to thank my family. Especially to Sarah, my wife, I owe a debt of gratitude that I can hardly put into words. Thank you for putting up with all my moods when once again an idea did not work out or a proof got broken. Your constant belief in me, your motivational skills, but also your ability to distract me from work at the right moments has helped me to go this way. *I love you*. Last but not least, you gifted me with our daughter Ida, who is already almost 3 years old at the time of these words. Dear Ida, you have been a big part of this journey. There was nothing better than reading a book to you or playing with you when I was supposed to be writing this dissertation. I am very thankful that you are with us.

Jannik Castenow
March, 2023

Contents

1. Introduction

> The future is in the hands of those who explore.
>
> Jacques-Yves Cousteau,
> Oceanographer

Since ancient times, humans have aimed to develop mechanical machines to automate various tasks. If the machine is programmable, operates to some degree autonomously and senses and manipulates its environment, such a machine is usually called a *robot* (although there is no common definition). Robots occur in various application fields, ranging from industrial robots for manufacturing, over warehouse robots that automate logistics, to medical robots assisting in surgeries. When the robots can move, as the warehouse robots typically do, they are called *mobile* robots. A further important application of mobile robots is the exploration of areas that are difficult or, with current technical developments, even impossible for humans to reach. Examples of such areas are the deep sea or distant planets since humans can reach them either only with considerable effort or not at all (yet). In 2021, for instance, NASA's Perseverance Rover reached Mars [73]. Since then, the rover autonomously explores the Martian surface, collects scientific data and sends it back to earth. Researchers evaluate the data and, among other research questions, hope to find evidence for earlier life on Mars. Similar robots exist to explore the deep sea. In contrast to Mars, even the deepest points of earth's seabed – called *Challenger Deep* – can be reached by humans with specialized deep-sea submarines. Nevertheless, this requires extreme effort and is still very dangerous. To overcome these drawbacks, for instance, the Nereus robotic vehicle has been developed [17]. After reaching the ocean floor – Nereus can reach Challenger Deep – Nereus autonomously explores the ocean ground. Despite their advantage that no humans are involved in the exploration, both the Perseverance Rover and Nereus are highly optimized and very costly special-purpose robotic vehicles. The estimated costs for the Perseverance Rover are 2.7 billion US dollars and the development took several years [56]. Moreover, the maintenance periods for those vehicles are very long and single technical errors can have a severe impact. Nereus, for instance, imploded in 2014 on the ocean ground [4].

As a consequence, there is a natural desire for a cheaper, scalable and robust solution for exploration tasks. Scalability is important to be able to explore larger areas and to speed up the exploration. Both the deep sea and the Martian surface are immense such that a single robot would take more than a human lifetime to explore a reasonable portion of the entire surface. As the Nereus example sophisticates, robustness is also important since the exploration area might be hazardous such that a single incident can destroy a robot. To ensure robustness and scalability,

current research projects, including NASA's *Marsbees* [75] or the *CORATAM* project for (deep) sea exploration [57], aim to distribute the exploration task over a large amount of cheap(er) and tiny robots that collectively explore the environment. Such an approach is, among others, inspired by swarms of fish, birds, bees or ants. While each animal on its own has only very limited capabilities, the swarm collectively solves complex tasks. Ants, for instance, can build bridges with their bodies to enable other ants to shortcut certain distances or even reach places that could not be reached before. The most interesting part about ants is that they coordinate themselves without any leader. They operate autonomously and each ant decides based on its local information what seems to be best for the entire swarm [101]. Based on these observations, the vision is to replace single, costly special-purpose robots with a *swarm* of lightweight and cheap robots that solve complex tasks collectively.

However, to benefit from the previously described advantages, lots of challenges have to be solved. The first challenge is the design of the robots including the choice of appropriate sensors and hardware. To keep the cost of each robot low, it is necessary to determine the essential hardware to solve the task the robot is designed for. Furthermore, it is interesting to determine tradeoffs: e.g., larger memory is more expensive but potentially leads to a much faster exploration time since the robots can locally determine which areas they have already visited. The second challenge is the design of correct and (time-)efficient protocols (algorithms). The main challenge here stems from the environment where the robots live and the local information. Consider for instance the seemingly simple tasks of recollecting all robots after finishing the exploration – we will later denote this task as the GATHERING problem. The task is easy to solve as long as the robots have a global coordinate system (e.g., GPS) and can communicate over larger distances (e.g., via the internet). With GPS or global communication, the robots could simply agree on a common position where to meet and every robot moves there. In environments such as the deep sea or the Martian surface, however, no GPS or large-scale communication infrastructure is available. Instead, each robot can only sense its local vicinity and must base its decisions upon this small piece of information. Therefore, seemingly simple tasks turn out to be challenging or, without appropriate sensory capabilities, even impossible to solve.

Since already simple tasks become challenging, we focus on the basic algorithmic building blocks of mobile robot swarms in this thesis. Concretely, we study *robot formation problems* (also denoted as PATTERN FORMATION [111]), where the goal is to arrange the robots in a certain formation. One example is the GATHERING problem, where the robots have to gather at the same location – an important subroutine to recollect spread-out robots. Other examples involve the MAX-LINE-FORMATION problem, where the robots have to bridge a large area. More precisely, we look at robot formation problems for swarms of mobile robots from the perspective of *distributed computing*. The research area of distributed computing studies the theoretical foundations of distributed systems. In this research area, we abstract from the technical details of the robots and work with common models, where robots are modeled as points in d-dimensional Euclidean spaces that move around based on the protocols we design. The previously raised question about the required hardware of a robot translates into the question of how powerful the robot model has to be. The typical intriguing research question is: Which capabilities do the robots need (at least) to solve a formation problem? As mentioned earlier, GATHERING is trivial if robots can communicate over large distances or have access to a common positioning system. But how about robots that can only observe other robots in their close vicinity? The second main research question we study in this thesis is: How fast can robots solve a certain formation problem and how does the robot model influence the runtime? For instance, how fast can *oblivious* robots (no persistent memory) solve a formation task, and can the robots solve the task faster if they have access to a small amount of persistent memory? In this thesis, we mostly focus on the second research question, i.e., our focus lies on the interplay between the robot model and the runtime efficiency of robot formation protocols. Nevertheless, we will also see impossibility results stating that some formation problems are impossible to solve without certain robot capabilities. In the following

Section 1.1, we briefly summarize the robot models and introduce the specific formation problems. Afterward, we summarize the results of this thesis in Section 1.2.

1.1 Scope

The broad topic of this thesis is to design *local* protocols for different types of robot formation problems, to prove their correctness and to analyze their runtime. All protocols are designed for suitable theoretical models that abstract from the physical aspects of the robots and allow us to derive formal correctness proofs and analyze the runtime complexity. In the following, we briefly summarize the robot model and introduce the formation problems studied in this thesis.

Model Summary. Here, we focus on the most important aspects of the robot models. A detailed description of the models is contained in Chapter 2. In all models, the robots are point-shaped and live in a d-dimensional Euclidean space. Each robot has its own local coordinate system that is self-centered (the robot is located at the origin). Mostly, we assume that the robots are *disoriented*, i.e., the axes of their local coordinate systems can be arbitrarily inverted and rotated such that the robots do not agree on any directions. The disorientation is even *variable*, meaning that the axes and the orientation of the coordinate system of a fixed robot might change from time to time. In one exception (Chapter 8), we consider robots with *one-axis agreement*. Those robots agree on the orientation and direction of one coordinate axis. The other axes are commonly aligned; however, the directions might differ from robot to robot. Moreover, the robots have *limited visibility*. Precisely, we consider a *connectivity range* of 1 and assume that the initial configuration is connected concerning the connectivity range. In other words, we assume that the initial Unit Ball Graph (robots are vertices and two robots share an edge if their distance is at most 1) is connected. Additionally, the robots have a *viewing range* of $\mathcal{V}$. In most cases (except for Chapter 8) we consider a *circular* viewing range, i.e., robots can observe all other robots at a distance of at most $\mathcal{V}$. Otherwise, we consider a *square* viewing range of $\mathcal{V}$, i.e., robots can observe other robots within an axis-aligned square of side length $2 \cdot \mathcal{V}$. In many cases, the viewing range and the connectivity range are identical. However, in some cases (e.g., Chapters 5 and 6), it is beneficial to consider a viewing range that is larger than the connectivity range.

Moreover, the robots are *autonomous* (there is no central control guiding the movements of the robots), *identical* (all robots have the same appearance) and *anonymous* (there are no internal identifiers). Furthermore, the robots are *homogeneous* and *deterministic*, i.e., all robots execute the same deterministic protocol. In case the robots are also *oblivious* (no persistent memory) and *silent* (no communication), this is commonly known as the $\mathcal{OBLOT}$ model [66]. The $\mathcal{LUMI}$ model [53] replaces the properties of being oblivious and silent by equipping the robots with a light that can, at each point in time, have one out of a constant number of colors. On the one hand, the light is persistent (unless intentionally switched off) such that the robots obtain a constant-sized memory. On the other hand, the light can be observed by visible robots and thus, the robots gain the ability to communicate information of constant size to their close neighbors.

The robots operate in LCM cycles (rounds). Each cycle consists of three operations: Look, Compute and Move. First, robots take a snapshot of their environment (Look), compute a target point based on the snapshot (Compute) and finally move there (Move). The cycles might be fully synchronous ($\mathcal{F}$SYNC), semi-synchronous ($\mathcal{S}$SYNC: the cycles are synchronous but only a subset of all robots might be active) or completely asynchronous ($\mathcal{A}$SYNC). Time is measured in *epochs*: an epoch is the smallest time period such that each robot completes at least one LCM cycle. Under the $\mathcal{F}$SYNC scheduler, an epoch is equal to a round. Additionally, time can be continuous, i.e., the movement of robots is defined for each *real* point in time by a velocity vector of length at most 1 (the *speed* of each robot is at most 1). We refer to this as the *continuous time model*.

Formation Problems. We study four different formation problems for robots with limited visibility. The formation problems can be categorized into two classes. The first class of problems can be described as *contracting* formation problems. Here, the goal is either to move robots closer

together or to minimize a target structure. On the opposite side, there are *expanding* formation problems, where the goal is to spread the robots out to cover as much area as possible. Regarding these two classes of formation problems, the goal of this thesis is twofold. Our studies of contracting formation problems add new, improved and generalized results to the existing literature. While the area of contracting formation problems of robots with limited visibility is well-established in the literature, there is not much known about expanding formation problems. Hence, our goal regarding expanding formation problems is to shed some light on what kinds of formation problems robots with limited visibility can solve.

More concretely, we study the contracting formation problems GATHERING and CHAIN-FORMATION as well as the expanding formation problems MAX-CHAIN-FORMATION and MAX-LINE-FORMATION. GATHERING demands to move n robots $r_0, \ldots, r_{n-1}$ to a single, not predefined point. Either no fixed neighborhood is given and robots can observe all other robots up to their viewing range (we refer to this as the *standard connectivity model*) or the robots have fixed neighborhoods that form a *closed chain*. GATHERING in the standard connectivity model is depicted in Figure 1.1.

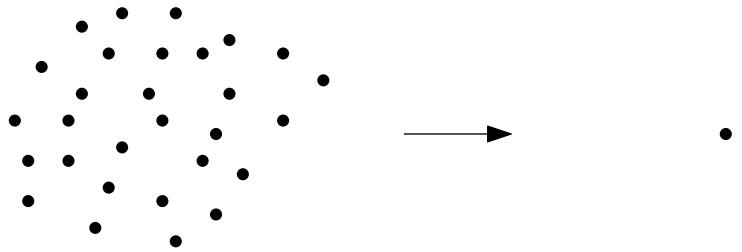

Figure 1.1: A visualization of the GATHERING problem in the standard connectivity model.

In a closed chain, the robot r_i has the fixed neighbors r_{i-1} and r_{i+1} (all operations on indices are modulo n: r_0 and r_{n-1} are each other's neighbors). Still, the robots have limited visibility requiring the robots to ensure that the distance between themselves and their neighbors does not exceed the connectivity range. The GATHERING problem of a closed chain of robots is visualized in Figure 1.2.

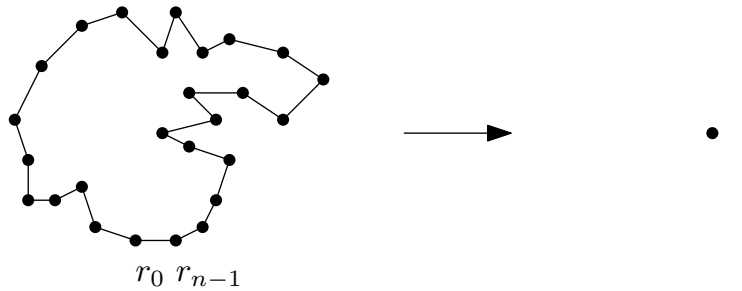

Figure 1.2: A visualization of the GATHERING problem of a closed chain of robots.

The CHAIN-FORMATION problem considers an *open chain* of robots. An open chain has inner robots $r_1, \ldots, r_{n-2}$ and two outer robots r_0 and r_{n-1}. Similar to the closed chain, each *inner* robot r_i has neighbors r_{i-1} and r_{i+1}. The outer robots r_0 and r_{n-1} have only a single neighbor: r_1 is the neighbor of r_0 and r_{n-2} is r_{n-1}'s neighbor. Both outer robots are stationary, i.e., both r_0 and r_{n-1} have fixed positions throughout the entire execution of a protocol. All other robots aim to reach the straight line between r_0 and r_{n-1} while maintaining connectivity to their neighbors. See Figure 1.3 for a visualization of the CHAIN-FORMATION problem.

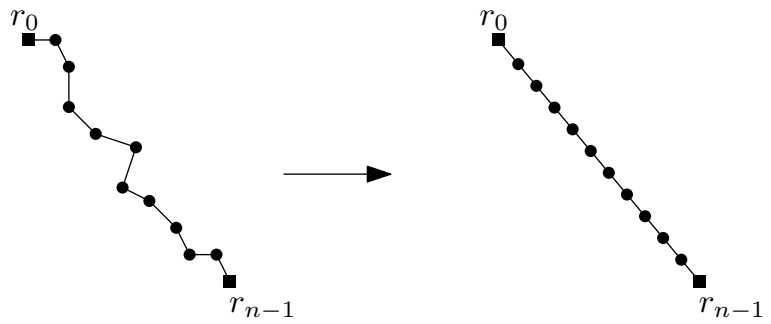

Figure 1.3: A visualization of the CHAIN-FORMATION problem.

MAX-CHAIN-FORMATION is a modification of the CHAIN-FORMATION problem. The difference is that the outer robots r_0 and r_{n-1} do not remain stationary. Instead, the goal is to arrange the chain of robots on a straight line of length $n-1$. The problem is depicted in Figure 1.4. Since the viewing range is assumed to be 1, this is the maximum line length that can be achieved. Due to the chain structure, the ordering along the final line is fixed: r_0 and r_{n-1} are endpoints of the line. The position and orientation of the line are, however, not predefined. Moreover, since r_0 and r_{n-1} can only observe the position of their chain neighbors, they do not know a fixed direction to move along to stretch the length of the chain. The movement of all robots must ensure that r_0 and r_{n-1} eventually move in opposite directions although they cannot observe each other.

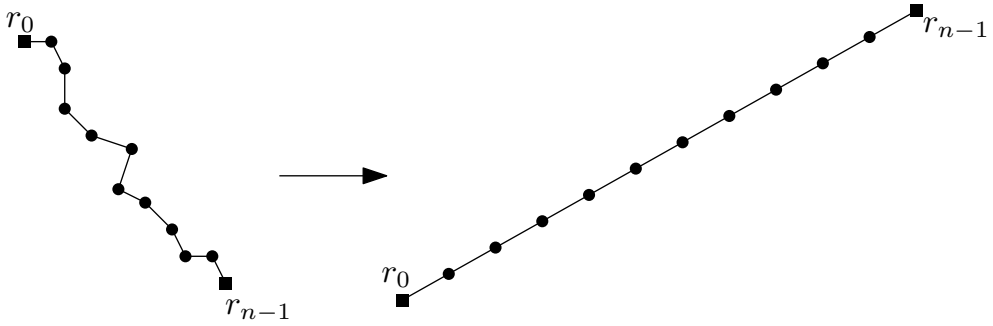

Figure 1.4: A visualization of the MAX-CHAIN-FORMATION problem.

The MAX-LINE-FORMATION problem has the same goal as the MAX-CHAIN-FORMATION problem: to arrange n robots on a straight line of length $n-1$. However, the robots are not connected in a chain but in the standard connectivity model (see Figure 1.5 for a visualization). Henceforth, there are no dedicated robots that must form the end of the line. Rather, the robots have to find their roles in the final line themselves. Additionally, the position and orientation of the line can be chosen by the robots.

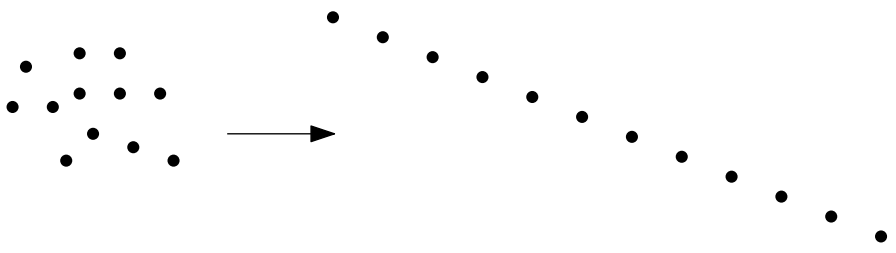

Figure 1.5: A visualization of the MAX-LINE-FORMATION problem.

1.2 Outline of the Thesis and Main Results

The thesis starts with three preliminary chapters (Chapters 1 to 3). Chapter 2 contains a detailed description of the models studied in this thesis and introduces common notation. Afterward, related work is discussed in Chapter 3. The main contents of the thesis are split into two parts. Part I studies the contracting formation problems GATHERING and CHAIN-FORMATION and consists of Chapters 4 to 6. Part II introduces the expanding formation problems MAX-CHAIN-FORMATION and MAX-LINE-FORMATION in Chapters 7 and 8. In what follows, we summarize the main results of each chapter.

GATHERING in the $\mathcal{OBLOT}$ Model: In Chapter 4, we consider the $\mathcal{OBLOT}$ model and study the GATHERING problem of disoriented robots with limited visibility in two different time models: fully synchronous rounds (the $\mathcal{F}$SYNC scheduler) and the continuous time model. For both time models, a very general approach is used: we study entire classes of protocols that can be applied to robots that live in a Euclidean space $\mathbb{R}^d$ of arbitrary dimension $d \geq 1$.

The class of continuous protocols generalizes work presented in [96]. There, the Euclidean plane ($d = 2$) is considered and the class of *contracting* protocols is presented which gathers in time $\Theta(n \cdot \Delta)$, where Δ represents the initial configuration's Euclidean diameter (the initial maximum distance of two robots). The definition of contracting protocols demands that robots that are part of the global convex hull of all robots' positions move at full speed in the global closed convex hull. Every known protocol to solve GATHERING of robots with limited visibility in the continuous time model has this property. We show that contracting protocols generalize to an arbitrary dimension d (since convex hulls are analogously defined in higher dimensions) and prove an upper runtime bound of $\mathcal{O}\left(n^{H_d} \cdot \Delta\right)$, where $H_d = \sum_{i=1}^{d} 1/i \in \Theta(\log d)$ represents the d-th Harmonic number. For the runtime analysis, we use a projection technique showing that through a series of carefully chosen projections, the analysis of high-dimensional protocols can be reduced to the analysis of two-dimensional protocols. To the best of our knowledge, these are the first results for high-dimensional continuous protocols to solve GATHERING of disoriented robots with limited visibility. The specific case of $d = 3$ has already been published in the following publication which is based on some ideas derived in the Master's thesis of Michael Braun [21] (the analysis presented here was not part of the thesis). The case of $d > 3$ has not been published yet.

> **2020** (with M. Braun and F. Meyer auf der Heide) "Local Gathering of Mobile Robots in Three Dimensions" In: *Proceedings of the 2020 International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, Best Student Paper Award, cf. [22].

When considering the $\mathcal{F}$SYNC scheduler, we generalize and improve the work presented in [5, 50]. There, robots in the Euclidean plane are studied. The GO-TO-THE-CENTER (GTC) protocol is presented that moves robots towards the center of the smallest enclosing circle of all visible robots while maintaining connectivity. GTC requires $\mathcal{O}\left(n+\Delta^2\right)$ rounds to gather all robots. We prove that the runtime can be improved to $\Theta(\Delta^2)$. To do so, we present the class of λ-contracting protocols ($\lambda \in (0,1]$) that solves GATHERING in $\Theta(\Delta^2)$ rounds and we also prove that GTC belongs to this class. λ-contracting protocols restrict the allowed target points to a specific subset of a robot's local convex hull (formed by the positions of all visible robots, including itself) in the following way. Let *diam* denote the diameter of a robot's local convex hull. Then, a target point p is an allowed target point if it is the center of a line segment of length $\lambda \cdot diam$, completely contained in the local convex hull. The definition guarantees that the target point lies far enough inside the local convex hull (at least along one dimension) to decrease the swarm's diameter sufficiently. The class of protocols as well as the runtime bounds apply to robots that live in Euclidean spaces of arbitrary dimension. The lower bound of $\Omega(\Delta^2)$ even holds for a larger class of (natural) protocols: for all protocols in which robots compute target points inside the local convex hull of all visible robots (including themselves). Notably, in contrast to the time bound for

the continuous time model, the bound of $\Theta(\Delta^2)$ is independent of the dimension and the number of robots. The class of protocols as well as their analysis are based on the first part of the following publication[1].

> **2022** (with J. Harbig, D. Jung, P. Kling, T. Knollmann and F. Meyer auf der Heide) "A Unifying Approach to Efficient (Near-)Gathering of Disoriented Robots with Limited Visibility" In: *Proceedings of the 26th International Conference on Principles of Distributed Systems (OPODIS)*, cf. [26].

CHAIN-FORMATION and GATHERING in the $\mathcal{LUMI}$ Model: For Chapters 5 and 6, we switch from the $\mathcal{OBLOT}$ to the $\mathcal{LUMI}$ model and study the CHAIN-FORMATION problem for an open chain and the GATHERING problem for a closed chain of robots. Regarding the CHAIN-FORMATION problem, we improve upon work in [91] that has introduced the HOPPER protocol. The HOPPER protocol makes use of locally visible states, which can be seen as an early implementation of the $\mathcal{LUMI}$ model. After $\mathcal{O}(n)$ synchronous rounds ($\mathcal{F}$SYNC), the HOPPER protocol achieves a $\sqrt{2}$-approximation of the optimal configuration regarding the CHAIN-FORMATION problem. It is important to mention that the HOPPER protocol uses distinguishable outer robots, i.e., one outer robot participates in the protocol while the other one remains idle. We improve upon the HOPPER protocol in the following aspects:

1. We achieve a $(1+\varepsilon)$-approximation of the optimal configuration in $\mathcal{O}(n/\varepsilon)$ rounds for an à priori chosen constant $\varepsilon > 0$.
2. We show how to use a slightly increased viewing range of 2 to drop the assumption of distinguishable outer robots while keeping the time bound of $\mathcal{O}(n/\varepsilon)$ rounds.
3. We introduce a synchronization technique to transfer the protocol to the $\mathcal{A}$SYNC scheduler with a runtime of $\mathcal{O}(n/\varepsilon)$ asynchronous epochs.

Furthermore, the ideas derived for the CHAIN-FORMATION problem serve as a basis for a more involved protocol that solves GATHERING of a closed chain of robots in the Euclidean plane in $\mathcal{O}(n)$ rounds. Prior to this work, linear time GATHERING protocols assuming disorientation were only known for robots that are located on a two-dimensional grid [2, 40]. Our protocol consists of two sub-protocols: one protocol for asymmetric configurations and a second one for highly symmetric configurations. The asymmetric protocol exploits the main ideas and movement operations of the CHAIN-FORMATION protocol (in a more involved and complex way). For symmetric configurations, the ideas of the CHAIN-FORMATION protocol cannot be used as they depend on asymmetries. To overcome this drawback, we present an additional protocol for symmetric configurations. Locally, the robots can not decide whether the global chain is symmetric or asymmetric and do their decisions only based on their local view. Hence, some robots may follow the asymmetric protocol while others follow the symmetric one. Some additional coordination rules ensure that the two protocols do not hinder each other. For the GATHERING protocol, we also present a synchronization procedure to achieve a runtime of $\mathcal{O}(n)$ epochs under the $\mathcal{A}$SYNC scheduler. Both the CHAIN-FORMATION and the GATHERING protocol have been published in the following journal article.

> **2023** (with J. Harbig, D. Jung, T. Knollmann and F. Meyer auf der Heide) "Gathering a Euclidean Closed Chain of Robots in Linear Time and Improved Algorithms for Chain-Formation" In: *Theoretical Computer Science (TCS)*, cf. [28].

A preliminary version of the GATHERING result appeared in the conference proceedings of ALGOSENSORS 2021:

[1] The second part of the publication is mainly authored by the co-author Jonas Harbig. There, we transformed the class of protocols to solve the NEAR-GATHERING problem. NEAR-GATHERING requires the robots to keep unique positions while gathering in close proximity. We show that also NEAR-GATHERING can be solved in $\Theta(\Delta^2)$ rounds by disoriented robots with limited visibility.

> **2021** (with J. Harbig, D. Jung, T. Knollmann and F. Meyer auf der Heide) "Gathering a Euclidean Closed Chain of Robots in Linear Time" In: *Proceedings of the 17th International Symposium on Algorithms and Experiments for Wireless Sensor Networks (ALGOSENSORS)*, Best Paper & Best Student Paper Award, cf. [27].

The MAX-CHAIN-FORMATION Problem: Chapter 7 opens the second part of the thesis and studies the MAX-CHAIN-FORMATION problem. On a conceptual level, the problem can be seen as an extension of the CHAIN-FORMATION problem: the outer robots gain the ability to move and the goal changes to maximize the distance between the two outer robots. We observe that these changes add a lot of complexity to the problem because the outer robots must move in opposite directions despite not being able to see each other. We study the problem in the $\mathcal{OBLOT}$ model and in two different time models: fully synchronous rounds ($\mathcal{F}$SYNC) and the continuous time model. For the $\mathcal{F}$SYNC time model, we present a conceptually very simple protocol: inner robots move to the midpoint between their neighbors and outer robots move as far as possible away from their direct neighbors without losing connectivity. It turns out that even in one-dimensional configurations (all robots' positions are collinear), the protocol exhibits a complex behavior: some configurations converge to the optimal configuration while others turn into a final configuration that keeps moving through the plane forever. We characterize this behavior and prove time bounds of $\Omega(n^2 \cdot \log(1/\varepsilon))$ and $\mathcal{O}(n^2 \cdot \log(n/\varepsilon))$ rounds until a $(1-\varepsilon)$- approximation of one of the final configurations is reached. The time bounds match the currently best known time bounds for the CHAIN-FORMATION problem under the same robot model [60, 88]. Moreover, we prove that we can construct two-dimensional configurations causing the protocol to have an arbitrarily high runtime.

Regarding the continuous time model, we present a similar protocol with an interesting twist: while all inner robots move at full speed, the outer robots are (mostly) slowed down by a constant. We prove that the speed difference leads to a time-optimal protocol, requiring $\Theta(n)$ time. Some configurations, however, collapse to a single point instead of converging to the optimal configuration. Furthermore, we prove that the speed difference does not lead to a speedup in the $\mathcal{F}$SYNC time model. All in all, we see time bounds that are similar to the comparable contracting problem CHAIN-FORMATION; however, the protocols are more complex in terms of convergence behavior. The problem itself, the presented protocols and their analyses are based on the following journal article.

> **2022** (with P. Kling, T. Knollmann and F. Meyer auf der Heide) "A Discrete and Continuous Study of the Max-Chain-Formation Problem" In: *Information and Computation*, cf. [30].

A preliminary version appeared in the conference proceedings of SSS 2020:

> **2020** (with P. Kling, T. Knollmann and F. Meyer auf der Heide) "A Discrete and Continuous Study of the Max-Chain-Formation Problem – Slow down to Speed Up" In: *Proceedings of the 22nd International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, Best Paper Award, cf. [29].

The MAX-LINE-FORMATION Problem: Chapter 8 is about the MAX-LINE-FORMATION problem, which has the same goal as the MAX-CHAIN-FORMATION problem: to arrange the robots on a straight line of length $n-1$. In contrast to the MAX-CHAIN-FORMATION problem, the robots follow the standard connectivity model (the robots are not arranged in a chain). On one hand, the robots obtain more freedom in arranging themselves since there are no predefined roles (any robot could for instance be the endpoint of the line). On the other hand, we observe additional complexity because robots sometimes cannot locally decide whether they should be an endpoint of the global line. For the $\mathcal{OBLOT}$ model, we present an impossibility result stating that the problem is impossible to solve by robots with a constant-sized viewing range, even if the robots agree on

both axes of their local coordinate systems. Even worse, it is impossible to find protocols that only converge to the optimal configuration.

Afterward, we show how switching from circular to square connectivity and viewing ranges enables the robots to solve the problem, at least if they agree on one axis of their local coordinate systems. For square connectivity and viewing ranges, we present three protocols. All protocols operate in two global phases (not distinguishable by the robots due to their limited visibility). First, the robots form a line parallel to the y-axis. Afterward, the robots stretch the line to a maximum length by applying ideas from the MAX-CHAIN-FORMATION problem.

The first protocol considers the $\mathcal{OBLOT}$ model and the $\mathcal{S}$SYNC scheduler and converges to the optimal configuration in $\mathcal{O}(n^2 \cdot \log(n/\varepsilon))$ epochs (for every $\varepsilon \in (0,1)$, a $(1-\varepsilon)$-approximation is reached after that time). The analysis transfers techniques designed for the analysis of specific time inhomogeneous Markov chains to the deterministic robot world.

Lastly, we consider the $\mathcal{LUMI}$ model to solve the problem exactly. The second protocol is designed for the $\mathcal{S}$SYNC scheduler and reaches the optimal configuration on $\mathcal{O}(n^2)$ epochs. Finally, we design a protocol for the $\mathcal{F}$SYNC scheduler that solves the problem optimally in $\Theta(n)$ rounds. The results are based on the following publication.

> **2021** (with T. Goette, T. Knollmann and F. Meyer auf der Heide) "The Max-Line-Formation Problem – and new Insights for Gathering and Chain-Formation" In: *Proceedings of the 23rd International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, cf. [25].

2. Robot Models & Notation

The following sections introduce the robot models as well as the notation we use in this thesis. We first define the $\mathcal{OBLOT}$ model thoroughly in Section 2.1. Afterward, we discuss the differences of the $\mathcal{LUMI}$ model in Section 2.2. Subsequently, we define robot chains, an extension of the $\mathcal{OBLOT}$ and $\mathcal{LUMI}$ models in Section 2.3. Finally, we introduce an abbreviated notation to define robot models in Section 2.4.

2.1 $\mathcal{OBLOT}$

The $\mathcal{OBLOT}$ model (an abbreviation of *oblivious robots*) [66] has a set of fundamental features (such as the name-giving obliviousness) we present first in Section 2.1.1. Apart from that, the $\mathcal{OBLOT}$ model offers a variety of choices regarding other aspects of the robots, e.g., the (dis-) agreement on the local coordinate systems, limited vs. unlimited visibility or the synchronization of the robots' actions. We introduce our assumptions on the local coordinate systems of the robots in Section 2.1.2 and present the different time models in Section 2.1.3.

2.1.1 Fundamental Features

The swarm consists of n robots $\mathcal{R} = \{r_0, \dots, r_{n-1}\}$ living in $\mathbb{R}^d$ ($d \in \mathbb{N}$). The position of a robot r_i at time t is denoted as $p_i(t) \in \mathbb{R}^d$ in a global coordinate system (not known to the robots). We collectively call the set of all robots' positions at time t the *configuration*. The robots do not have any extent, i.e., there can be multiple robots located at the same position. The initial positions of all robots are pairwise distinct. All robots have a local memory and can do computations with infinite precision real arithmetic. Moreover, the robots have precise movement capabilities, i.e., they can turn and move in any direction. The robots are *autonomous*, i.e., they operate without any central control. Moreover, they are *identical* and *anonymous*, i.e., they all have the same external appearance and no (unique) identifiers. Furthermore, the robots are *homogeneous* and *deterministic*: all robots execute the same deterministic protocol. The two most characteristic features of the $\mathcal{OBLOT}$ model are that robots are *oblivious* and *silent*. Oblivious robots do not have any memory of the past (see Section 2.1.3 for a precise notion of time) and silent robots do not have any ability to communicate directly with other robots. Since the robots can only communicate via observations and movements, the communication is also called to be *stigmergic*.

2.1.2 Orientation & Visibility

Each robot has its local coordinate system that is self-centered, i.e., the robot itself is located at the origin. We describe by $p_i^j(t)$ the position of robot r_j in the local coordinate system of r_i. The

robots have *limited visibility*, i.e., they can observe other robots up to a constant distance, denoted as the *viewing range*. We distinguish the terms *circular* and *square* viewing ranges. With a circular viewing range, robots can observe all other robots at a constant distance of $\mathcal{V}$. The initial visibility graph, i.e., the graph where the nodes represent the robots and two robots share an edge if their distance is at most $\mathcal{V}$, is always connected. Sometimes it is beneficial or even necessary to assume a *well-connected* initial configuration, i.e., that the swarm is even connected by a value that is a constant smaller than the viewing range. More formally, we distinguish the terms *connectivity range* and viewing range. The swarm is connected concerning the connectivity range $\mathcal{C}$ while the robots have a viewing range $\mathcal{V} \geq \mathcal{C}$. Mostly, we assume $\mathcal{C} = \mathcal{V}$. For simplicity, we assume that $\mathcal{C} = 1$ throughout the entire thesis. Then, the notion of connectivity ranges induces a *Unit Ball Graph* $\mathrm{UBG}(t) = (\mathcal{R}, E(t))$, where $\{r_i, r_j\} \in E(t)$ if and only if $\|p_i(t) - p_j(t)\|_2 \leq 1$, where $\|\cdot\|_2$ denotes the Euclidean distance. The initial Unit Ball Graph $\mathrm{UBG}(0)$ is connected. Robots with a *square* viewing range of $\mathcal{C}$ can observe other robots that are located in a self-centered and axis-aligned square of side length $2 \cdot \mathcal{V}$. Two robots at a distance of at most 1 are also called *neighbors*. For a robot r_i, the set $N_i(t)$ describes the set of all visible neighbors including itself at time t. The set $N_i(t)$ is also called the *neighborhood* of a robot r_i at time t.

Furthermore, the robots do not share a common chirality, i.e., the robots do not agree on a cyclic orientation of their environment (e.g., *clockwise*). Usually, with one exception in Chapter 8, we consider *disoriented* robots. Disoriented robots have no agreement on their local coordinate systems, i.e., the axes can be arbitrarily rotated and inverted. The disorientation is even *variable*, i.e., the rotation and orientation of the axes of one fixed robot can change from time to time. A stronger assumption, we (have to) use in Chapter 8, is denoted as k-axes agreement: the robots agree on the direction and orientation of $k < d$ axes (the axes are fixed).

2.1.3 Time

Time can be either discrete (round based) or continuous[1]. In the following, we describe the different models in detail.

Discrete Time. When time is discrete, robots operate in LCM cycles consisting of the operations LOOK, COMPUTE and MOVE. During LOOK, each robot takes a snapshot of all visible robots in its local coordinate system, it computes a target point based on the snapshot during COMPUTE and finally moves there within the MOVE operation. We assume a *rigid* movement, i.e., the robots always reach their target points. The LCM cycles are also denoted as *rounds*. The timing of the cycles is controlled by an (adversarial) scheduler that can either be fully synchronous ($\mathcal{F}$SYNC), i.e., each robot is active in every round and all robots start a new round simultaneously, semi-synchronous ($\mathcal{S}$SYNC), i.e., in every round, only a non-empty subset of all robots is activated and starts the new round simultaneously or completely asynchronous ($\mathcal{A}$SYNC), i.e., the starting points and durations of different robots' cycles can be completely different. Both the $\mathcal{S}$SYNC or $\mathcal{A}$SYNC scheduler are *fair*, i.e., the time between two successive activations of the same robot is always finite. Moreover, we call a schedule to be *k-fair* if, between every two successive activations of any robot r_i, every other robot is activated at most k times. By definition, the $\mathcal{F}$SYNC scheduler is 1-fair.

For the $\mathcal{F}$SYNC and $\mathcal{S}$SYNC schedulers, we define the class of discrete formation protocols that are a convenient way of formally describing the behavior of the robots in each round.

Definition 2.1 A *discrete robot formation protocol* $\mathcal{P}$ defines for each robot r_i and each $t \in \mathbb{N}$ a *target point* $\mathrm{target}_i^{\mathcal{P}}(t)$ such that $p_i(t+1) = \mathrm{target}_i^{\mathcal{P}}(t)$.

Continuous Time. We also study the continuous time model which defines movements for every *real* point in time $t \in \mathbb{R}_{\geq 0}$. At every point in time, each robot computes a target point and

[1] In the original $\mathcal{OBLOT}$ definition, only discrete time is considered [66]. However, since except for the time model all other capabilities are identical, we decided to add continuous time here.

a speed between 0 and 1 and moves with the given speed towards this point. More formally, we obtain the following notion of continuous protocols.

Definition 2.2 A *continuous robot formation protocol* $\mathcal{P}$ defines for each robot r_i and each $t \in \mathbb{R}_{\geq 0}$ a *target point* $\text{target}_i^{\mathcal{P}}(t)$ and a *speed* $s_i^{\mathcal{P}}(t) \in [0,1]$ describing the robots current movement direction and speed. Together, both define the *velocity vector*

$$v_i^{\mathcal{P}}(t) = \frac{s_i^{\mathcal{P}}(t)}{\|\text{target}_i^{\mathcal{P}}(t) - p_i(t)\|_2} \left(\text{target}_i^{\mathcal{P}}(t) - p_i(t)\right).$$

Often, we describe only the velocity vectors. However, typical protocols are designed such that each robot computes a target point and moves with a certain speed towards it. The function $p_i : \mathbb{R}_{\geq 0} \to \mathbb{R}^d$ is the *trajectory* of r_i. The trajectories are continuous but not necessarily differentiable. In some occasions, robots can change their direction and speed non-continuously (e.g., when a new robot becomes visible) leading to a non-differentiable trajectory. Typical protocols have right-differentiable trajectories. This way, we can interpret the velocity vectors as the (right) derivative of the trajectories, i.e., $v_i^{\mathcal{P}} : \mathbb{R}_{\geq 0} \to \mathbb{R}^d = \dot{p}_i$.

There is one specific, at first sight, counterintuitive behavior of the continuous time model, we want to emphasize. The continuous time model is certainly idealized since robots do infinitely many observations in finite time, and there is no delay between sensing and adjusting the movement. Therefore, the model is physically not realizable but approximates the real world reasonably well [72]. One effect of the idealized definition is the *Zenoness* phenomenon: in some cases, robots move as if they knew the velocity vectors of their neighbors (or even robots that are farther away) although they cannot know them explicitly due to the limited visibility. Consider the following example: For simplicity, we assume that the robots have a common understanding of up and down. Additionally, robots can observe the positions of other robots up to a distance of 1. There are three robots, r_1, r_2 and r_3 with positions $p_1(t) = (0,0)$, $p_2(t) = (0.5,0)$ and $p_3(t) = (1,0)$. Our protocol $\mathcal{P}$ moves the robots r_1 and r_3 with a speed of 1 upwards. For r_2, it demands not to move in case it is located on the midpoint between r_1 and r_3, i.e., $p_2(t) = \frac{1}{2}p_1(t) + \frac{1}{2}p_3(t)$, and otherwise move with speed 1 towards $\frac{1}{2}p_1(t) + \frac{1}{2}p_3(t)$. Figure 2.1 visualizes the example.

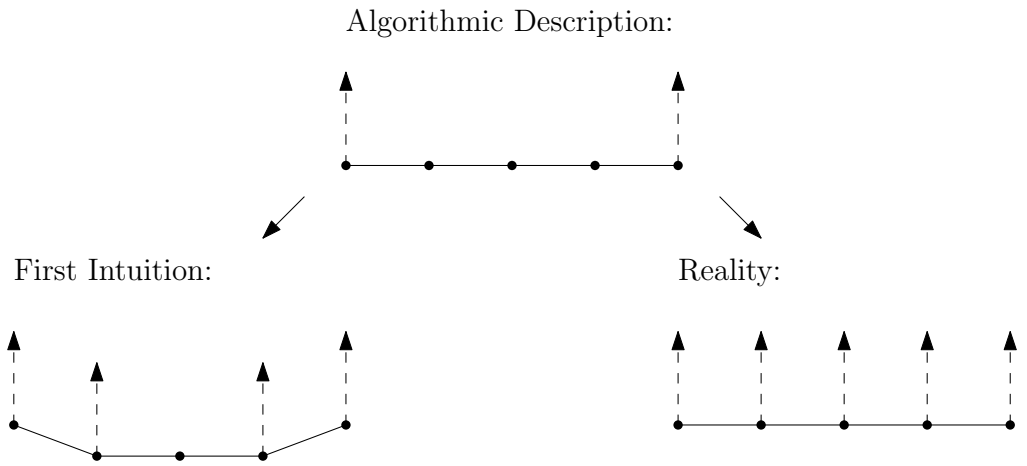

Figure 2.1: Above, the algorithmic description is depicted: robots at the end of the line move with speed 1 upwards. All other robots do not move as they are already located at the midpoint between their direct neighbors. The lower left picture shows the first intuition: robots at the end of the line move upwards, and the next robots follow shortly. The lower right picture shows the reality: all robots simultaneously move upwards with a speed of 1.

Based on the previously described movement rules, one could think $v_1^{\mathcal{P}}(t) = v_3^{\mathcal{P}}(t) = (0,1)$ and $v_2^{\mathcal{P}}(t) = (0,0)$ since r_2 is already located on the midpoint between r_1 and r_3. Consequently, the

first intuition would be that r_1 and r_3 start moving upwards and r_2 follows the movement shortly. However, this does not happen and it holds $v_2^{\mathcal{P}}(t) = (0,1)$ as well. To understand this, suppose there is a time $t' > t$ with $p_1(t) = (0, t'-t)$, $p_3(t) = (1, t'-t)$ and $p_2(t) = (0.5, 0)$. Since r_1 and r_3 move always with speed 1 upwards, it holds for all $t'' \in (t, t')$: $v_1^{\mathcal{P}}(t'') = v_3^{\mathcal{P}}(t'') = (0,1)$. Thus, for infinitely many $t'' \in (t, t')$ it holds $p_2(t'') \neq \frac{1}{2}p_1(t'') + \frac{1}{2}p_3(t'')$ and r_2 must have moved upwards. As t' can be chosen arbitrarily small, we conclude $v_2^{\mathcal{P}}(t) = (0,1)$. Now, suppose that there are n robots $r_0, \ldots, r_{n-1}$ with $p_i(t) = (\frac{i}{2}, 0)$ located on the x-axis. Robots that do not see another robot to the left or the right, holding for r_0 and r_{n-1} in the given configuration, move with speed 1 upwards. All other robots do not move in case they are located on the midpoint between their neighbors, and otherwise, they move with a speed of 1 to the midpoint. With the same arguments as for the configuration with 3 robots, all robots simultaneously move upwards with speed 1 although they cannot know their neighbors' velocity vectors due to the limited visibility of 1.

2.2 $\mathcal{LUMI}$

The $\mathcal{LUMI}$ (long: $\mathcal{LUMINOUS}$) model is in many parts identical to the $\mathcal{OBLOT}$ model but adds a memory and communication ability to the robots: (locally) visible lights [53]. Hence, the robots are not oblivious and silent. Instead, each robot is equipped with a light that has at any time of color out of a color set $\mathcal{C}$. The light can be perceived by nearby robots during their LOOK phase. During the COMPUTE phase, robots can change the color of their light such that the new color of the light becomes visible to other robots in the next round. Since the light can be perceived by other robots, constant-size information can be communicated to the neighbors. Moreover, since the own light is persistent (unless intentionally switched off), it can be used to store constant-size information for the next round. Sometimes it is convenient to describe our protocols with help of *multiple* lights $\ell_1, \ldots, \ell_k$, each having its own color set $\mathcal{C}_1, \ldots, \mathcal{C}_k$. Note that the multiple lights can always be represented by a single light with $|\mathcal{C}| = \prod_{i=1}^{k} |\mathcal{C}_i|$ colors.

2.3 Chains

As mentioned in Chapter 1, we also consider chains of robots which can be seen as an extension (or modification) of the $\mathcal{OBLOT}$ and $\mathcal{LUMI}$ models. The main difference is that robots have predefined neighborhoods of robots they can observe. More precisely, in an open chain, the robots r_0 and r_{n-1} are endpoints of the chain having only a single neighbor. The *direct* neighbor of r_0 is r_1 and r_{n-2} is the direct neighbor of r_{n-1}. Both r_0 and r_{n-1} are also called *outer* robots. All other robots are *inner* robots. The *direct* neighbors of a robot r_i are r_{i-1} and r_{i+1}. In a closed chain, all robots are inner robots (no outer robots exist) and r_0 and r_{n-1} are each other's direct neighbors. Whenever closed chains are considered, all operations on indices have to be understood modulo n.

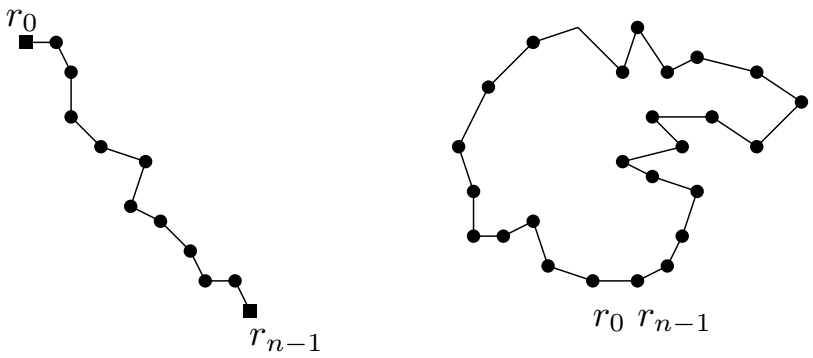

Figure 2.2: A visualization of an open chain (left) and a closed chain (right).

For chains, we define the vectors $w_i(t) = p_i(t) - p_{i-1}(t)$. In open chains $w_0(t)$ is undefined. These vectors are important to express the impact of protocols on the chain. Moreover, we define the

length of a chain at time t as $L(t) := \sum_{i=1}^{n-1} \|w_i(t)\|_2$ for open chains and $L(t) := \sum_{i=0}^{n-1} \|w_i(t)\|_2$ for closed chains. Sometimes we consider the normalized vectors: $\widehat{w}_i(t) := \frac{w_i(t)}{\|w_i(t)\|_2}$. The angle created by anchoring $w_{i+1}(t)$ at the terminal point of $w_i(t)$ is denoted by $\alpha_i(t) = \measuredangle(w_i(t), w_{i+1}(t)) \in [0, \pi]$. Moreover, $\text{sgn}_i(\alpha_i(t)) \in \{-1, 0, 1\}$ denotes the orientation of $\alpha_i(t)$ from r_i's point of view (this can differ from robot to robot) and $\text{sgn}(\alpha_i(t))$ denotes the orientation in a global coordinate system. Both in open and closed chains, robots have limited visibility. The robots have a connectivity range of $\mathcal{C} = 1$ which means that the distance between two *direct* neighbors is allowed to be at most 1. The initial configuration is always connected. Additionally, the robots have a viewing range of $\mathcal{V} \geq C (\mathcal{V} \in \mathbb{N})$. The viewing range of $\mathcal{V}$ allows the robots to observe the positions of V neighbors along the chain in each direction. More formally, a robot r_i is able to observe the positions of the robots $r_{i-\mathcal{V}}, \ldots, r_i, \ldots, r_{i+\mathcal{V}}$. Apart from these robots, no other robots are visible even if they are nearby. Although the robots can identify their neighbors, they do not have a common understanding of left or right neighbors, i.e., the chain itself is also disoriented.

2.4 Naming

Throughout the thesis, we use several instances of the above-described robot models. Next, we introduce a brief notation to describe the used models compactly.

Definition 2.3 Let M^F, M^S, M^A and M^C denote the swarm of robots in the model $M \in \{\mathcal{OBLOT}, \mathcal{LUMI}\}$ under the $\mathcal{F}$SYNC (F), $\mathcal{S}$SYNC (S) or $\mathcal{A}$SYNC (A) schedulers or in the continuous time model (C).

Following Definition 2.3, $\mathcal{OBLOT}^F$ represents robots that are characterized by the $\mathcal{OBLOT}$ model under the $\mathcal{F}$SYNC scheduler and $\mathcal{LUMI}^A$ represents robots in the $\mathcal{LUMI}$ model under the $\mathcal{A}$SYNC scheduler. Moreover, we introduce a notation that reflects also the viewing ranges of the robots.

Definition 2.4 Let M^T, $M \in \{\mathcal{OBLOT}, \mathcal{LUMI}\}$, $T \in \{F, S, A, C\}$ defined as in Definition 2.3. $M^T_{\mathcal{V}}$, $\mathcal{V} \in \mathbb{N}$ denotes M^T for robots with *limited visibility*, a connectivity range of 1 and a viewing range of $\mathcal{V}$.

According to Definition 2.4, $\mathcal{OBLOT}$ robots in the continuous time model that have an equal viewing and connectivity range of 1 are described by $\mathcal{OBLOT}^C_1$. Similarly, $\mathcal{LUMI}^S_4$ represents robots in the $\mathcal{LUMI}$ model with a connectivity range of 1 and a viewing range of 4 that operate under the $\mathcal{S}$SYNC scheduler.

Lastly, in the case that chains are considered, we abuse the notation and naming slightly. Usually, as mentioned in Section 2.3, the existence of a chain contradicts some core properties of the $\mathcal{OBLOT}$ or $\mathcal{LUMI}$ models (since robots can recognize their predefined neighbors). Still, in all other aspects, the robots have the same properties as robots in the $\mathcal{OBLOT}$ or $\mathcal{LUMI}$ models. To simplify the model descriptions, whenever we discuss GATHERING of a closed chain of robots, the CHAIN-FORMATION or the MAX-CHAIN-FORMATION problems, we say that the robots follow the $\mathcal{OBLOT}$ or $\mathcal{LUMI}$ models and mean implicitly that the robots follow the models except for the existence of a chain.

3. Related Work

We discuss related work falling into the research area of distributed computing of mobile robots. For practical applications of (mobile) robot swarms, we refer the reader to the comprehensive survey [112]. In the context of distributed computing by mobile robots, we focus on the following broad topics: models and their relations, the ARBITRARY-PATTERN-FORMATION problem, the GATHERING problem, the CHAIN-FORMATION problem and expanding formation problems. A survey on the entire research field of distributed computing by mobile robots can be found in [1]. The first two paragraphs about models and their relations as well as the ARBITRARY-PATTERN-FORMATION problem discuss the foundations of the thesis: Which robot models exist and how are they related? Moreover, research about the ARBITRARY-PATTERN-FORMATION problem studies which formation problems can theoretically be solved at all. The paragraphs about GATHERING and CHAIN-FORMATION embed the results of Part I. Lastly, the paragraph on expanding formation problems studies related work regarding Part II. Most of the literature we discuss in the next paragraphs studies robots in the Euclidean plane. For simplicity, we always implicitly assume that the robots live in the Euclidean plane. Whenever results hold for other environments (e.g., higher dimensional Euclidean spaces or discrete domains), we mention the environment explicitly.

Models and Their Relations

The $\mathcal{OBLOT}$ model has emerged as the de-facto standard robot model. While the name and the standardization are quite young (from 2019 [66]), the first occurrences of the model (without the name) date back to the 90s [5, 6, 113, 114]. In the following, we describe relations between different models with a notation first introduced in [42]. For two models M_1 and M_2, we write $M_1 > M_2$ if the robots in model M_1 can solve all problems that robots in model M_2 can solve and there is at least one problem robots in M_1 can solve that cannot be solved by robots in M_2. In this case, we call M_1 to be *more powerful* than M_2. Additionally, M_1 and M_2 are called *computationally equivalent*, in symbols $M_1 \equiv M_2$, if the robots in M_1 and M_2 can solve the same set of problems. The last option is that the models are *incomparable*, $M_1 \perp M_2$, in case there is a problem robots in M_1 can solve but robots in M_2 cannot and vice versa.

It is well known that $\mathcal{OBLOT}^F > \mathcal{OBLOT}^S$. As the $\mathcal{F}$SYNC scheduler is a valid $\mathcal{S}$SYNC scheduler, robots in the $\mathcal{OBLOT}^F$ model can solve all problems that robots in the $\mathcal{OBLOT}^S$ model can solve. The $\mathcal{OBLOT}^F$ model is more powerful than the $\mathcal{OBLOT}^S$ model since GATHERING of two disoriented robots (also denoted as RENDEZVOUS) is possible under the $\mathcal{F}$SYNC but impossible under the $\mathcal{S}$SYNC scheduler [114]. The relation between $\mathcal{OBLOT}^S$ and $\mathcal{OBLOT}^A$ was for a very long time unclear. Trivially, $\mathcal{OBLOT}^S$ robots can solve every problem $\mathcal{OBLOT}^A$ robots can solve (since every $\mathcal{S}$SYNC schedule is also a valid $\mathcal{A}$SYNC schedule). Very recently, it was

proven that the COHESIVE CONVERGENCE problem of robots with limited visibility but with measurement errors can be solved in the $\mathcal{OBLOT}^S$ model but not in the $\mathcal{OBLOT}^A$ model, proving that $\mathcal{OBLOT}^S > \mathcal{OBLOT}^A$ [87]. In the COHESIVE CONVERGENCE problem, the robots have to converge to a single point and robots that are visible to each other have to remain visible (a natural assumption most protocols have). Due to the combination of measurement errors and asynchrony, it is impossible to keep connectivity in some cases, as robots do not know whether their neighbors are moving. It is still open whether unreliable measurements are crucial for this relationship or if there is a problem $\mathcal{OBLOT}^S$ robots can solve that $\mathcal{OBLOT}^A$ robots cannot solve without considering any measurement errors.

Inspired by visible light communication, for instance in car networks [86], the $\mathcal{LUMI}$ model has been proposed [41, 42, 53]. While it seems to be evident that the $\mathcal{LUMI}$ models are more powerful than the $\mathcal{OBLOT}$ models, it is not entirely clear yet. Clearly, robots in the $\mathcal{LUMI}^T$ model, $T \in \{F,S,A\}$ can solve all problems $\mathcal{OBLOT}^T$ robots can solve (just by *not* using their light). By showing that the RENDEZVOUS problem can be solved in the $\mathcal{LUMI}^A$ model (which is impossible in $\mathcal{OBLOT}^S$), it was proven that $\mathcal{LUMI}^A > \mathcal{OBLOT}^S$ (and thus also $\mathcal{LUMI}^S > \mathcal{OBLOT}^S$ since $\mathcal{LUMI}^S$ robots can solve all problems $\mathcal{LUMI}^A$ robots can solve) [42]. Moreover, it is also known that there are problems $\mathcal{LUMI}^A$ robots can solve that $\mathcal{OBLOT}^F$ robots cannot solve. An example of such a problem is the OSCILLATING POINTS problem, where two robots have to move alternatingly towards and away from each other. Since $\mathcal{OBLOT}^F$ robots cannot remember what they did last, there are configurations in which the robots do not know whether they should move towards or away from each other. The presence of lights (and thus memory) allows $\mathcal{LUMI}^A$ robots to solve the problem. Nevertheless, it is open whether $\mathcal{LUMI}^A > \mathcal{OBLOT}^F$ or $\mathcal{LUMI}^A \perp \mathcal{OBLOT}^F$ [42]. Notably, if the robots move on the nodes of graphs, it could be proven that the corresponding $\mathcal{LUMI}^A$ and $\mathcal{OBLOT}^F$ models are incomparable [51].

Interestingly, through the presence of lights, the difference between the $\mathcal{S}$SYNC and $\mathcal{A}$SYNC schedulers disappear. More formally, $\mathcal{LUMI}^S \equiv \mathcal{LUMI}^A$ [42]. The equivalence was proven through a simulation approach: the authors in [42] provide a protocol that simulates every $\mathcal{LUMI}^S$ protocol by $\mathcal{LUMI}^A$ robots. The lights are used to synchronize the robots such that their movements follow a valid $\mathcal{S}$SYNC schedule.

Furthermore, there are two intermediate models, $\mathcal{FCOM}$ and $\mathcal{FSTA}$ [65]. Both can be interpreted similarly to the $\mathcal{LUMI}$ model: robots are equipped with a light. In the $\mathcal{FCOM}$ model, robots cannot see their own light (robots gain only the ability to communicate) and in the $\mathcal{FSTA}$ model, the light is internal and cannot be observed by other robots (robots gain only the memory capability). The relation between the $\mathcal{FCOM}$, $\mathcal{FSTA}$, $\mathcal{OBLOT}$ and $\mathcal{LUMI}$ models has been studied in [70]. First of all, the authors prove that $\mathcal{FCOM}^F \equiv \mathcal{LUMI}^F$, again by simulating the models in each other. In addition, $\mathcal{LUMI}^F > \mathcal{FSTA}^F$ and $\mathcal{FSTA}^F > \mathcal{OBLOT}^F$. The relation $\mathcal{FSTA}^F > \mathcal{OBLOT}^F$ can again be proven with the OSCILLATING POINTS problem since the light has only been used as a memory to store the last direction in which the robots move. The relation $\mathcal{LUMI}^F > \mathcal{FSTA}^F$ was proven by a problem in which the robots have to form a sequence of patterns. Lots of further relations are known, we refer the interested reader to [70].

Lastly, we would like to mention the $\mathcal{R}$SYNC scheduler [23] which can be seen as an intermediate scheduler between $\mathcal{F}$SYNC and $\mathcal{S}$SYNC. Practically, it is motivated by energy-constrained robots: after completing one LCM cycle, the robots have to rest for one cycle and can continue afterward. On the theoretical side, the scheduler can be seen as a strong intermediate scheduler between $\mathcal{S}$SYNC and $\mathcal{F}$SYNC. Let R denote the $\mathcal{R}$SYNC scheduler. The authors prove $\mathcal{LUMI}^F > \mathcal{LUMI}^R \equiv \mathcal{LUMI}^S$ [23]. Hence the authors got closer to the real boundary between the $\mathcal{F}$SYNC scheduler and less synchronized models. In the paper, lots of further relations are studied which are out of scope for this chapter.

Apart from the previously described models, lots of related, biologically-inspired models with different foci exist [45, 52, 90, 116]. Probably closest to the previously described ones, the $\mathcal{MOBLOT}$ model describes a two-stage process of robots combining themselves in molecules

first and afterward form patterns as molecules [32, 33]. The smallest computational entities are robots following mostly the $\mathcal{OBLOT}$ model with the difference that the robots are not identical in their appearance but might have different colors. Based on their colors, the robots get certain roles in molecules they have to form. As soon as the robots have arranged themselves in a molecule, the robots in the molecule can move collectively to build larger structures. The model itself is quite new and the first work presents the necessary conditions to build the molecules first, which can be seen as a generalized PATTERN-FORMATION problem [32]. A subsequent work studies the $\mathcal{MOBLOT}$ model by robots that are located on a two-dimensional grid [33]. Other models are inspired by the vision of *programmable matter*. The celebrated Amoebot model [45, 52] considers tiny *particles* (similar to robots) that live on grid graphs. The name of the model stems from the movement of the particles that is inspired by amoeba: the particles first expand to the next cell and contract afterward. The Amoebot model has lots of similarities to the different robot models, e.g., local communication ($\mathcal{LUMI}$) or persistent memory ($\mathcal{LUMI}$ or $\mathcal{FSTA}$). Apart from the movement through contraction and expansion, a further severe difference is that particles can update variables in the local memory of neighboring particles. In a new extension, the particles can also connect themselves in reconfigurable circuits through which beeps can send (e.g., to synchronize certain movements) [63]. For a survey about computing by programmable particles, including the Amoebot but also further models, we refer to [44].

The ARBITRARY PATTERN-FORMATION Problem

The ARBITRARY PATTERN FORMATION problem is the most general robot formation problem: the robots have to form a pattern they receive as input. Usually, the robots can choose where to form the pattern (the positions are not predefined) and the robots can form arbitrary rotations of the pattern at an arbitrary scale. The pattern is either given as a set of points or as a geometric predicate, e.g., "circle". We first summarize the dense literature about robots with *unlimited visibility* (robots can observe all other robots) and focus afterward on robots with limited visibility. Unless otherwise stated, the results consider the $\mathcal{OBLOT}$ model.

Unlimited Visibility. In the general form, the ARBITRARY-PATTERN-FORMATION problem is impossible to solve under most robot capabilities due to potential symmetries of initial configurations. Precisely, without agreement on both axes of the local coordinate systems, ARBITRARY-PATTERN-FORMATION is impossible even in the $\mathcal{OBLOT}^F$ model with chirality (i.e., the robots agree on a common handedness, e.g., clockwise) [69]. The impossibility, however, only holds for configurations where the number of robots is even. If the robots agree on only one axis of their local coordinate systems and the number of robots is odd, ARBITRARY-PATTERN-FORMATION can be solved in $\mathcal{OBLOT}^A$ [69]. For both an odd and an even number of robots ARBITRARY-PATTERN-FORMATION can only be solved if the robots agree on both axes of their local coordinate systems (also denoted as *consistent compasses*) [69]. However, the impossibility results mostly rely on the observation that starting from a highly symmetric configuration, e.g., the positions of the robots form a circle, no configuration with fewer symmetries can be constructed. Still, some other configurations might be formable. The resulting question is: Depending on the input configuration, which target patterns can be constructed? In [114], the question has been answered based on the notion of the *symmetricity* of a configuration. Consider a set $\mathbb{P}$ of distinct points. The symmetricity $\rho(\mathbb{P})$ of $\mathbb{P}$ is defined to be 1 if the center of the smallest enclosing circle of $\mathbb{P}$ contains a point of $\mathbb{P}$. Otherwise, $\rho(\mathbb{P})$ is the number of angles $\alpha \in (0, 2\pi]$ such that rotation of $\mathbb{P}$ by α yields $\mathbb{P}$ again. Based on this definition, a target pattern $\mathbb{F}$ is only formable starting in an input configuration $\mathbb{I}$ if $\rho(\mathbb{I})$ divides $\rho(\mathbb{F})$. An important consequence is that the only patterns that might be formable from *any* input configuration must have a high symmetry. More precisely, the patterns POINT (in the literature denoted as the GATHERING problem) and UNIFORM-CIRCLE are the only patterns that might be formed independent of the input configuration. This observation underlines the importance of the GATHERING problem (and the UNIFORM-CIRCLE-FORMATION problem) from a theoretical point of view.

On the positive side, at least for the $\mathcal{S}$SYNC scheduler, there are protocols to form any of the potentially formable target patterns. In [117], a protocol for the $\mathcal{OBLOT}^S$ model for robots with a common chirality has been designed that can arrange the robots in any possibly formable pattern. For the $\mathcal{A}$SYNC scheduler, no such result is known. Notably, the authors of [43] study the formation of a series of patterns of robots in the $\mathcal{OBLOT}$ model. Since the robots have no memory, it is not possible to form arbitrary series of patterns as the robots must be able to recall which pattern they have to form next. In [43], a complete characterization of the relation between the patterns in the series is derived.

Similar results have been derived for robots living in the three-dimensional Euclidean space [118, 119]. Moreover, the ideas of symmetricity have also been transferred to discrete domains, i.e., graphs, and the ARBITRARY-PATTERN-FORMATION has been studied for robots located on graphs of different classes, mostly two-dimensional grids [13, 34, 76].

Limited Visibility. In the case of limited visibility, the picture is worse compared to the unlimited visibility case. Some results from the unlimited visibility case transfer to this case. More precisely, no target pattern $\mathbb{F}$ can be formed starting in an input configuration $\mathbb{I}$ with $\rho(\mathbb{I}) > \rho(\mathbb{F})$. However, in [120] it has been proven that in the $\mathcal{OBLOT}^F$ model, there always exists a target pattern $\mathbb{F}$ that cannot be formed starting in an input configuration $\mathbb{I}$ even if $\rho(\mathbb{I})$ divides $\rho(\mathbb{F})$. As a consequence, the limited visibility capability of the robots severely weakens the formation problems the robots can solve. On the positive side, it could be shown that equipping the robots with memory makes the robots as powerful as in the case with unlimited visibility. Precisely, non-oblivious robots under the $\mathcal{F}$SYNC scheduler (even with non-rigid moves) can form any target pattern $\mathbb{F}$ from an input configuration $\mathbb{I}$ if and only if $\rho(\mathbb{I})$ divides $\rho(\mathbb{F})$. Under the $\mathcal{S}$SYNC scheduler, the same result holds at least under the assumption of rigid moves [120].

The GATHERING Problem

In the following, we discuss literature about the GATHERING problem and the two related problems CONVERGENCE and NEAR-GATHERING. CONVERGENCE does not demand that the robots finally reach the same point but only requires them to converge towards the same point. NEAR-GATHERING considers robots with limited visibility and demands the robots to keep unique positions and to move such that every robot can observe the entire swarm. In other words, NEAR-GATHERING requires the robots to gather in a small area while keeping unique positions.

Possibilities & Impossibilities. In the context of robots with *unlimited visibility*, GATHERING can be solved in the $\mathcal{OBLOT}^F$ model by disoriented robots without multiplicity detection [36]. Robots without multiplicity detection cannot determine whether a position is occupied by one or multiple robots. Under the same assumptions, GATHERING is impossible in the less synchronized $\mathcal{OBLOT}^S$ and $\mathcal{OBLOT}^A$ models [110]. Multiplicity detection plays a crucial role: at least three disoriented robots with multiplicity detection can be gathered in the $\mathcal{OBLOT}^A$ model (and thus also in $\mathcal{OBLOT}^S$) [35]. The case of two robots remains impossible [114]. Besides multiplicity detection, an agreement on one axis of the local coordinate systems also allows the robots to solve GATHERING in $\mathcal{OBLOT}^A$ [10]. CONVERGENCE requires less assumptions than GATHERING. No multiplicity detection is needed in the $\mathcal{OBLOT}^A$ model [36].

Under the assumption of *limited visibility*, disoriented robots without multiplicity detection can be gathered under the $\mathcal{F}$SYNC scheduler ($\mathcal{OBLOT}^F$) [5] with the GO-TO-THE-CENTER (GTC) protocol that moves every robot towards the center of the smallest circle enclosing its neighborhood. GTC has also been generalized to three dimensions [22]. Under the $\mathcal{A}$SYNC scheduler ($\mathcal{OBLOT}^A$), current solutions require more capabilities: GATHERING can be achieved by robots with limited visibility that agree additionally on the axes and orientation of their local coordinate systems [68]. It is open whether fewer assumptions are sufficient to solve GATHERING of robots with limited visibility under the $\mathcal{S}$SYNC and $\mathcal{A}$SYNC scheduler ($\mathcal{OBLOT}^S$ and $\mathcal{OBLOT}^A$). In $\mathcal{OBLOT}^S$, CONVERGENCE can be solved even by disoriented robots with limited visibility without multiplicity detection [5]. However, similar to GATHERING, it is still open whether disoriented robots with

limited visibility can solve CONVERGENCE under the $\mathcal{A}$SYNC scheduler. Recently, it could be shown that multiplicity detection suffices to solve CONVERGENCE under the more restricted k-$\mathcal{A}$SYNC scheduler. The constant k bounds how often other robots can be activated within one LCM cycle of a single robot [85, 87].

The NEAR-GATHERING problem has been introduced in [104, 105] together with a protocol to solve NEAR-GATHERING by robots with limited visibility and agreement on one axis of their local coordinate systems under the $\mathcal{A}$SYNC scheduler ($\mathcal{OBLOT}^A$). An important tool to prevent collisions is a well-connected initial configuration, i.e., the initial configuration is connected concerning the *connectivity range* which is by an additive constant smaller than the viewing range [104, 105]. In earlier work, NEAR-GATHERING has been used as a subroutine to solve ARBITRARY PATTERN FORMATION by robots with limited visibility [120]. The solution, however, uses infinite persistent memory for each robot. Further research directions study GATHERING and CONVERGENCE under crash faults or Byzantine faults [3, 7, 11, 14, 15, 16, 19, 47, 78, 106] or inaccurate measurement and movement sensors of the robots [37, 80, 87].

Runtime. Considering disoriented robots with *unlimited* visibility, it is known that CONVERGENCE can be solved in $\mathcal{O}(n \cdot \log \Delta/\varepsilon)$ epochs under the $\mathcal{A}$SYNC scheduler, where the diameter Δ denotes the initial maximum distance of two robots [39] and ε is a convergence parameter (initially a bound of $\mathcal{O}(n^2 \cdot \log \Delta/\varepsilon)$ has been proven in [36]). When considering disoriented robots with *limited* visibility and the $\mathcal{F}$SYNC scheduler, the GTC protocol solves GATHERING both in two and three dimensions in $\Theta(n+\Delta^2)$ rounds [22, 50]. It is conjectured that the runtime is optimal in worst-case instances, where $\Delta \in \Omega(n)$ [22, 28]. A similar result could be achieved under the $\mathcal{OBLOT}^F$ model for robots that are located on a two-dimensional grid [24]. The protocol gathers the robots in $\mathcal{O}(n^2)$ rounds.

Faster GATHERING protocols could only be achieved by assuming agreement on one or two axes of the local coordinate systems or considering the $\mathcal{LUMI}$ model in combination with robots located on a two-dimensional grid. In [109], a universally optimal protocol with runtime $\Theta(\Delta)$ for robots in the Euclidean plane assuming one-axis agreement in the $\mathcal{OBLOT}$ model is introduced. Beyond the agreement on one axis, the robots have a viewing range of $\sqrt{10}$ and a connectivity range of 1. Notably, this protocol also works under the $\mathcal{A}$SYNC scheduler and hence, the robots fulfill the properties of the $\mathcal{OBLOT}^A_{\sqrt{10}}$ model.

Assuming disoriented robots, the protocols that achieve a runtime of $o(n^2)$ assume robots that are located on a two-dimensional grid and with visible states (which can be interpreted as robots in the $\mathcal{LUMI}$ model): there exist two protocols having an asymptotically optimal runtime of $\mathcal{O}(n)$; one protocol for *closed chains* [2] and another one in the standard connectivity model [40]. Both protocols use the idea of run sequences, initially introduced in [91]. A run sequence is a locally sequential movement of the robots realized with lights. To start a run sequence, locally asymmetric robots of the swarm are identified (on the grid such robots always exist – except for the final configuration). The protocol for closed chains considers the $\mathcal{LUMI}^F_{19}$ and the other protocol is designed for the $\mathcal{LUMI}^F_{11}$ model.

Also, a different time model – the continuous time model – leads to a faster runtime: there are protocols with a runtime of $\mathcal{O}(n)$ in the $\mathcal{OBLOT}^C$ model [20, 49]. The MOB protocol [49] even has a runtime of $\mathcal{O}(OPT \cdot \log(OPT))$, where OPT denotes the optimal time a protocol for robots with unlimited visibility would require. In [97], a more general class of continuous protocols has been introduced, the *contracting* protocols. Contracting protocols demand that each robot part of the global convex hull of all robots' positions moves at full speed towards the inside. Any contracting protocol gathers all robots in time $\mathcal{O}(n \cdot \Delta)$. One such protocol also needs a runtime of $\Omega(n \cdot \Delta)$ in a specific configuration. For instance, the continuous variant of GTC is contracting [97] but also the protocols of [20, 49].

For the NEAR-GATHERING problem, we introduced a class of protocols for the $\mathcal{OBLOT}^S_{1+\tau}$ ($\tau > 0$) model that solves the problem in $\Theta(\Delta^2)$ rounds. A closely related variant of the NEAR-GATHERING problem, called *collision-less* GATHERING, where robots are only allowed to collide

(move to the same position) at the final gathering point has been studied in [97] for robots in the $\mathcal{OBLOT}^C$ model. For $d = 1$, a collision-less protocol has been invented. For $d = 2$, the GO-TO-THE-GABRIEL-CENTER protocol, where robots move to the smallest enclosing circle of all their neighbors in the Gabriel graph (i.e., two robots are connected if the circle between their positions does not contain any other robot) has been proposed and conjectured that it is in almost all input configurations collision-less. Under a slightly stronger robot model, $\mathcal{LUMI}^C$ with a common chirality, a collision-less protocol with a runtime of $\mathcal{O}(n^2)$ has been designed in [96].

The CHAIN-FORMATION Problem.

The CHAIN-FORMATION problem has been initially introduced by [38] for robots located on a line (one-dimensional configurations). For the GO-TO-THE-MIDDLE (GTM) protocol that moves inner robots to the midpoint between their neighbors, it has been shown that it needs $\mathcal{O}(n^2 \cdot \log(n/\varepsilon))$ rounds to converge to the optimal configuration in the $\mathcal{OBLOT}^F$ model. Assuming $\mathcal{OBLOT}^S$, a runtime of $\mathcal{O}(n^5 \cdot \log(n/\varepsilon))$ epochs has been proven [38]. Recently, the $\mathcal{OBLOT}^S$ bound has been improved to $\mathcal{O}(n^2 \cdot \log(n/\varepsilon))$ epochs [25]. The GTM protocol has been transferred to two-dimensional configurations [60] with the same upper runtime bound in the $\mathcal{OBLOT}^F$ and $\mathcal{OBLOT}^S$ models [25, 60]. A close lower bound of $\Omega(n^2 \cdot \log(1/\varepsilon))$ rounds for a large class of protocols (including GTM) in the $\mathcal{OBLOT}^F$ model was proven in [88]. By considering locally visible states (comparable to the $\mathcal{LUMI}$ model) it was possible to design the HOPPER protocol – a linear time protocol that achieves a $\sqrt{2}$-approximation of the optimal configuration (also under the $\mathcal{F}$SYNC scheduler) [91]. The protocol does not converge to the optimal configuration. Additionally, the HOPPER protocol assumes that the two endpoints of the chain can be distinguished (they behave differently).

In [20], a variant of the GTM protocol has been designed to optimize the number of rounds as well as the maximal distance the robots have to move: the *δ-bounded* GTM protocol. The difference to the original GTM protocol is that the movement of each robot per round is at most δ. Still, the robots move towards the midpoint of their neighbors and either reach the point or are stopped after a distance of δ. The authors prove that the choice of $\delta \in \Theta(1/n)$ optimizes both the runtime and the maximum traveled distance of a robot. Moreover, the limit $\delta \to 0$ yields the continuous GTM protocol in the $\mathcal{OBLOT}^C_1$ model. The continuous GTM protocol solves the CHAIN-FORMATION problem in time $\Theta(n)$ [20].

In the same model ($\mathcal{OBLOT}^C_1$) the MOVE-ON-BISECTOR (MOB) protocol has been designed that solves the CHAIN-FORMATION problem also in time $\Theta(n)$ [49]. Instead of moving to the midpoints between their neighbors (GTM), robots follow with full speed the angle bisector between vectors pointing to their direct neighbors. Interestingly, for the MOB protocol also a competitive bound has been derived. The MOB protocol solves CHAIN-FORMATION in time $\mathcal{O}(OPT \cdot \log n)$, where OPT denotes the time an optimal protocol that has access to global visibility would require. Hence, the price of locality, i.e., the additional time protocols with local visibility require is at most a factor of $\log(n)$ in the $\mathcal{OBLOT}^C_1$ model.

Further research considers a more dynamic approach, where one of the outer robots is denoted as an *explorer*. The explorer moves through the plane and the goal is to make the entire chain follow the explorer using as few robots as possible. Different protocols exist for this setting, some require more powerful robots [60]. Furthermore, [46] studies the impact of crash faults on the GO-TO-THE-CENTER protocol (a protocol originally designed for the GATHERING problem) in one dimension. If exactly two robots crash, the authors observe a behavior similar to the GO-TO-THE-MIDDLE protocol for CHAIN-FORMATION: the robots arrange themselves equidistant between the two crashed robots.

Expanding Formation Problems

As previously discussed, the most prominent expanding formation problem is the UNIFORM-CIRCLE-FORMATION problem as the uniform circle can be (potentially) formed out of any input

configuration due to its high symmetry. We first discuss results considering robots with unlimited visibility. Early approaches showed that the problem can be solved by robots with persistent memory under the $\mathcal{S}$SYNC scheduler [114]. Later, a protocol for the $\mathcal{OBLOT}^S$ model was proposed that converges to the uniform circle [48]. Moreover, a combination of the protocols [54, 55, 84] yields a protocol to solve the UNIFORM-CIRCLE-FORMATION problem starting in every input configuration in the $\mathcal{OBLOT}^S$ model. Assuming $n \neq 4$, the protocol in [84] forms a biangular configuration, i.e., a configuration where robots are located on a circle but form two different alternating central angles. The protocol [54] forms a uniform circle starting in a biangular configuration and [55] solves the remaining case $n = 4$. Finally, the breakthrough result [67] in combination with the special case of $n = 4$ [100] showed that the UNIFORM-CIRCLE-FORMATION formation problem can be solved in the $\mathcal{OBLOT}^A$ model by *disoriented* robots. The core idea of the protocol [67] is quite simple: to move the robots to the boundary of the smallest enclosing circle and distribute them evenly afterward. However, due to the asynchrony lots of challenges have to be resolved, especially avoiding collisions (i.e., robots moving to the same position) or the avoidance of cyclic patterns in the robot's movement.

There is much less known about robots with limited visibility to solve the UNIFORM-CIRCLE-FORMATION problems. [58, 102] study the problems to form a uniform circle by *fat* robots, i.e., the robots are modeled as disks and thus, have an extent. Both works show that the problem is solvable by asynchronous robots; however, the robots agree on a common coordinate system. This way, the robots can first of all move towards the global origin that serves as the center of the circle to form. A further variant shows how to form *multiple* uniform circles under the same assumptions [12]. Our work in [26] shows that a combination of NEAR-GATHERING protocols with the protocol of [67] allows solving the UNIFORM-CIRCLE-FORMATION problem in the $\mathcal{OBLOT}^S_{1+\tau}$ ($\tau > 0$ is a constant) model by disoriented robots with limited visibility. The robots collect themselves first at distinct positions in a small area and form a circle at a small scale afterward.

Furthermore, the SCATTERING problem has a similar flaw as expanding formation problems. Here, multiple robots can occupy the same position in the initial configuration and have to spread out such that each robot occupies a unique position finally [9, 18, 79, 107, 108]. Naturally, such protocols require randomness as robots with the same position and the same view of the world cannot move to distinct locations deterministically.

Finally, we would like to mention works that study the *deployment* of robots in a certain area. Deployment has various meanings in the literature. One example is the formation of grid structures. In [92] the goal is to arrange the robots (in a model comparable to $\mathcal{OBLOT}$) in a triangular grid. The authors prove that their protocol converges but the resulting grid structure might contain holes. The results have also been extended to the three-dimensional Euclidean space [93]. Further work typically studies (much) more powerful robots, e.g., [62, 77].

To conclude, expanding formation problems are far less understood yet (compared to contraction problems), especially for natural models assuming robots with limited visibility. Certainly, with limited visibility, the task of spreading out seems more difficult at first sight as the robots do not observe what others do. However, as we will see in two examples in this thesis, the challenges to solve are quite similar to the ones of contraction problems (although certainly, new challenges occur).

Contracting Problems

4. GATHERING in the $\mathcal{OBLOT}$ Model

In this chapter, we study the GATHERING problem of n disoriented robots with limited visibility in the $\mathcal{OBLOT}_1^C$ and $\mathcal{OBLOT}_1^F$ models. For both time models (the $\mathcal{F}$SYNC scheduler and the continuous time model), we study entire classes of protocols and analyze their runtime. The class of protocols in the continuous time model is called *contracting protocols*. The criterion of contracting protocols is elegant, simple, easy to verify and many known protocols fulfill the property: A protocol is called contracting if robots that are vertices of the global convex hull of all robots' positions move with a speed of 1 inside the global closed convex hull [96]. We observe that the criterion naturally generalizes to robots located in Euclidean spaces of any dimension d and prove a runtime bound of $\mathcal{O}(n^{\log(d)} \cdot \Delta)$ for every contracting protocol.

In the case of the $\mathcal{OBLOT}_1^F$ model, we observe that an analogous criterion does not exist. Instead, we present λ-*contracting* protocols with help of a local criterion based on the neighborhood of a robot and show that if a robot moves sufficiently far (we derive later what this exactly means) inside the local convex hull enclosing its neighborhood, GATHERING can be achieved within $\Theta(\Delta^2)$ rounds. Notably, the number of rounds solely depends on the diameter Δ of the initial configuration and neither on the number of robots nor on the dimension d.

Section 4.3 considers contracting protocols for the $\mathcal{OBLOT}^C$ model. While the two-dimensional case was studied in [96], we consider $d \geq 3$ in this chapter. Notably, the case $d = 3$ relies on ideas that have been derived in the Master's thesis of Michael Braun [22] (the analysis itself is not contained in the thesis). In Section 4.3, we present the analysis for $d = 3$ that is based on the following publication. Afterward, we study $d > 3$ (not published yet) in Section 4.3.2.

> **2020** (with M. Braun and F. Meyer auf der Heide) "Local Gathering of Mobile Robots in Three Dimensions" In: *Proceedings of the 2020 International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, Best Student Paper Award, cf. [22].

In Section 4.4, we study λ-contracting protocols for the $\mathcal{OBLOT}^F$ model based on:

> **2022** (with J. Harbig, D. Jung, P. Kling, T. Knollmann and F. Meyer auf der Heide) "A Unifying Approach to Efficient (Near-)Gathering of Disoriented Robots with Limited Visibility" In: *Proceedings of the 26th International Conference on Principles of Distributed Systems (OPODIS)*, cf. [26].

4.1 Contribution

Our first contribution considers disoriented robots in the $\mathcal{OBLOT}_1^C$ model located in a Euclidean space of arbitrary dimension d. In [96], the class of *contracting protocols* for $d = 2$ has been presented. A continuous robot formation protocol is called contracting if robots that are located on vertices of the global convex hull of all robots' positions move with a speed of 1 inside the closed global convex hull. Typical protocols have this property, such as the continuous variant of the GTC protocol [96]. The authors of [96] prove that every contracting protocol gathers n robots located in $\mathbb{R}^2$ in time $\mathcal{O}(n \cdot \Delta)$, where Δ is the Euclidean diameter of the initial configuration. Moreover, there is at least one contracting protocol that needs a time of $\Omega(n \cdot \Delta)$ to gather all robots when the robots are initially placed on a regular polygon with a side length of 1 (equal to the viewing range) [96].

We show that the class naturally generalizes to higher dimensions (as convex hulls are defined analogously in higher dimensions) and prove an upper runtime bound of $\mathcal{O}(n^{H_d} \cdot \Delta)$, where $H_d \in \Theta(\log(d))$ represents the d-th Harmonic number. To familiarize ourselves with the concepts and definitions, we first study the case $d = 3$ in Section 4.3.1. The result has been published in [22]; in this chapter, we present a simplified and streamlined proof. The analysis projects the original three-dimensional configuration onto a two-dimensional projection plane and proves that the length of the boundary of the projected global convex hull (in the following denoted as the length of the convex hull) is monotonically decreasing at a rate in the order of $\Omega(1/\sqrt{n})$. Note that the projection plane must be chosen carefully, as otherwise, some robots would not move at all in the projection plane (if the plane is orthogonal to their velocity vector). Hence, we aim to find a projection plane in which all robots still move sufficiently fast to reduce the analysis of the original three-dimensional contracting protocol to the analysis of its behavior in a two-dimensional projection plane. We prove an upper time bound of $\mathcal{O}(n^{3/2} \cdot \Delta)$ for $d = 3$. Afterward, we study the case $d > 3$ in Section 4.3.2. The main analysis idea remains the same: to find a two-dimensional projection plane in which all robots move at a certain speed larger than 0 to analyze how the length of the projected global convex hull changes. To do so, we use a series of projections: we first project onto a $d-1$-dimensional plane, afterward onto a $d-2$-dimensional plane and so on until reaching a two-dimensional plane. The projection from dimension i to $i-1$ ensures that the length of the projected velocity vectors only decreases by a factor of $\mathcal{O}(1/\sqrt[i-1]{n})$. Thereby, we find a two-dimensional projection plane in which all velocity vectors still have a length of at least $n^{-\sum_{i=2}^{d-1} \frac{1}{i}}$. Based on this value, we prove that the length of the projected convex hull in this plane decreases with a speed of at least $n^{-H_{d-1}}$. As the initial length of the projected convex hull can be upper bounded by $\Delta \cdot \pi$, we derive a runtime of $\mathcal{O}(n^{H_d} \cdot \Delta)$. The result is summarized in the following Theorem 4.1.

Theorem 4.1 Every contracting protocol gathers a swarm of n disoriented robots located in $\mathbb{R}^d$ in the $\mathcal{OBLOT}_1^C$ model in time $\mathcal{O}(n^{\log(d)} \cdot \Delta)$.

Our second contribution introduces a large class of GATHERING protocols for robots in the $\mathcal{OBLOT}^F$ model that contains several natural protocols such as GTC [5]. We prove that *every* protocol from this class gathers in $\mathcal{O}(\Delta^2)$ rounds, where the diameter Δ denotes the initial maximal distance between two robots. Note that, the bound of $\mathcal{O}\left(\Delta^2\right)$ not only reflects how far a given initial swarm is from a gathering but also improves the GTC bound from $\mathcal{O}\left(n+\Delta^2\right)$[50] to $\mathcal{O}\left(\Delta^2\right)$. We call this class λ*-contracting protocols*. Such protocols restrict the allowed target points to a specific subset of a robot's local convex hull (formed by the positions of all visible robots, including itself) in the following way. Let *diam* denote the diameter of a robot's local convex hull. Then, a target point p is an allowed target point if it is the center of a line segment of length $\lambda \cdot diam$, completely contained in the local convex hull. This guarantees that the target point lies far enough inside the local convex hull (at least along one dimension) to decrease the swarm's diameter sufficiently. See Figure 4.1 for an illustration. We believe that these λ-contracting protocols encapsulate the core property of fast GATHERING protocols. Their analysis is comparatively clean, simple, and holds for

any dimension d. Thus, by proving that (the generalization of) GTC is λ-contracting for arbitrary dimensions, we give the first protocol that provably gathers in $\mathcal{O}(\Delta^2)$ rounds for any dimension.

Theorem 4.2 Every λ-contracting protocol gathers a swarm of n disoriented robots located in $\mathbb{R}^d$ in the $\mathcal{OBLOT}_1^F$ model in $\Theta(\Delta^2)$ rounds.

As a strong indicator that our protocol class might be asymptotically optimal, we prove that every GATHERING protocol for deterministic, disoriented robots whose target points lie *always inside* the robots' local convex hulls requires $\Omega(\Delta^2)$ rounds to gather all robots if the robots are initially located on the vertices of a regular polygon with a side length of 1. Staying in the convex hull of visible robots is a natural property for any known protocol designed for oblivious, disoriented robots with limited visibility. Thus, reaching a sub-quadratic runtime – if at all possible – would require the robots to compute target points outside of their local convex hulls sufficiently often.

Theorem 4.3 For every protocol for robots in the $\mathcal{OBLOT}_1^F$ model that computes the target point of a robot always inside of the local convex hull of all visible robots, there exists an initial configuration where the protocol requires $\Omega(\Delta^2)$ rounds to gather all robots.

4.2 Model Recap and Preliminaries

We study the $\mathcal{OBLOT}_1^C$ and the $\mathcal{OBLOT}_1^F$ models. For the $\mathcal{OBLOT}_1^C$ we analyze the class of contracting protocols. The class of protocols in the $\mathcal{OBLOT}_1^F$ model is denoted as λ-contracting protocols. We summarize the most important model features in Table 5.2, and more details can be found in Chapter 2.

Protocol	Time	Dimension	Viewing Range	Orientation	Chain
contracting protocols	$\mathcal{F}$SYNC	≥ 1	1	disoriented	no
λ-contracting protocols	continuous time model	≥ 1	1	disoriented	no

Table 4.1: A summary of the most important model details for this chapter.

Problem Statement. The goal of the GATHERING problem is to collect all robots on the same (not predefined) point. We say that a formation protocol (discrete or continuous) solves the GATHERING problem, if there is a point in time t such that $p_i(t) = p_j(t)$ for all pairs of indices $i, j \in \{0, \ldots, n-1\}$.

Definitions & Notation. We denote the closed convex hull of all robots' positions at time t by $CH(t)$. Observe that the definition is global and robots cannot observe $CH(t)$ due to their limited visibility. Locally, we define the *local convex hull* $CH_i(t)$ of a robot r_i as the closed convex hull of all robots' positions in $N_i(t)$, i.e., the convex hull of all visible robots of r_i (including itself) at time t. Moreover, $\Delta(t)$ denotes the Euclidean diameter of all robots' positions, i.e., the maximum distance of any pair of robots, at time t. Throughout the thesis, we use $\Delta := \Delta(0)$ as the Euclidean diameter of the initial configuration. Locally, for a robot r_i and time t, $\Delta_i(t)$ is the Euclidean diameter of all robots in $N_i(t)$.

4.3 Continuous Time GATHERING

In the following, we define the class of contracting protocols based on [89, 96], where two-dimensional protocols are studied. Since convex hulls are analogously defined in higher dimensions, we can define the class of contracting protocols independent of the environment where the robots live in. Informally, a continuous robot formation protocol is contracting if robots that are located on vertices of the global closed convex hull move with a speed of 1 inside the global closed convex hull.

Definition 4.1 A continuous robot formation protocol is called *(globally) contracting* if for every point in time t and every robot r_i whose position is a vertex of $CH(t)$, it holds that $\text{target}_i^{\mathcal{P}}(t) \in CH(t)$ and $s_i^{\mathcal{P}}(t) = 1$.

Observe that Definition 4.1 refers to the global convex hull of all robots' positions. However, due to the limited visibility of the robots, we typically look at *local* protocols. Such protocols define the target point of a robot based on its local neighborhood. Nevertheless, typical target points such as the center of the smallest enclosing circle of a robot's neighborhood lie inside the global convex hull and thus fulfill – at first sight – the criterion of being contracting. Still, target points might lie inside the global convex hull while the movements of the robots do not maintain connectivity of $\text{UBG}(t)$. Consider for instance the case that a protocol divides the configuration into two connected components. The robots of the connected components might gather at two different points. Afterward, the robots are not aware that there is a second position occupied by robots and remain in their current location. As a result, the protocol is not contracting at this point since the robots do not move anymore although they have not yet gathered at a single point. With this observation, we see that maintaining connectivity is an essential property of local protocols and define the following class of *locally contracting* protocols.

Definition 4.2 A continuous robot formation protocol is called *locally contracting* if for every point in time t,

1. $\text{UBG}(t)$ is connected
2. for every robot r_i whose position is a vertex of $CH_i(t)$, $\text{target}_i^{\mathcal{P}}(t) \in CH_i(t)$ and $s_i^{\mathcal{P}}(t) = 1$.

Informally speaking, a locally contracting protocol must maintain the connectivity of $\text{UBG}(t)$ and robots that are located on vertices of their *local* closed convex hull move with a speed of 1 inside. Since local convex hulls are completely contained in the global convex hull, every locally contracting protocol is also (globally) contracting.

Lemma 4.1 Every locally contracting protocol is globally contracting.

As a consequence, we focus on the analysis of globally contracting protocols in the following sections. For concrete protocols, we have to prove that they maintain the connectivity of $\text{UBG}(t)$ and that robots compute target points inside their local convex hulls (which is often easy to see).

4.3.1 Analysis of Three-dimensional Contracting Protocols

The main tool we use for the analysis of high-dimensional contracting protocols is an orthogonal projection of the robots' positions and velocity vectors onto a two-dimensional plane. In this section, we consider the three-dimensional case to make the reader familiar with the core ideas and notation. Let $h(\vec{x})$ be the plane through the origin with normal vector $\vec{x}$ and let $P_{\vec{x}}$ denote the orthogonal projection onto $h(\vec{x})$. Recall that we denote by a *configuration* at time t, the set containing all robots' positions at time t. Now, given a configuration C of n robots, consider their projection $C_{\vec{x}} = \{P_{\vec{x}} \cdot p_i(t) | p_i(t) \in C\}$ onto $h(\vec{x})$ along with the projections of their velocity vectors $v_i^{\mathcal{P}}(t,\vec{x}) = P_{\vec{x}} \cdot v_i^{\mathcal{P}}(t)$. Furthermore, we denote the convex hull of $C_{\vec{x}}$ as $PCH_t(\vec{x})$. If the robots perform a contracting protocol in the three-dimensional space, their projections also move towards the inside of the projected convex hull $PCH_t(\vec{x})$ since $P_{\vec{x}}$ is a linear transformation and therefore

preserves convexity. However, the lengths of the projected velocity vectors $v_i^{\mathcal{P}}(t,\vec{x})$ are smaller than 1 in general. For a given projection onto a plane $h(\vec{x})$, the minimum length of the $v_i^{\mathcal{P}}(t,\vec{x})$ will be called the *projected speed* and is denoted by $\varepsilon_{\vec{x}} = \min_i \|v_i^{\mathcal{P}}(t,\vec{x})\|_2$. Note that $\varepsilon_{\vec{x}}$ can even be 0 in case $h(\vec{x})$ is orthogonal to any velocity vector. The following notion of the length of $PCH_t(\vec{x})$ will be used as a progress measure for three-dimensional contracting protocols.

Definition 4.3 Let $c_1(t), c_2(t), ..., c_{k(t)}(t)$ be the vertices of $PCH_t(\vec{x})$ (ordered counter-clockwise), where $k(t)$ is the number of vertices at time t. The *length* $\ell(t,\vec{x})$ of $PCH_t(\vec{x})$ is defined as the sum of its edge lengths: $\ell(t,\vec{x}) = \sum_{\iota=1}^{k(t)} \|c_\iota(t), c_{\iota-1}(t)\|_2$, where $c_0 := c_{k(t)}(t)$.

Observe that $\ell(t,\vec{x}) \leq \Delta \cdot \pi$. The upper bound can be reached for large n if $PCH_t(\vec{x})$ is a regular n-gon. If $\ell(t,\vec{x}) = 0$, the robots either have gathered in the original three-dimensional Euclidean space or form a line parallel to $\vec{x}$. In the latter scenario, it will only take a maximum of $\mathcal{O}(\Delta)$ additional time for the robots to gather, as the robots at the endpoints of the line must move with a speed of 1 towards each other. Later on, we choose a plane with a projected speed as large as possible and analyze $\ell'(t,\vec{x})$, the rate at which the length of the projected convex hull decreases. To do so, we make use of a lemma that states how the distance between two robots changes based on their velocity vectors.

Lemma 4.2 — [49]. Consider two robots r_i and r_j and let $d_{i,j}(t) : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ represent their distance at time t. Furthermore, we assume that the robots execute the continuous robot formation protocol $\mathcal{P}$. The distance between r_i and r_j changes with speed

$$d_{i,j}{}'(t) = -(s_i^{\mathcal{P}}(t) \cdot \cos(\beta_{i,j}(t)) + s_j^{\mathcal{P}}(t) \cdot \cos(\beta_{j,i}(t))),$$

where $\beta_{i,j}(t)$ is the angle between $v_i^{\mathcal{P}}(t)$ and the line segment connecting r_i and r_j at time t.

With help of Lemma 4.2, we prove a bound on $\ell'(t,\vec{x})$ depending on $\varepsilon_{\vec{x}}$.

Lemma 4.3 Consider a plane $h(\vec{x})$ with projected speed $\varepsilon_{\vec{x}}$ at time t, such that $\ell(t,\vec{x}) > 0$ and no two robots with different positions in $\mathbb{R}^3$ get projected onto the same point on $h(\vec{x})$. Then $\ell'(t,\vec{x}) \leq -\frac{8\varepsilon_{\vec{x}}}{n}$.

Proof. Because $P_{\vec{x}}$ is a linear transformation, each of the $c_\iota(t)$ (see Definition 4.3) is also the projection of one of the vertices of the original, three-dimensional convex hull $CH(t)$. Therefore, the velocity vectors of the corresponding robots point towards the inside of $CH(t)$ by the definition of contracting protocols. Now, consider the projections of these velocity vectors onto $h(\vec{x})$. By our assumption, $\|v_i^{\mathcal{P}}(t,\vec{x})\|_2 \geq \varepsilon_{\vec{x}}$, for all i. Furthermore, let $\alpha_\iota(t)$ be the internal angle of $PCH_t(\vec{x})$ at $c_\iota(t)$. We know (by our assumption) that each corner $c_\iota(t)$ of $PCH_t(\vec{x})$ contains only a single robot. Hence, each $\alpha_\iota(t)$ is split into two parts, $\hat{\beta}_{\iota,\iota-1}(t)$ and $\hat{\beta}_{\iota-1,\iota}(t)$ by $c_\iota(t)$'s velocity vector $v_\iota^{\mathcal{P}}(t,\vec{x})$, such that $\alpha_\iota(t) = \hat{\beta}_{\iota,\iota-1}(t) + \hat{\beta}_{\iota-1,\iota}(t)$. Using Lemma 4.2, the derivative of $\ell(t,\vec{x})$ can now be bounded as follows: Recall that $\ell(t,\vec{x}) = \sum_{\iota=1}^{k(t)} \|c_\iota(t), c_{\iota-1}(t)\|_2$:

$$\begin{aligned}
\ell'(t,\vec{x}) &= \sum_{\iota=1}^{k(t)} -\left(\|v_\iota^{\mathcal{P}}(t,\vec{x})\|_2 \cos\hat{\beta}_{\iota,\iota-1}(t) + \|v_{\iota-1}^{\mathcal{P}}(t,\vec{x})\|_2 \cos\hat{\beta}_{\iota-1,\iota}(t)\right) \\
&\leq \sum_{\iota=1}^{k(t)} -\left(\varepsilon_{\vec{x}} \cos\hat{\beta}_{\iota,\iota-1}(t) + \varepsilon_{\vec{x}} \cos\hat{\beta}_{\iota-1,\iota}(t)\right) \\
&= -\varepsilon_{\vec{x}} \sum_{\iota=1}^{k(t)} \cos\hat{\beta}_{\iota,\iota-1}(t) + \cos\hat{\beta}_{\iota-1,\iota}(t)
\end{aligned}$$

$$\leq -\varepsilon_{\vec{x}} \sum_{\iota=1}^{k(t)} \frac{2(\alpha_\iota(t)-\pi)^2}{\pi^2} \tag{4.1}$$
$$= -\frac{2\varepsilon_{\vec{x}}}{\pi^2} \sum_{\iota=1}^{k(t)} (\alpha_\iota(t)-\pi)^2$$

For Equation (4.1) observe that for $\vartheta \in [0,1]$ and $\alpha \in [0,\pi]$, it holds that $\cos(\alpha\vartheta)+\cos(\alpha(1-\vartheta)) \geq \frac{2(\alpha-\pi)^2}{\pi^2}$ [97]. Now, we upper bound the sum of squares by the squared sum and use the fact that the sum of the inner angles of a convex polygon with k corners is $(k-2)\cdot\pi$.

$$\begin{aligned}
\ell'(t,\vec{x}) &\leq -\frac{2\varepsilon_{\vec{x}}}{k(t)\cdot\pi^2} \cdot \Big(\sum_{\iota=1}^{k(t)} (\alpha_\iota(t)-\pi)\Big)^2 \\
&= -\frac{2\varepsilon_{\vec{x}}}{k(t)\cdot\pi^2} \cdot \big((k(t)-2)\cdot\pi - k(t)\cdot\pi\big)^2 \\
&= -\frac{2\varepsilon_{\vec{x}}}{k(t)\cdot\pi^2} \cdot (2\pi)^2 \\
&= -\frac{8\varepsilon_{\vec{x}}}{k(t)} \\
&\leq -\frac{8\varepsilon_{\vec{x}}}{n}
\end{aligned}$$

■

The next goal is to find a plane $h(\vec{x})$ such that $\varepsilon_{\vec{x}}$ is large. Note that the length of $\vec{x}$ does not matter. Hence, we look at all possible vectors $\vec{x}$ of length 1. Those vectors form the unit sphere U. Now, consider a projection plane $h(\vec{x})$. If this plane has projected speed smaller than ε at time t, then there is a velocity vector $v_i^{\mathcal{P}}(t)$, such that $\angle(\vec{x}, v_i^{\mathcal{P}}(t)) < \sin^{-1}\varepsilon$. We say that $v_i^{\mathcal{P}}(t)$ *blocks* $h(\vec{x})$. Conversely, given a $v_i^{\mathcal{P}}(t)$, we can determine the set of all the $h(\vec{x})$ that are blocked by $v_i^{\mathcal{P}}(t)$:

Lemma 4.4 At time t, the velocity vector $v_i^{\mathcal{P}}(t)$ blocks vectors from a surface area of $2\pi\big(1-\sqrt{1-\varepsilon^2}\big)$ on U from reaching projected speed ε.

Proof. Consider the spherical cap of U with base radius ε, apex $v_i^{\mathcal{P}}(t)$ and colatitude angle θ. Further, let S be its curved surface. For all vectors $\vec{x} \in S$, $h(\vec{x})$ is blocked from reaching projected speed ε. The area of S can be computed by $A_S = 2\pi r^2(1-\cos\theta) = 2\pi(1-\cos(\sin^{-1}\varepsilon)) = 2\pi(1-\sqrt{1-\varepsilon^2})$. ■

Since there are n robots, the area blocked by their velocity vectors is at most $n\cdot 2\pi(1-\sqrt{1-\varepsilon^2})$, whereas the total surface area of U is 4π. Now, we choose ε such that at least one plane with a projected speed of at least ε remains.

Lemma 4.5 There exists a vector $\vec{x}$ on U with $\varepsilon_{\vec{x}} \geq \frac{1}{\sqrt{n}}$.

Proof. U has a surface area of 4π and the robots' velocity vectors block an area of at most $n\cdot 2\pi(1-\sqrt{1-\varepsilon^2})$. We want to choose ε such that the following holds:

$$4\pi > n\cdot 2\pi(1-\sqrt{1-\varepsilon^2}) \iff \varepsilon < 2\cdot\sqrt{\frac{n-1}{n^2}}.$$

Lastly, observe that for $n \geq 2$ it holds $\frac{1}{\sqrt{n}} < 2\cdot\sqrt{\frac{n-1}{n^2}}$ and the statement follows.

■

Theorem 4.4 A swarm of n robots located in $\mathbb{R}^3$ and following a contracting protocol $\mathcal{P}$, gathers in time $\mathcal{O}(n^{3/2} \cdot \Delta)$.

Proof. Via Lemma 4.5, we obtain a vector $\vec{x}$ such that the projected speed of the robot configuration on $h(\vec{x})$ is at least $\frac{1}{\sqrt{n}}$. With help of Lemma 4.3, we get a bound on $\ell'(t,\vec{x})$:

$$\begin{aligned}\ell'(t,\vec{x}) &\leq -\frac{8}{n} \cdot \frac{1}{\sqrt{n}} \\ &= -\frac{8}{n^{3/2}}.\end{aligned}$$

Since the initial length of $PCH_t(\vec{x})$ is upper bounded by $\Delta \cdot \pi$, we obtain a time bound of $\frac{\Delta \cdot \pi \cdot n^{3/2}}{8}$ until $\ell(t,\vec{x}) = 0$. Afterward, the robots are either gathered or form a line parallel to $\vec{x}$. In the latter case, the robots gather after further time $\Delta/2$. Hence, we derive a total time bound of $\frac{\Delta \cdot \pi \cdot n^{3/2}}{8} + \Delta/2 \in \mathcal{O}(n^{3/2} \cdot \Delta)$. ■

4.3.2 Analysis of Higher-dimensional Contracting Protocols

Next, we consider $d > 3$. We generalize the proof of Section 4.3.1. Instead of projecting only once, we use a series of orthogonal projections to project the original, high-dimensional configuration to a two-dimensional plane. For the analysis, we need some preliminary definitions to compute analogously to the three-dimensional case, the surface area of d-dimensional hyperspherical caps.

Definition 4.4 The d-sphere of radius r is defined as

$$S^d(r) := \{x \in \mathbb{R}^{d+1} \mid \|x\| = r\}.$$

Definition 4.5

1. The Beta function is defined as

$$B(a,b) = \int_0^1 t^{a-1} \cdot (1-t)^{b-1} dt.$$

2. The regularized incomplete Beta function is defined as

$$I_x(a,b) = \frac{1}{B(a,b)} \cdot \int_0^x t^{a-1} \cdot (1-t)^{b-1} dt.$$

With help of the Beta functions, we now state a formula for the surface area of d-dimensional hyperspherical caps.

Lemma 4.6 — [95]. The surface area of a d-dimensional hyperspherical cap $A_d^{cap}(r)$ of a sphere with radius r can be computed As

$$A_d^{cap}(r) = \frac{1}{2} \cdot A_d(r) \cdot I_{\sin^2(\theta)}\left(\frac{d-1}{2}, \frac{1}{2}\right)$$

where $A_d(r)$ is the surface area of a d-sphere with radius r, θ is the colatitude angle of the cap and I represents the regularized, incomplete Beta function.

The following bound on the regularized incomplete beta function is a useful tool to derive an upper bound on the surface area of hyperspherical caps.

Lemma 4.7 For $x \in (0,1)$ and $d \geq 2$, we obtain

$$I_{x^2}\left(\frac{d-1}{2}, \frac{1}{2}\right) \leq x^{d-1}.$$

Proof. First observe that $I_{x^2}(a,b)$ is for fixed $a \geq 1$ and $0 < x < 1$ monotonically increasing for $b \in [\frac{1}{2}, 1]$. Hence, for $d \geq 2$, $I_{x^2}\left(\frac{d-1}{2}, \frac{1}{2}\right) \leq I_{x^2}\left(\frac{d-1}{2}, 1\right)$. Applying the identify $I_x(a,1) = x^a$ yields

$$I_{x^2}\left(\frac{d-1}{2}, \frac{1}{2}\right) \leq I_{x^2}\left(\frac{d-1}{2}, 1\right) = x^{2\frac{d-1}{2}} = x^{d-1}.$$

■

The following lemma is the key lemma of the proof, stating that for every collection of n d-dimensional vectors all of length $\geq \varepsilon$, there exists a $d-1$-dimensional plane such that the vectors projected orthogonally to that plane all have a length of at least $\frac{\varepsilon}{\sqrt[d-1]{n}}$ in the projection.

Lemma 4.8 For every collection of n d-dimensional vectors $\vec{v}_1, \ldots, \vec{v}_n$ with $\|\vec{v}_i\|_2 \geq r$ for all i and $0 < r < 1$, there exists a $d-1$-dimensional projection plane $h(\vec{x})$ with normal vector $\vec{x}$ and a projected speed of at least $\frac{r}{\sqrt[d-1]{n}}$.

Proof. Similar to the three-dimensional analysis, we look at all possible normal vectors $\vec{x}$ of $d-1$-dimensional projection planes $h(\vec{x})$. In contrast to the three-dimensional case, we do not look at unit d-spheres but at spheres of a certain radius r since already after the first projection, the vectors have lost in length. Hence, we consider normal vectors of length r. Now, consider a projection plane $h(\vec{x})$. If this plane has projected speed smaller than $\varepsilon < r$ at time t, then there is a velocity vector $v_i^{\mathcal{P}}(t)$, such that $\angle(\vec{x}, v_i^{\mathcal{P}}(t)) < \cos^{-1}\left(\frac{\sqrt{r^2-\varepsilon^2}}{r}\right)$. Analogously, for a single velocity vector $v_i(t)$, the set of all blocked planes, i.e., the set of all planes in which $v_i^{\mathcal{P}}(t)$ has a length of less than ε, can be expressed as the surface area of a hyperspherical cap with apex $v_i^{\mathcal{P}}(t)$ and colatitude angle $\cos^{-1}\left(\frac{\sqrt{r^2-\varepsilon^2}}{r}\right)$. We want to choose ε such that the hyperspherical caps of all velocity vectors do not block the entire surface of the d-sphere with radius r. If not the entire surface is blocked, there is at least one $d-1$-dimensional plane with a projected speed of at least ε. Precisely, we aim to choose ε, such that

$$\begin{aligned} A_d(r) &> n \cdot \frac{1}{2} \cdot A_d(r) \cdot I_{\sin^2(\theta)}\left(\frac{d-1}{2}, \frac{1}{2}\right) \\ &= n \cdot \frac{1}{2} \cdot A_d(r) \cdot I_{\frac{\varepsilon^2}{r^2}}\left(\frac{d-1}{2}, \frac{1}{2}\right). \end{aligned}$$

Rearranging yields

$$n \cdot I_{\frac{\varepsilon^2}{r^2}}\left(\frac{d-1}{2}, \frac{1}{2}\right) < 2.$$

Now, we apply Lemma 4.7 to get an upper bound on ε.

$$n \cdot I_{\frac{\varepsilon^2}{r^2}}\left(\frac{d-1}{2}, \frac{1}{2}\right) \leq n \cdot \left(\frac{\varepsilon}{r}\right)^{d-1} < 2 \iff \varepsilon \leq \frac{\sqrt[d-1]{2} \cdot r}{\sqrt[d-1]{n}}.$$

Choosing $\varepsilon = \frac{r}{\sqrt[d-1]{n}}$ proves the claim. ■

Lemma 4.9 For every d-dimensional contracting protocol $\mathcal{P}$, there exists a two-dimensional projection plane with a projected speed of at least $n^{-\sum_{i=2}^{d-1} \frac{1}{i}}$.

Proof. We apply Lemma 4.8 inductively as follows. Initially, we take the n unit vectors $v_i^{\mathcal{P}}(t)$ and project them onto a $d-1$-dimensional plane with a projected speed of at least $\frac{1}{\sqrt[d-1]{n}}$ (Lemma 4.8). Afterward, we take the n projected vectors of length at least $\frac{1}{\sqrt[d-1]{n}}$ and project them onto a $d-2$-dimensional plane and so on. Via Lemma 4.8, we get that there is a series of $d-2$ projections such the projected speed of the resulting two-dimensional plane is at least

$$\frac{1}{\prod_{i=2}^{d-1} n^{\frac{1}{i}}} = \frac{1}{n^{\sum_{i=2}^{d-1} \frac{1}{i}}}.$$

■

Theorem 4.1 Every contracting protocol gathers a swarm of n disoriented robots located in $\mathbb{R}^d$ in the $\mathcal{OBLOT}_1^{\mathcal{C}}$ model in time $\mathcal{O}(n^{\log(d)} \cdot \Delta)$.

Proof. We need to combine the statements of Lemma 4.3 and Lemma 4.9: There exists a two-dimensional normal vector $\vec{x}$ with a projected speed of at least $n^{-\sum_{i=2}^{d-1} \frac{1}{i}}$ (Lemma 4.9). For the plane $h(\vec{x})$, we conclude with help of Lemma 4.3 that

$$\begin{aligned} \ell'(t, \vec{x}) &\leq -\frac{8}{n} \cdot n^{-\sum_{i=2}^{d-1} \frac{1}{i}} \\ &= -\frac{8}{n^{\sum_{i=1}^{d-1} \frac{1}{i}}} \\ &= -\frac{8}{n^{H_{d-1}}}. \end{aligned}$$

Since $\ell(0, \vec{x}) \leq \pi \cdot \Delta$, we conclude that after a time of $\frac{n^{H_{d-1}} \cdot \pi \cdot \Delta}{8} \in \mathcal{O}(n^{H_{d-1}} \cdot \Delta)$, the length of the projected convex hull in $h(\vec{x})$ is 0, i.e., $\ell(t, \vec{x}) = 0$. In this case, either the robots have gathered in the initial d-dimensional space or they have formed a line parallel to $\vec{x}$. The latter case leads to an additional time of $\mathcal{O}(\Delta)$ since the robots at the end of the line always move with a speed of 1 towards the midpoint of the line. We conclude that there is a point in time t_{final} with $t_{\text{final}} \in \mathcal{O}(n^{H_{d-1}} \cdot \Delta)$ such that all robots have gathered at time t_{final}.

■

4.3.3 An Exemplary Contracting Protocol

Next, a continuous version of the GTC protocol [5] will be considered as a concrete example of a contracting protocol. The two-dimensional version of this protocol was adapted for the continuous time model by [97]. In the discrete time version of the GTC protocol, connectivity is maintained with help of *limit circles*, i.e., robots sometimes do not move towards the center of the smallest enclosing circle of their neighborhood but stop earlier. Compared to the discrete time version, no

Algorithm 1 Continuous d-GTC

1: $\mathcal{R}_i(t) :=$ {positions of robots visible from r_i, including r_i at time t}
2: $\mathcal{S}_i(t) :=$ smallest enclosing hypersphere of $\mathcal{R}_i(t)$
3: $c_i(t) :=$ center of $\mathcal{S}_i(t)$
4: Move towards $c_i(t)$ with speed 1, or stay on $c_i(t)$ if r_i is already positioned on it.

additional measures have to be taken to preserve connectivity in the continuous time model, as it can be shown that this happens naturally. The protocol is summarized below.

To show that the continuous d-GTC protocol is contracting, it must first be verified that connectivity of UBG(t) is maintained. The same reasoning that was used in the two-dimensional case by [97] can also be applied here.

Lemma 4.10 Let $\mathcal{R}$ be a set of robots in $\mathbb{R}^d$ that follows the continuous d-GTC protocol. If $\{r_i, r_j\}$ is an edge in UBG(t) at time t, then $\{r_i, r_j\}$ is an edge in UBG(t') at $t' \geq t$. Thus, the continuous d-GTC protocol maintains the connectivity of UBG(t).

Proof. Consider a robot r_i with neighborhood $N_i(t)$ at time t. Let $Q_i(t)$ be the intersection of the unit balls of all robots in $N_i(t)$. The center of the smallest hypersphere of $N_i(t)$ can have a radius of at most 1 and contains all robots in $N_i(t)$. Hence, its center $c_i(t)$ must lie in $Q_i(t)$ [97].

Now, we consider some neighbor $r_j \in N_i(t)$ of r_i and we assume that there is some future point in time $t' > t$, such that the distance between r_i and r_j is larger than 1 at time t' (and hence, r_i and r_j are no longer neighbors). Since the movement of robots is continuous, there must be some time $t^* \in [t, t']$, for which $\|p_i(t^*), p_j(t^*)\|_2 = 1$. Now, let L denote the intersection of the unit balls of r_i and r_j at time t^*. Any point in L is within a distance of at most 1 of both r_i and r_j. Furthermore L is a superset of both $Q_i(t^*)$ and $Q_j(t^*)$, meaning the target points $c_i(t^*)$ and $c_j(t^*)$ of both r_i and r_j also lie in L. Therefore, r_i and r_j can only move in the direction of points that are at a distance of at most 1 from both of them, meaning their distance can never exceed 1, creating a contradiction to the assumption that their distance is greater than 1 at time t'. ■

As the center of a smallest enclosing hypersphere is always a convex combination of the points it encloses [61], we obtain that the continuous d-GTC protocol is a locally contracting protocol and thus, also globally contracting.

Theorem 4.5 The continuous d-GTC protocol is locally contracting and therefore gathers a swarm of n disoriented robots located in $\mathbb{R}^d$ with limited visibility in time $\mathcal{O}\left(n^{\log(d)} \cdot \Delta\right)$.

4.4 Discrete Time GATHERING

In this section, we study the GATHERING problem in the $\mathcal{OBLOT}^F_1$ model. More precisely, we describe the class of λ-contracting protocols – a class of protocols which solve GATHERING in $\Theta\left(\Delta^2\right)$ rounds. Furthermore, we derive a subclass of λ-contracting protocols, called (α, β)-contracting protocols. The class of (α, β)-contracting contracting protocols is a powerful tool to determine whether a given gathering protocol (such as GTC) fulfills the property of being λ-contracting. The first intuition to define a class of protocols to solve GATHERING would be to transfer the class of continuous contracting protocols (see Section 4.3) to the discrete LCM case. However, as we emphasize with some preliminary observations in Section 4.4.1, demanding to move inside the local or global convex hulls is not enough to solve GATHERING in the discrete case. Based on these observations, we derive the definition of λ-contracting protocols in Section 4.4.2. Afterward, we state and prove upper and lower runtime bounds in Section 4.4.3 and present the subclass of (α, β)-contracting protocols as well as concrete examples of protocols in Section 4.4.4.

4.4.1 Preliminary Observations

Before heading to the concrete definition of λ-contracting protocols, we make some preliminary observations. Based on these observations, we derive the definition of λ-contracting protocols. The first main observation is analogous to locally contracting protocols in the continuous time model: each protocol must always maintain connectivity of UBG(t). Since the robots neither agree on a common coordinate system nor any common direction it would be impossible to reconnect the robots deterministically.

Observation 4.1 Each discrete robot formation protocol to solve GATHERING must preserve the connectivity of UBG(t).

Furthermore, it must be ensured that robots with the same view, i.e., robots that observe the same set of robots, eventually compute the same target point. Consider, for instance, the case of $n = 2$ (also known as RENDEZVOUS). A discrete robot formation protocol that solves GATHERING does not need to compute the same target point in every round. However, there must be a condition such that at some point the two robots compute the same target point. The condition could, for example, be based on the distance: if the distance between the two robots is less than $1/2$, the robots move to their midpoint, otherwise, each robot moves a constant distance towards the other robot.

Observation 4.2 Each discrete robot formation protocol to solve GATHERING must – at some point – enforce robots with the same view to compute the same target points.

Moreover, it is important *which* points the robots compute. There are certainly invalid points either leading to no gathering at all or a very large number of rounds. The first observation is that robots that are located at the global boundary of the swarm have to move inside (while keeping connectivity). Since robots do have limited visibility, they cannot decide whether they are part of the global boundary. Hence, their decision must always be based on their local neighborhood. As a consequence, we state that robots that are vertices of the convex hull of their neighborhood or very close to the boundary of the convex hull should always move inside unless their movement would destroy the connectivity.

Observation 4.3 For a robot r_i, staying in its position should only be allowable if it is necessary to maintain connectivity or if the robot itself lies sufficiently far inside of its local convex hull.

Additionally, the robots should move sufficiently far inside the local convex hull to realize a fast runtime.

Observation 4.4 Each robot should move sufficiently far inside its local convex hull.

Lastly, robots should also compute target points that are sufficiently far away from other vertices of their local convex hull. Consider for instance the case that robots are always allowed to move to positions of their neighbors. In that case, one could design a protocol that does not lead to any progress started by robots located on the vertices of a regular polygon with a side length of 1.

Observation 4.5 Target points should lie sufficiently far away from *any* vertex of a local convex hull.

4.4.2 λ-contracting Protocols

Based on Observation 4.1, any discrete protocol to solve GATHERING must be *connectivity preserving*, i.e., it always maintains the connectivity of UBG(t). Next, we define the meaning of *sufficiently far* inside of local convex hulls (Observations 4.3 to 4.5) precisely, leading to a characterization of valid target points.

Definition 4.6 Let Q be a convex polytope with diameter $diam$ and $0 < \lambda \leq 1$ a constant. A point $p \in Q$ is called to be λ-*centered* if it is the midpoint of a line segment that is completely contained in Q and has a length of $\lambda \cdot diam$.

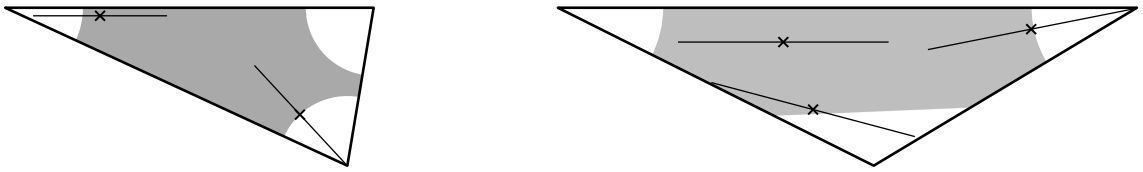

Figure 4.1: A visualization of λ-centered points for the values of $\lambda = 4/7$ (left) and $\lambda = 4/11$ (right).

Two examples of λ-centered points are depicted in Figure 4.1. Observe that moving to λ-centered points while maintaining connectivity does not necessarily enforce a final gathering of the protocols (cf. Observation 4.2). Consider, for instance, two robots. A protocol that demands the two robots to move halfway towards the midpoint between themselves would compute $1/4$-centered target points, but the robots would only *converge* towards the same position. The robots must be guaranteed to compute the same target point eventually to obtain a final gathering. We demand this by requiring that there is a constant $c < 1$, such that $N_i(t) = N_j(t)$ and that $\Delta_i(t) = \Delta_j(t) \leq c$ implies that the robots compute the same target point. Protocols that have this property are called *collapsing*. Observe that being collapsing is reasonable since λ-centered points are always inside local convex hulls and hence, the robots' local diameters are monotonically decreasing in case no further robot enters their neighborhood. Hence, demanding a threshold to enforce moving to the same point is necessary to ensure a final gathering. For ease of description, we fix $c = 1/2$ in this work. However, c could be chosen as an arbitrary constant by scaling the obtained runtime bounds with a factor of $1/c$. The combination of connectivity preserving, collapsing and λ-centered points leads to the notion of λ-contracting protocols.

Definition 4.7 A connectivity preserving and collapsing discrete robot formation protocol $\mathcal{P}$ is called λ-*contracting* if $\text{target}_i^{\mathcal{P}}(t)$ is a λ-centered point of $CH_i(t)$ for every robot r_i and every $t \in \mathbb{N}_0$.

4.4.3 Analysis of λ-contracting Protocols

In the following, we state the upper and lower bounds about λ-contracting protocols. We start with a lower bound that holds for an even larger class of protocols. The lower bound holds for all discrete gathering protocols that compute robot target points always inside local convex hulls.

Theorem 4.3 For every protocol for robots in the $\mathcal{OBLOT}_1^F$ model that computes the target point of a robot always inside of the local convex hull of all visible robots, there exists an initial configuration where the protocol requires $\Omega(\Delta^2)$ rounds to gather all robots.

To familiarize ourselves with the core ideas, we first focus on the two-dimensional case and state a matching upper bound for λ-contracting protocols of robots located in the Euclidean plane.

Theorem 4.6 Consider a swarm of n robots located in $\mathbb{R}^2$. Every λ-contracting protocol gathers all robots in $\frac{171 \cdot \pi \cdot \Delta^2}{\lambda^3} + 1 \in \mathcal{O}\left(\Delta^2\right)$ rounds.

Afterward, we prove a similar theorem (with a slightly increased runtime bound) for robots located in $\mathbb{R}^d$.

Theorem 4.7 Consider a swarm of n robots located in $\mathbb{R}^d$. Every λ-contracting protocol gathers all robots in $\frac{256 \cdot \pi \cdot \Delta^2}{\lambda^3} + 1 \in \mathcal{O}\left(\Delta^2\right)$ rounds.

4.4.3.1 Lower Bound

Theorem 4.3 For every protocol for robots in the $\mathcal{OBLOT}_1^F$ model that computes the target point of a robot always inside of the local convex hull of all visible robots, there exists an initial configuration where the protocol requires $\Omega(\Delta^2)$ rounds to gather all robots.

Proof. The presented lower bound is a generalization of the lower bound of the GTC protocol [50]. In the following, we assume $n \geq 5$. Consider n robots that are located on the vertices of a regular polygon with side length 1. Observe first that due to the disorientation and because the protocols are deterministic, the local coordinate systems of the robots could be chosen such that the configuration remains a regular polygon forever (see Figure 4.2 for an example).

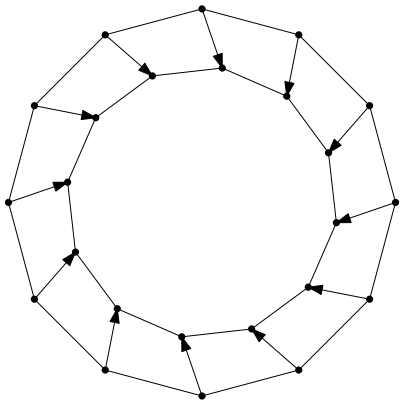

Figure 4.2: Initially, the robots are located on the surrounding regular polygon. The local coordinate systems of the robots can be chosen such that all robots execute the same movement in a rotated fashion such that the configuration remains a regular polygon.

Henceforth, we assume in the following that the robots remain on the vertices of a regular polygon. Let C be the surrounding circle and r_C its radius. For large n, the circumference p_C of C is $\approx n$ and $r_C \approx n/2\pi$. Hence, $\Delta \approx n/\pi$. We show that any λ-contracting protocol requires $\Omega\left(\Delta^2\right)$ rounds until $p_C \leq 2n/3$. As long as $p_C \geq 2n/3$, each robot can observe exactly two neighbors at distance $2/3 \leq s \leq 1$.

The internal angles of a regular polygon have a size of $\gamma = (n-2) \cdot \pi/n$. Fix any robot r_i, translate and rotate the global coordinate system such that $p_i(t) = (0,0)$ and

$$p_{i-1}(t) = \left(-s \cdot \sin\left(\gamma/2\right), s \cdot \cos\left(\gamma/2\right)\right)$$
$$p_{i+1}(t) = \left(s \cdot \sin\left(\gamma/2\right), s \cdot \cos\left(\gamma/2\right)\right).$$

Now, consider the target point $\text{target}_i^{\mathcal{P}}(t) = \left(x_{\text{target}_i^{\mathcal{P}}(t)}, y_{\text{target}_i^{\mathcal{P}}(t)}\right)$. Observe that the radius r_C decreases by exactly $y_{\text{target}_i^{\mathcal{P}}(t)}$. Next, we derive an upper bound on $y_{\text{target}_i^{\mathcal{P}}(t)}$.

$$y_{\text{target}_i^{\mathcal{P}}(t)} = s \cdot \cos\left(\gamma/2\right) \leq \cos\left(\gamma/2\right) = \cos\left((n-2) \cdot \pi/2n\right) = \cos\left(\pi/2 - \pi/n\right) = \sin\left(\pi/n\right).$$

For $x \geq 0$ it always holds $\sin(x) \leq x$. Hence, $\sin\left(\pi/n\right) \leq \pi/n$. Therefore, it takes at least $n^2/3\pi$ rounds until r_C has decreased by at least $n/3$. The same holds for the perimeter. All in all, it takes at least $n^2/3\pi \in \Omega\left(\Delta^2\right)$ rounds until r_C decreases by at least $n/3$. ■

4.4.3.2 Upper Bound for $d = 2$

The proof aims to show that the radius of the global smallest enclosing circle (SEC), i.e., the SEC that encloses all robots' positions in a global coordinate system, decreases by $\Omega(1/\Delta)$ every two rounds. Since the initial radius is upper bounded by Δ, the runtime of $\mathcal{O}(\Delta^2)$ follows. See Figure 4.3 for a visualization. The proof idea is inspired by the analysis of the GTC protocol [50].

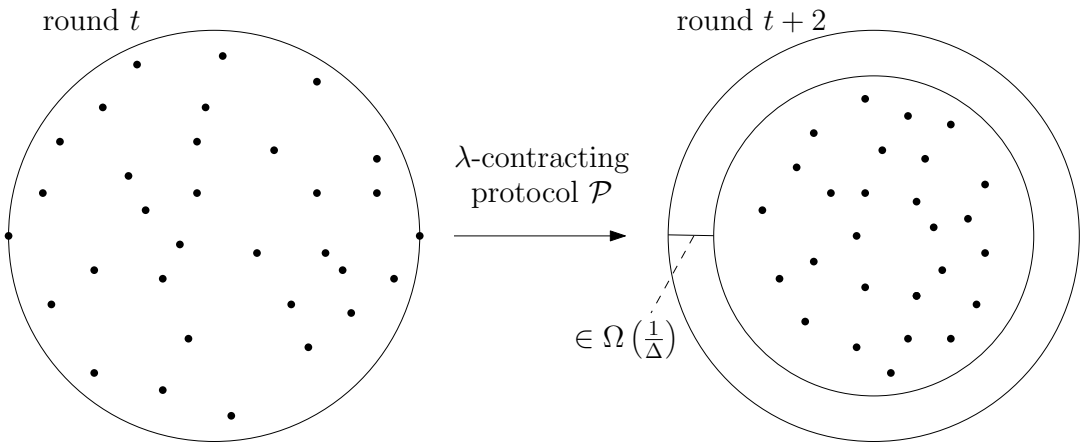

Figure 4.3: We show that the radius of the global SEC decreases by $\Omega(1/\Delta)$ every two rounds.

High-level Analysis: We consider the fixed circular segment S_λ of the global SEC and analyze how the inside robots behave. A circular segment is a region of a circle "cut off" by a chord. The circular segment S_λ has a chord length of at most $\lambda/4$ (for a formal definition, see below) and we can prove a height of S_λ in the order of $\Omega(1/\Delta)$ (Lemma 4.11). Observe that in any circular segment, the chord's endpoints are the points that have a maximum distance within the circular segment, and hence, the maximum distance between any pair of points in S_λ is at most $\lambda/4$. Now, we split the robots inside of S_λ into two classes: the robots r_i with $\Delta_i(t) > 1/4$ and the others with $\Delta_i(t) \leq 1/4$. Recall that every robot r_i moves to the λ-centered point $\text{target}_i^{\mathcal{P}}(t)$. Moreover, $\text{target}_i^{\mathcal{P}}(t)$ is the midpoint of a line segment ℓ of length $\lambda \cdot \Delta_i(t)$ that is completely contained in the local convex hull of r_i. For robots with $\Delta_i(t) > 1/4$ we have that ℓ is larger than $\lambda/4$ and thus, ℓ cannot be completely contained in S_λ. Hence, ℓ either connects two points outside of S_λ or one point inside and another outside. In the former case, $\text{target}_i^{\mathcal{P}}(t)$ is outside of S_λ, and in the latter case $\text{target}_i^{\mathcal{P}}(t)$ is outside of a circular segment with half the height h of S_λ. See Lemma 4.12 for a formal statement of the first case.

It remains to argue about robots with $\Delta_i(t) < 1/4$. Here, we consider a circular segment with an even smaller height, namely $h \cdot \lambda/4$. We will see that all robots which compute a target point inside this circular segment (which can only be robots with $\Delta_i(t) < \lambda/4$) will move exactly to the same position. Hence, in round $t+1$ there is only one position in the circular segment with height $h \cdot \lambda/4$ occupied by robots. All other robots are located outside of the circular segment with height $h/2$. As a consequence, for all robots r_i in the circular segment with height $h \cdot \lambda/4$, it must hold $\text{target}_i^{\mathcal{P}}(t)$ is outside of the circular segment with height $h \cdot \lambda/4$. See Lemma 4.13 for a formal statement. Finally, Lemma 4.14 combines the previous statements and gives a lower bound on how much the radius of the global SEC decreases.

Detailed Analysis: First, we introduce some definitions. Let $N := N(t)$ be the (global) smallest enclosing circle of all robots in round t and $R := R(t)$ its radius. Now, fix any point b on the boundary of N. The two points at a distance of $\lambda/8$ of b on the boundary of N determine the circular segment S_λ with height h. In the following, we determine by $S_\lambda(c)$ for $0 < c \leq 1$ the circular segment with height $c \cdot h$ that is contained in S_λ. See Figure 4.4 for a depiction of the circular segments S_λ and $S_\lambda(1/2)$ (that is used in the proofs). All subsequent lemmata consider robots that move according to a λ-contracting protocol $\mathcal{P}$.

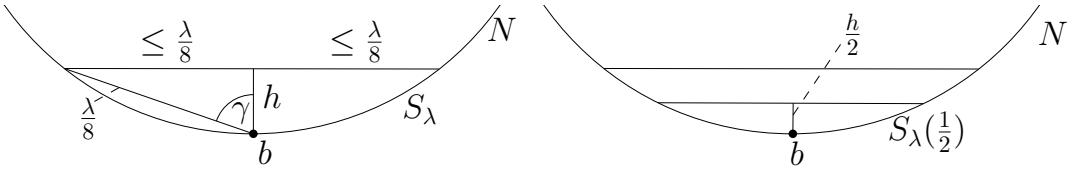

Figure 4.4: The circular segments S_λ (to the left) and $S_\lambda\left(1/2\right)$ of the global SEC N is depicted.

In the following, we prove that all robots leave the circular segment $S_\lambda\left(\lambda/4\right)$ every two rounds. As a consequence, the radius of N decreases by at least $\lambda/4 \cdot h$. Initially, we give a bound on h. We use Jung's Theorem to obtain a bound on R and also on h.

Theorem 4.8 — Jung's Theorem [82, 83]. The smallest enclosing hypersphere of a point set $K \subset \mathbb{R}^d$ with diameter $diam$ has a radius of at most $diam \cdot \sqrt{\frac{d}{2\cdot(d+1)}}$.

Lemma 4.11 $h \geq \frac{\sqrt{3}\cdot\lambda^2}{64\pi\Delta}$.

Proof. Initially, we give an upper bound on the angle γ, see Figure 4.4 for its definition. The circumference of N is $2\pi R$. We can position at most $\frac{16}{\lambda}\pi R$ points on the boundary of N that are at distance $\frac{\lambda}{8}$ from the points closest to them and form a regular convex polygon. The internal angle of this regular polygon is 2γ. Hence, the sum of all internal angles is $\left(\frac{16}{\lambda}\pi R - 2\right)\cdot\pi$. Thus, each individual angle has a size of at most $\frac{\left(\frac{16}{\lambda}\pi R-2\right)\cdot\pi}{\frac{16}{\lambda}\pi R} = \pi - \frac{2\pi}{\frac{16}{\lambda}\pi R} = \pi - \frac{\lambda}{8R}$. Hence, $\gamma \leq \frac{\pi}{2} - \frac{\lambda}{16R}$. Next, we can bound h. First of all, we derive a relation between h and γ: $\cos(\gamma) = \frac{h}{\frac{\lambda}{8}} = \frac{8h}{\lambda} \iff h = \frac{\lambda\cdot\cos(\gamma)}{8}$. With help of the inequality $\cos(x) \geq -\frac{2}{\pi}x + 1$ for $x \in [0, \frac{\pi}{2}]$, we obtain

$$h = \frac{\lambda\cdot\cos(\gamma)}{8} \geq \frac{\lambda\cdot\cos\left(\frac{\pi}{2} - \frac{\lambda}{16R}\right)}{8} \geq \frac{\lambda\cdot\left(-\frac{2}{\pi}\cdot\left(\frac{\pi}{2} - \frac{\lambda}{16R}\right) + 1\right)}{8} = \frac{\lambda\cdot\frac{\lambda}{8\pi R}}{8} = \frac{\lambda^2}{64\pi R}.$$

Applying Theorem 4.8 with $d = 2$ yields $h \geq \frac{\sqrt{3}\cdot\lambda^2}{64\pi\Delta}$. ■

We continue to prove that all robots leave $S_\lambda\left(\lambda/4\right)$ every two rounds. First of all, we analyze robots for which $\Delta_i(t) > 1/4$. These robots even leave the larger circular segment $S_\lambda\left(1/2\right)$.

Lemma 4.12 For any robot r_i with $\Delta_i(t) > 1/4$: $\text{target}_i^{\mathcal{P}}(t) \in N \setminus S_\lambda\left(1/2\right)$.

Proof. Since $\Delta_i(t) > 1/4$ and $\mathcal{P}$ is λ-contracting, $\text{target}_i^{\mathcal{P}}(t)$ is the midpoint of a line segment $\ell_i^{\mathcal{P}}(t)$ of length at least $\lambda \cdot \Delta_i(t) > \lambda/4$. As the maximum distance between any pair of points inside of S_λ is $\frac{\lambda}{4}$, it follows that $\ell_i^{\mathcal{P}}(t)$ either connects two points outside of S_λ or one point inside and another point outside. In the first case, $\text{target}_i^{\mathcal{P}}(t)$ lies outside of S_λ (since the maximum distance between any pair of points inside of S_λ is $\frac{\lambda}{4} \leq 1/4 < \Delta_i(t)$). In the second case, $\text{target}_i^{\mathcal{P}}(t)$ lies outside of $S_\lambda\left(1/2\right)$ since, in the worst case, one endpoint of $\ell_i^{\mathcal{P}}(t)$ is the point b used in the definition of N and the second point lies very close above of $S_\lambda\left(1/2\right)$. Since $\text{target}_i^{\mathcal{P}}(t)$ is the midpoint of $\ell_i^{\mathcal{P}}(t)$, it lies closely above $S_\lambda\left(1/2\right)$. Every other position of the two endpoints of $\ell_i^{\mathcal{P}}(t)$ would result in a point $\text{target}_i^{\mathcal{P}}(t)$ that lies even farther above $S_\lambda\left(1/2\right)$. ■

Now, we consider the case of a single robot in $S_\lambda\left(\lambda/4\right)$, and its neighbors are located outside of $S_\lambda\left(1/2\right)$. We prove that this robot leaves $S_\lambda\left(\lambda/4\right)$. Additionally, we prove that none of the robots outside of $S_\lambda\left(1/2\right)$ that see the single robot in $S_\lambda\left(\lambda/4\right)$ enters $S_\lambda\left(\lambda/4\right)$.

Lemma 4.13 Consider a robot r_i located in $S_\lambda\left(\lambda/4\right)$. If all its neighbors are located outside of $S_\lambda\left(1/2\right)$, $\text{target}_i^{\mathcal{P}}(t) \in N \setminus S_\lambda\left(\lambda/4\right)$. Similarly, for a robot r_i that is located outside of $S_\lambda\left(1/2\right)$ and that has only one neighbor located in $S_\lambda\left(\lambda/4\right)$, $\text{target}_i^{\mathcal{P}}(t) \in N \setminus S_\lambda\left(\lambda/4\right)$.

Proof. First, consider a robot r_i that is located in $S_\lambda\left(\lambda/4\right)$ and all its neighbors are above $S_\lambda\left(1/2\right)$. Let p_1 and p_2 be the two points of $CH_i(t)$ closest to the intersection points of $CH_i(t)$ and the boundary of $S_\lambda\left(1/2\right)$ (p_1 and p_2 are infinitesimally above of $S_\lambda\left(1/2\right)$). In case $CH_i(t)$ consists of only two robots, define p_1 to be the intersection point of $CH_i(t)$ and $S_\lambda\left(1/2\right)$ and $p_2 = p_i(t)$. The maximum distance $d_{\max}$ between any pair of points in $CH_i(t) \cap S_\lambda\left(1/2\right)$ is less than $\max\{\|p_1 - p_2\|_2, \|p_1 - p_i(t)\|_2, \|p_2 - p_i(t)\|_2\}$, since p_1 and p_2 are slightly above of $S_\lambda\left(1/2\right)$. Clearly, $\Delta_i(t) \geq d_{\max}$. Thus, the maximum distance between any pair of points in $CH_i(t) \cap S_\lambda\left(\frac{\lambda}{2}\right)$ is less than $\lambda \cdot d_{\max}$. We conclude that $\text{target}_i^{\mathcal{P}}(t)$ must be located above of $S_\lambda\left(\lambda/4\right)$ since $\text{target}_i^{\mathcal{P}}(t)$ is the midpoint of a line segment of length $\lambda \cdot \Delta_i(t) \geq \lambda \cdot d_{\max}$ either connecting two robots above of $S_\lambda\left(\lambda/4\right)$ or one robot inside of $S_\lambda\left(\lambda/4\right)$ and one robot outside of $S_\lambda\left(\lambda/4\right)$. The arguments for the opposite case – r_i is located in $S_\lambda\left(1/2\right)$, one neighbor of r_i is located in $S_\lambda\left(\lambda/4\right)$ and all others are also outside of $S_\lambda\left(1/2\right)$ – are analogous. ■

Next, we derive with help of Lemmata 4.12 and 4.13 that $S_\lambda\left(\lambda/4\right)$ is empty after two rounds. Additionally, we analyze how much $R(t)$ decreases.

Lemma 4.14 For any round t with $\Delta(t) \geq 1/2$, $R(t+2) \leq R(t) - \frac{\lambda^3 \cdot \sqrt{3}}{256 \cdot \pi \cdot \Delta}$.

Proof. Fix any circular S_λ and consider the set of robots R_S that are located in $S_\lambda\left(\lambda/4\right)$ or compute a target point in $S_\lambda\left(\lambda/4\right)$. Via Lemma 4.12, we obtain that for every robot $r_i \in R_S$ that computes a target point in $S_\lambda\left(\lambda/4\right)$, $\Delta_i(t) \leq 1/4$. Since the maximum distance between any pair of points in $S_\lambda\left(\lambda/4\right)$ is $1/4$, we conclude that a robot that is not located in $S_\lambda\left(\lambda/4\right)$ but computes its target point inside, is at distance at most $1/4$ from S_λ. Hence, via the triangle inequality, it is located at distance at most $1/2$ from any other robot in R_S. Thus, all robots in R_S can see each other. Now, consider the robot $r_{\min} \in R_S$ which is the robot of R_S with the minimal number of visible neighbors. Furthermore, $A_{\min}$ is the set of robots that have the same neighborhood as $r_{\min}$. For all robots $r_j \in R_S \setminus A_{min}$, we have that r_j can see $r_{\min}$ and at least one robot that $r_{\min}$ cannot see. Thus, $\Delta_j(t) > 1$. We can conclude with help of Lemma 4.12 that all robots in $R_S \setminus A_{\min}$ compute a target point outside of $S_\lambda\left(1/2\right)$. Since all robots $r_i \in A_{\min}$ have the same neighborhood and $\Delta_i(t) < 1/4$, they also compute the same target point. Thus, at the beginning of round $t+1$, at most one position in $S_\lambda\left(\lambda/4\right)$ is occupied. In round $t+1$ we have the picture that one position in $S_\lambda\left(\lambda/4\right)$ is occupied and all neighbors are located above $S_\lambda\left(1/2\right)$. Lemma 4.13 yields that the robots in $S_\lambda\left(\lambda/4\right)$ compute a target point outside. Moreover, Lemma 4.13 yields as well that no robot outside of $S_\lambda\left(\lambda/4\right)$ computes a target point inside and thus, $S_\lambda\left(\lambda/4\right)$ is empty in round $t+2$. Since the circular segment S_λ has been chosen arbitrarily, the arguments hold for the entire circle N and thus, $R(t+2) \leq R(t) - \lambda/4 \cdot h \leq R(t) - \frac{\lambda^3\sqrt{3}}{256 \cdot \pi \cdot \Delta}$. ■

Finally, we can conclude with help of Lemma 4.14 the main Theorem 4.6.

Theorem 4.6 Consider a swarm of n robots located in $\mathbb{R}^2$. Every λ-contracting protocol gathers all robots in $\frac{171 \cdot \pi \cdot \Delta^2}{\lambda^3} + 1 \in \mathcal{O}\left(\Delta^2\right)$ rounds.

Proof. First, we bound the initial radius of N: $R(0) \leq \Delta/\sqrt{3}$ (Theorem 4.8). Lemma 4.14 yields that $R(t)$ decreases every two rounds by at least $\frac{\lambda^3 \cdot \sqrt{3}}{256 \cdot \pi \cdot \Delta}$. Thus, it requires $2 \cdot \frac{256 \cdot \pi \cdot \Delta}{\lambda^3}$ rounds until $R(t)$ decreases by at least $\sqrt{3}$. Next, we bound how often this can happen until $R(t) \leq \frac{1}{4}$ and thus $\Delta(t) \leq \frac{1}{2}$:

$$\frac{\Delta}{\sqrt{3}} - x \cdot \sqrt{3} \leq \frac{1}{4} \iff \frac{\Delta}{3} - \frac{1}{4 \cdot \sqrt{3}} \leq x.$$

All in all, it requires $x \cdot \frac{512 \cdot \pi \cdot \Delta}{\lambda^3} = \left(\frac{\Delta}{3} - \frac{1}{4 \cdot \sqrt{3}}\right) \cdot \frac{512 \cdot \pi \cdot \Delta}{\lambda^3} \leq \frac{171 \cdot \pi \cdot \Delta^2}{\lambda^3}$ rounds until $\Delta(t) \leq \frac{1}{2}$. As soon as $\Delta(t) \leq \frac{1}{2}$, all robots can see each other, compute the same target point (the protocol is collapsing) and will reach it in the next round.

■

4.4.3.3 Upper Bound for Arbitrary d

The upper bound we derived for two dimensions can also be generalized to every dimension d. Only the constants in the runtime increase slightly.

Theorem 4.7 Consider a swarm of n robots located in $\mathbb{R}^d$. Every λ-contracting protocol gathers all robots in $\frac{256 \cdot \pi \cdot \Delta^2}{\lambda^3} + 1 \in \mathcal{O}\left(\Delta^2\right)$ rounds.

The analysis is in most parts analogous to the analysis of λ-contracting protocols in two dimensions (Section 4.4.3.2). Let $GS := GS(t)$ be the smallest enclosing hypersphere (SEH) of all robots in round t and $R := R(t)$ its radius. Let B be an arbitrary point on the surface and define the hyperspherical cap HSC_λ with apex B as follows. Choose the height h of HSC_λ such that the inscribed hypercone has a slant height of $\lambda/8$. Note that this implies that the radius a of the base of the cap is upper bounded by $\lambda/8$. Hence, the maximal distance between any pair of points in HSC_λ is $\lambda/4$. In the following, we denote by $\text{HSC}_\lambda(c)$ for $0 \leq c \leq 1$ the hyperspherical cap with apex B of height $c \cdot h$.

Lemma 4.15 $h \geq \frac{\sqrt{2} \cdot \lambda^2}{64 \cdot \pi \cdot \Delta}$.

Proof. Initially, we give a bound on the angle γ which is the angle between the height h and the slant height of $\lambda/8$ of the inscribed hypercone. Consider a circle K with radius R that has the same center as GS and contains B. The angle γ can now be seen as the internal angle of a regular polygon with side length $\lambda/8$ whose vertices lie on K. The circumference of K is $2 \cdot \pi \cdot R$. Thus, we can position at most $\frac{16}{\lambda} \cdot \pi \cdot R$ points on the boundary of K that are at a distance of $\lambda/8$ from the points closest to them and form a regular convex polygon. The internal angle of this regular polygon is $2 \cdot \gamma$. Hence, the sum of all internal angles is $\left(\frac{16}{\lambda} \cdot \pi \cdot R - 2\right) \cdot \pi$. Thus, each individual angle has a size of at most $\frac{\left(\frac{16}{\lambda} \cdot \pi \cdot R - 2\right) \cdot \pi}{\frac{16}{\lambda} \cdot \pi \cdot R} = \pi - \frac{2 \cdot \pi}{\frac{16}{\lambda} \cdot \pi \cdot R} = \pi - \frac{\lambda}{8 \cdot R}$. Hence, $\gamma \leq \frac{\pi}{2} - \frac{\lambda}{16 \cdot R}$. Now, we can bound h. First of all, we derive a relation between h and γ: $\cos(\gamma) = \frac{h}{\frac{\lambda}{8}} = \frac{8h}{\lambda} \iff h = \frac{\lambda \cdot \cos(\gamma)}{8}$. In the following upper bound, we make use of the fact that $\cos(x) \geq -\frac{2}{\pi}x + 1$ for $x \in [0, \frac{\pi}{2}]$.

$$h = \frac{\lambda \cdot \cos(\gamma)}{8} \geq \frac{\lambda \cdot \cos\left(\frac{\pi}{2} - \frac{\lambda}{16 \cdot R}\right)}{8} \geq \frac{\lambda \cdot \left(-\frac{2}{\pi} \cdot \left(\frac{\pi}{2} - \frac{\lambda}{16 \cdot R}\right) + 1\right)}{8} = \frac{\lambda \cdot \frac{\lambda}{8 \cdot \pi \cdot R}}{8} = \frac{\lambda^2}{64 \pi R}$$

Lastly observe that $R \leq \Delta \cdot \sqrt{\frac{d}{2 \cdot (d+1)}}$ (Theorem 4.8). For any $d \geq 1$, it holds $\sqrt{\frac{d}{2 \cdot (d+1)}} \leq \frac{1}{\sqrt{2}}$ and thus $R \leq \frac{\Delta}{\sqrt{2}}$. We obtain a final lower bound on h: $h \geq \frac{\lambda^2}{64 \pi R} \geq \frac{\sqrt{2} \cdot \lambda^2}{64 \cdot \pi \cdot \Delta}$.

■

Lemma 4.16 For a robot r_i with $\Delta_i(t) > 1/4$ it holds $\text{target}_i^{\mathcal{P}}(t) \in GS \setminus \text{HSC}_\lambda\left(\frac{1}{2}\right)$.

Proof. Analogous to the proof of Lemma 4.12. ■

Lemma 4.17 Consider a robot r_i located in $\mathrm{HSC}_\lambda\left(\frac{\lambda}{4}\right)$. If all its neighbors are located outside of $\mathrm{HSC}_\lambda\left(\frac{1}{2}\right)$, it holds $\mathrm{target}_i^{\mathcal{P}}(t) \in GS \setminus \mathrm{HSC}_\lambda\left(\frac{\lambda}{4}\right)$. Similarly, for a robot r_i that is located outside of $\mathrm{HSC}_\lambda\left(\frac{1}{2}\right)$ and that has only one neighbor located in $\mathrm{HSC}_\lambda\left(\frac{\lambda}{4}\right)$, it holds $\mathrm{target}_i^{\mathcal{P}}(t) \in GS \setminus \mathrm{HSC}_\lambda\left(\frac{\lambda}{4}\right)$.

Proof. Analogous to the proof of Lemma 4.13. ■

Lemma 4.18 For any round t with $\Delta(t) \geq 1/2$, it holds $R(t+2) \leq R(t) - \frac{\lambda^3 \cdot \sqrt{2}}{256 \cdot \pi \cdot \Delta}$.

Proof. Analogous to the proof of Lemma 4.14 by replacing the lower bound of h by $\frac{\sqrt{2} \cdot \lambda^2}{64 \cdot \pi \cdot \Delta}$ (Lemma 4.15). ■

Theorem 4.2 Every λ-contracting protocol gathers a swarm of n disoriented robots located in $\mathbb{R}^d$ in the $\mathcal{OBLOT}_1^F$ model in $\Theta(\Delta^2)$ rounds.

Proof. First, we bound the initial radius of GS: $R(0) \leq \frac{\Delta}{\sqrt{2}}$ (Theorem 4.8). Lemma 4.18 yields that $R(t)$ decreases every two rounds by at least $\frac{\lambda^3 \cdot \sqrt{2}}{256 \cdot \pi \cdot \Delta}$. Thus, it requires $2 \cdot \frac{256 \cdot \pi \cdot \Delta}{\lambda^3}$ rounds until $R(t)$ decreases by at least $\sqrt{2}$. Next, we bound how often this can happen until $R(t) \leq \frac{1}{4}$ and thus $\Delta(t) \leq \frac{1}{2}$ holds: $\frac{\Delta}{\sqrt{2}} - x \cdot \sqrt{2} \leq \frac{1}{4} \iff \frac{\Delta}{2} - \frac{1}{4 \cdot \sqrt{2}} \leq x$.

All in all, it requires $x \cdot \frac{512 \cdot \pi \cdot \Delta}{\lambda^3} = \left(\frac{\Delta}{2} - \frac{1}{4 \cdot \sqrt{2}}\right) \cdot \frac{512 \cdot \pi \cdot \Delta}{\lambda^3} \leq \frac{256 \cdot \pi \cdot \Delta^2}{\lambda^3}$ rounds until $\Delta(t) \leq \frac{1}{2}$. As soon as $\Delta(t) \leq \frac{1}{2}$, all robots can see each other, compute the same target point (the protocol is collapsing) and will reach it in the next round.

■

4.4.4 Examples of λ-contracting Protocols

Next, we present an exemplary λ-contracting protocol. Before introducing the concrete protocols, we describe an important subclass of λ-contracting protocols, denoted as (α, β)-contracting protocols, a powerful tool to decide whether a given protocol is λ-contracting. Afterward, we introduce the known protocol GTC [5], prove it to be (α, β)-contracting, and, thus, also λ-contracting.

(α, β)-contracting Protocols. While the definition of λ-contracting protocols describes the core properties of efficient protocols to solve GATHERING, it might be practically challenging to determine whether a given protocol is λ-contracting. Concrete protocols often are designed as follows: robots compute a *desired* target point and move as close as possible towards it without losing connectivity [5, 22, 98]. The GTC protocol, for instance, uses this rule. Since the robots do not necessarily reach the desired target point, it is hard to determine whether the resulting point is λ-centered. Therefore, we introduce a two-stage definition: (α, β)-contracting protocols. The parameter α represents an α-centered point (Definition 4.6) and β describes how close the robots move towards the point. As an intermediate step, we define the β-scaled convex hull around an arbitrary point.

Definition 4.8 Let $c_1, \ldots, c_k$ with $c_i \in \mathbb{R}^d$ be the vertices of a convex polytope Q, $p \in Q$ some point of Q and $\beta \in (0, 1]$ a constant. Then, $Q(p, \beta)$ is the convex polytope with vertices $p + (1 - \beta) \cdot (c_i - p)$.

Now, we are ready to define the class of (α, β)-contracting protocols. It uses a combination of Definitions 4.6 and 4.8: the target points of the robots must be inside of the β-scaled local convex hull around an α-centered point. See also Figure 4.5 for a visualization of valid target points

in (α,β)-contracting protocols. Recall that $CH_i(t)$ defines the convex hull of all neighbors of r_i including r_i in round t and $CH_i(t)(p,\beta)$ is the scaled convex hull around p (Definition 4.8).

Definition 4.9 A connectivity preserving and collapsing discrete robot formation protocol $\mathcal{P}$ is called to be (α,β)-*contracting*, if there exists an α-centered point α-center$_i^{\mathcal{P}}(t)$ such that target$_i^{\mathcal{P}}(t) \in CH_i(t)\big(\alpha\text{-center}_i^{\mathcal{P}}(t),\beta\big)$ for every robot r_i and every $t \in \mathbb{N}_0$.

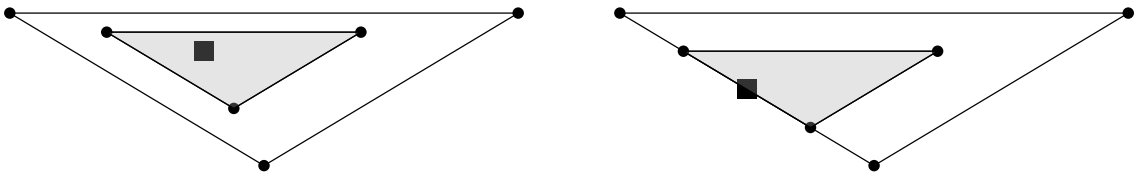

Figure 4.5: Two examples of valid target points of (α,β)-contracting protocols. The small gray triangle represents the $\frac{1}{2}$-scaled convex hull around an $\frac{1}{4}$-centered point marked with a square.

Next, we state the relation between (α,β)-contracting and λ-contracting protocols.

Theorem 4.9 Every (α,β)-contracting protocol $\mathcal{P}$ is λ-contracting with $\lambda = \alpha\cdot\beta$.

Proof. From the definition of (α,β)-contracting protocols, we know that for a target point target$_i^{\mathcal{P}}(t)$, there exists a point α-center$_i^{\mathcal{P}}(t)$ such that target$_i^{\mathcal{P}}(t) \in CH_i(t)(\alpha\text{-center}_i^{\mathcal{P}}(t),\beta)$. We do the following geometric construction in Figure 4.6. Let $p = \alpha\text{-center}_i^{\mathcal{P}}(t)$ and $p' = \text{target}_i^{\mathcal{P}}(t)$. We draw a line segment from α-center$_i^{\mathcal{P}}(t)$ through target$_i^{\mathcal{P}}(t)$ to the boundary of $CH_i(t)$. Let c be the endpoint of this line segment. Because p is α-centered, there exists a line segment with length $\Delta_i(t)\cdot\alpha$ through p, let this be the line segment $\overline{ab}$. The line segment $\overline{a'b'}$ is a parallel to $\overline{ab}$ inside the triangle $\triangle_{abc}$. We know that $p' \in CH_i(t)(p,\beta)$, therefore $|\overline{cp'}| \geq \beta|\overline{cp}|$. By the intercept theorem, it follows that $|\overline{a'b'}| \geq \beta|\overline{ab}| = \beta\cdot\alpha\cdot\Delta_i(t)$. Because the points a,b and c are all inside $CH_i(t)$, the entire triangle $\triangle_{abc}$ and $\overline{a'b'}$ are inside $CH_i(t)$ as well. Therefore, target$_i^{\mathcal{P}}(t)$ is a λ-centered point with $\lambda = \alpha\cdot\beta$.

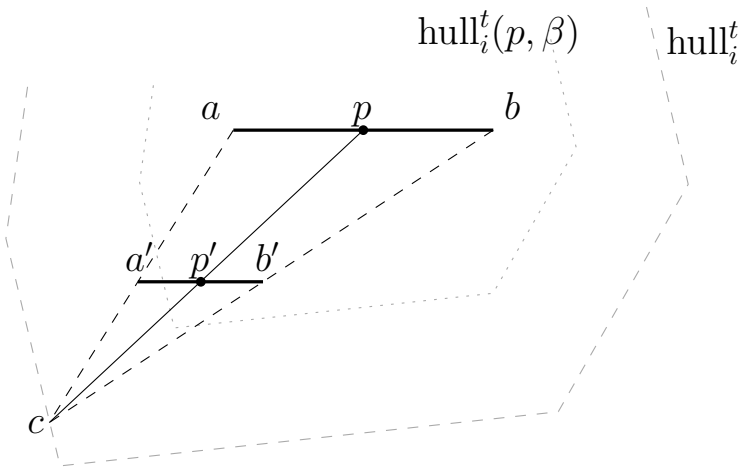

Figure 4.6: The construction used in the proof of Theorem 4.9.

■

Go-To-The-Center. Next, we study the two-dimensional GTC protocol [5] and its generalization to d-dimension and prove the protocols to be (α,β)-contracting. First, we focus on the original (two-dimensional) protocol for which it is already known that it gathers all robots in $\mathcal{O}\left(n+\Delta^2\right)$ rounds [50]. We show that GTC is (α,β)-contracting (hence also λ-contracting) and thus, obtain an

improved upper runtime bound of $\mathcal{O}(\Delta^2)$. Robots always move towards the center of the smallest enclosing circle of their neighborhood. To maintain connectivity, *limit circles* are used. Each robot r_i always stays within the circle of radius $1/2$ centered in the midpoint m_j of every visible robot r_j. Since each robot r_j does the same, it is ensured that two visible robots always stay within a circle of radius $1/2$ and thus, they remain connected. Consequently, robots move only that far towards the center of the smallest enclosing circle such that no limit circle is left. The protocol is formally described in Algorithm 2.

Algorithm 2 GTC (view of robot r_i)

1: $C_i(t) :=$ smallest enclosing circle of $N_i(t)$
2: $c_i(t) :=$center of $C_i(t)$
3: $\forall r_j \in N_i(t) : m_j :=$ midpoint between r_i and r_j
4: D_j : disk with radius $\frac{1}{2}$ centered at m_j
5: seg := line segment $\overline{p_i(t), c_i(t)}$
6: $A := \bigcap_{r_j \in N_i(t)} D_j \cap$ seg
7: $x :=$ point in A closest to $c_i(t)$
8: $\text{target}_i^{\text{GTC}}(t) := x$

Subsequently, we prove that GTC is (α,β)-contracting. First, we derive a bound on α.

Lemma 4.19 The center of the SEC of a convex polygon Q is $\frac{\sqrt{3}}{8}$-centered.

Proof. Let C denote the smallest enclosing circle (SEC) of Q and diam the maximal distance of two points in Q. We need to distinguish two cases: either two points are located on the boundary of the SEC or (at least) 3 that form an acute triangle. In the first case, the two points are located on the diameter of C. Hence, the center of the smallest enclosing circle is 1-centered since it equals the midpoint of the two robots that define the diameter.

In the second case, we focus on the three points that form the acute triangle. We denote the three points by ABC and the triangle by $\triangle_{ABC}$. Moreover, a,b,c denote the edges of $\triangle_{ABC}$. Since $\triangle_{ABC}$ is acute, we have that its circumcircle equals its SEC. Consequently, C equals to the SEC of $\triangle_{ABC}$. Let p denote the center of C and r its radius. The proof aims to show that there exists a line segment ℓ with midpoint p that is parallel to a,b or c and has a length of $\alpha \cdot$ diam, where diam denotes the diameter of Q (we will determine the concrete value for α shortly). Without loss of generality, we assume that a is the longest edge of $\triangle_{ABC}$. Since C is also the SEC of $\triangle_{ABC}$, $r \leq \frac{a}{\sqrt{3}}$ and, thus, also $r \cdot \sqrt{3} \leq a$ (Theorem 4.8). Additionally, we have $r \geq \frac{\text{diam}}{2}$. Hence, $\frac{\text{diam} \cdot \sqrt{3}}{2} \leq a$. Now, we rotate the coordinate system, such that $B = \left(0, -\frac{a}{2}\right), C = \left(0, \frac{a}{2}\right)$ and $A = (x_a, y_a)$. See Figure 4.7 for a visualization of the setting. We now consider $y_a \leq 0$. The arguments for $y_a > 0$ can be derived analogously with swapped roles of B and C.

Observe first, that $p = (x_p, 0)$ since p is located on the intersection of the perpendicular bisectors of a,b and c. Additionally, we have $x_p \leq \frac{x_a}{2}$ since both the starting points of the perpendicular bisectors of b and c have the x-coordinate $\frac{x_p}{2}$ and the bisectors have monotonically decreasing x-coordinates. Now, we distinguish two cases: $-\frac{a}{4} \leq y_a \leq 0$ and $y_a < -\frac{a}{4}$. In the first case, we prove that there exists a line segment parallel to a with its midpoint in p that has a length of at least $\frac{a}{4}$. We model the edge b as a linear function $f(x) = -\frac{\frac{a}{2} - y_a}{x_a} \cdot x + \frac{a}{2}$. The value $f(x_p)$ is minimized for $x_p = \frac{x_a}{2}$ and $y_a = -\frac{a}{4}$. Hence,

$$\begin{aligned} f(x_p) \geq f\left(\frac{x_a}{2}\right) &= -\frac{\frac{a}{2} - y_a}{x_a} \cdot \frac{x_a}{2} + \frac{a}{2} = -\frac{a}{4} + \frac{y_a}{2} + \frac{a}{2} = \frac{a}{4} + \frac{y_a}{2} \\ &\geq \frac{a}{4} - \frac{a}{8} = \frac{a}{8}. \end{aligned}$$

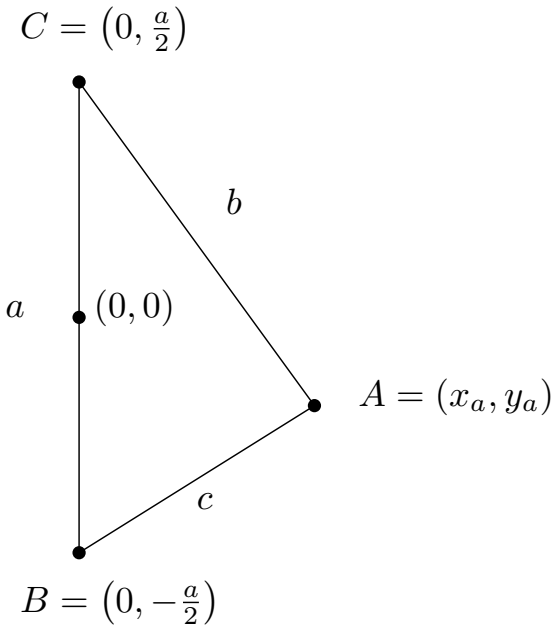

Figure 4.7: A visualization of $\triangle_{ABC}$, where a is parallel to the y-axis.

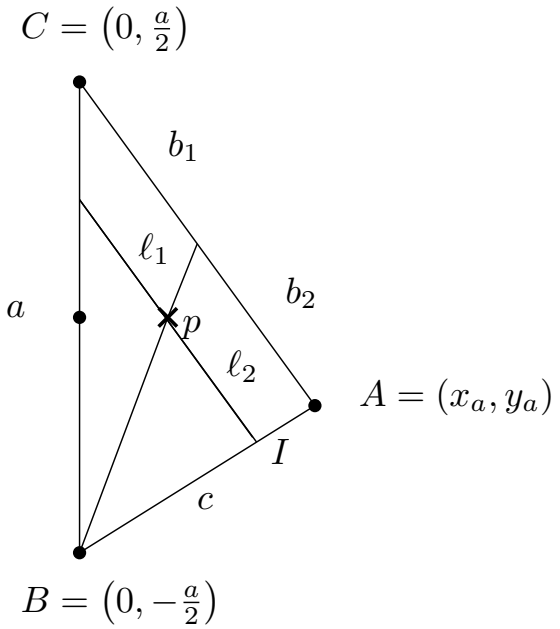

Figure 4.8: A visualization of the second case, where the line ℓ is parallel to the edge b.

Hence, we can center a line segment of length $\frac{a}{4}$ on p that is parallel to a and completely contained in $\triangle_{ABC}$.

Next, we consider $y_a < -\frac{a}{4}$. As long as $f(x_p) \geq \frac{a}{8}$, we can use the same arguments as for the first case. Thus, we assume that $f(x_p) < \frac{a}{8}$. Now, we show that there is a line segment parallel to b with midpoint p and with a length of at least $\frac{\text{diam}\cdot\sqrt{3}}{8}$. Since $y_a < -\frac{a}{4}$, we conclude $b \geq \frac{3}{4}a$. Let ℓ be the line segment parallel to y that is completely contained in $\triangle_{ABC}$. Note that p does not need to be the midpoint of ℓ, see also Figure 4.8 for a depiction. Let $I = (x_i, y_i)$ be the intersection of ℓ and c. We conclude $x_i \geq \frac{x_a}{2}$ since p lies on the perpendicular bisector centered in the midpoint of c with x-coordinate $\frac{x_a}{2}$ and ℓ is parallel to b. Hence, $|BI| \geq \frac{c}{2}$. Applying the intercept theorem yields $\frac{|BI|}{c} = \frac{\ell}{b}$ and $\frac{\ell}{b} \geq \frac{1}{2} \iff \ell \geq \frac{b}{2}$. Since $b \geq \frac{3}{4}a$, we conclude $\ell \geq \frac{3}{8}a$.

It remains to estimate the position of p on c to give a final bound for α. We use the line segment starting in B, leading through p and intersecting b to split b and ℓ into b_1 and b_2 as well as ℓ_1 and ℓ_2. See also Figure 4.8 for a visualization. As $f(x_p) < \frac{a}{8}$, we obtain $b_1 \geq \frac{3}{8}a$ and $b_2 \leq \frac{5}{8}a$. Hence, also $b_1 \geq \frac{3}{8}b$. The intercept theorem yields $\ell_1 \geq \frac{3}{8}\ell$ and hence, we can center a line segment of length $\frac{2}{3}\ell$ in p that is parallel to b. Finally, we conclude $\frac{2}{3}\ell \geq \frac{2}{3} \cdot \frac{3}{8}a = \frac{a}{4} \geq \frac{\text{diam}\cdot\sqrt{3}}{8}$. All in all, we obtain that we can always center a line segment of length at least $\frac{\text{diam}\cdot\sqrt{3}}{8}$ in p that is completely contained in Q. ■

Subsequently, we state general properties of SECs to derive a lower bound on the constant β afterward.

Theorem 4.10 — [31]. Let C be the SEC if a point set S. Then, either there are two points $P, Q \in S$ on the circumference of C such that the line segment $\overline{PQ}$ is a diameter of C or there are three points $P, Q, R \in S$ on the circumference of S such that the center c of C is inside the acute-angled $\triangle_{PQR}$. Furthermore, C is always unique.

Lemma 4.20 A robot r_i that is at distance d_{target} from the center c of its SEC moves at least a distance of $\frac{d_{\text{target}}}{2}$ towards c.

Proof. Let c be the center of r_i's SEC C. We rotate and translate the coordinate system such that $c = (0,0)$ and r_i is located at $(x_i, 0)$, i.e., r_i is at distance x_i from c. Additionally, we define a to be the radius of C. Observe first that $a \leq 1$ since there must be at least one robot $r_j = (x_j, y_j)$ with $x_j \leq 0$ on the boundary of C (see Theorem 4.10) and r_j is at distance at most 1 from r_i.

Now, let $r_k = (x_k, y_k)$ be a robot in r_i's neighborhood and m_k be the midpoint between r_i and r_k. We will prove that m_k is at distance at most $\frac{1}{2}$ from the point $\left(\frac{x_i}{2}, 0\right)$. First of all, we calculate the coordinates of m_k: $m_k = \left(\frac{1}{2} \cdot (x_i + x_k), \frac{1}{2} \cdot y_k\right)$. The distance between $\left(\frac{x_i}{2}, 0\right)$ and m_k is $\sqrt{\frac{1}{4} \cdot x_k^2 + \frac{1}{4} y_k^2}$. Basic calculus yields $\sqrt{\frac{1}{4} \cdot x_k^2 + \frac{1}{4} y_k^2} \leq \frac{1}{2} \iff -1 \leq x_k \leq 1$ and $-\sqrt{1 - x_k^2} \leq y_k \leq \sqrt{1 - x_k^2}$. Since $a \leq 1$, the inequalities for x_k and y_k are fulfilled. Hence, r_i can move at least half its distance towards c. ■

The combination of Lemmata 4.19 and 4.20 yields the following theorem.

Theorem 4.11 GTC is $\left(\sqrt{3}/8, 1/2\right)$-contracting.

GTC can be generalized to d-dimensions by moving robots towards the center of the smallest enclosing hypersphere of their neighborhood. We denote the resulting protocol by d-GTC.

Algorithm 3 d-GTC (view of robot r_i)

1: $C_i(t) :=$ smallest enclosing hypersphere of $N_i(t)$
2: $c_i(t) :=$center of $C_i(t)$
3: $\forall r_j \in N_i(t) : m_j :=$ midpoint between r_i and r_j
4: D_j : hypersphere with radius $\frac{1}{2}$ centered at m_j
5: seg := line segment $\overline{p_i(t), c_i(t)}$
6: $A := \bigcap_{r_j \in N_i(t)} D_j \cap$ seg
7: $x :=$ point in A closest to $c_i(t)$
8: $\text{target}_i^{\text{GTC}}(t) := x$

For the analysis, we first state two general properties of smallest enclosing hyperspheres.

Lemma 4.21 — [61]. Let S be the smallest enclosing hypersphere of a set of points $P \subset \mathbb{R}^m$. The center c of S is a convex combination of at most $m+1$ points on the surface of S.

Lemma 4.22 — [64]. Let T be a set of points on the boundary of some hypersphere H with center c. H is the smallest enclosing hypersphere of T if and only if c is a convex combination of the points in T.

Next, we state that the center of the smallest enclosing hypersphere is, in general, $\frac{\sqrt{2}}{8}$-centered, in contrast to $\frac{\sqrt{3}}{8}$ for $d = 2$.

Lemma 4.23 The center of the smallest enclosing hypersphere of a convex polytope $Q \subset R^d$ is $\frac{\sqrt{2}}{8}$-centered.

Proof. Let C denote the smallest enclosing hypersphere (SEH) of Q and c_i its center. We need to distinguish two cases: either two points are located on the boundary of SEH or (at least) 3. It is well known, that c_i is a convex combination of at most $d+1$ points on the boundary of C (Lemma 4.21). Those points form a simplex S. From Lemma 4.22, it follows that C is also the SEH of S since C is a circumsphere of S and c_i is inside of S.

In case, there are only two points on the boundary of C, they must be the endpoints of a diameter of C. Hence, the center of the SEH is 1-centered since it equals the midpoint of the two points that define the diameter.

Otherwise, S consists of at least 3 points. We take two points of S that have the maximal distance of all points in S and denote those points as B and C. Additionally, we take an arbitrary third point of S and call it A. The points A, B and C form a triangle $\triangle_{ABC}$. Moreover, a, b and c denote the edges of $\triangle_{ABC}$.

Let r denote the radius of C. The proof aims to show that there exists a line segment ℓ with midpoint c_i that is parallel to a, b or c and has a length of $\alpha \cdot d$, where d denotes the diameter of Q (we will determine the concrete value for α shortly). Recall that a is the longest edge of $\triangle_{ABC}$. Since C is also the SEH of S, it holds $r \leq a \cdot \sqrt{\frac{d}{2\cdot(d+1)}}$ and thus also $\frac{r}{\sqrt{\frac{d}{2\cdot(d+1)}}} \leq a$ (Theorem 4.8). Additionally, it holds $r \geq \frac{d}{2}$. Hence, $\frac{d}{2\cdot\sqrt{\frac{d}{2\cdot(d+1)}}} \leq a$. Now, we rotate the coordinate system, such that $B = \left(0, -\frac{a}{2}\right), C = \left(0, \frac{a}{2}\right)$ and $A = (x_a, y_a)$. See Figure 4.7 for a visualization of the setting. We now consider $y_a \leq 0$, and the arguments for $y_a > 0$ can be derived analogously with swapped roles of B and C.

Observe first, that $c_i = (x_{c_i}, 0)$ since x_i is located on the intersection of the perpendicular bisector hyperplanes of a, b and c. Additionally, it holds $x_{c_i} \leq \frac{x_a}{2}$ since both the midpoints of b and c have the x-coordinate $\frac{x_p}{2}$ and the parts of bisector hyperplanes inside of S have monotonically decreasing x-coordinates.

Now, we distinguish two cases: $-\frac{a}{4} \leq y_a \leq 0$ and $y_a < -\frac{a}{4}$. In the first case, we prove that there exists a line segment parallel to a with its midpoint in c_i that has a length of at least $\frac{a}{4}$. We model the edge b as a linear function $f(x) = -\frac{\frac{a}{2}-y_a}{x_a} \cdot x + \frac{a}{2}$. The value $f(x_{c_i})$ is minimized for $x_p = \frac{x_a}{2}$ and $y_a = -\frac{a}{4}$. Hence,

$$\begin{aligned} f(x_{c_i}) \geq f\left(\frac{x_a}{2}\right) &= -\frac{\frac{a}{2}-y_a}{x_a} \cdot \frac{x_a}{2} + \frac{a}{2} = -\frac{a}{4} + \frac{y_a}{2} + \frac{a}{2} = \frac{a}{4} + \frac{y_a}{2} \\ &\geq \frac{a}{4} - \frac{a}{8} \\ &= \frac{a}{8}. \end{aligned}$$

Hence, we can center a line segment of length $\frac{a}{4}$ on c_i that is parallel to a and completely contained in $\triangle_{ABC}$.

Next, we consider $y_a < -\frac{a}{4}$. As long as $f(x_{c_i}) \geq \frac{a}{8}$ holds, we can use the same arguments as for the first case. Thus, we assume that $f(x_{c_i}) < \frac{a}{8}$. Now, we show that there is a line segment parallel to b with midpoint c_i and a length of at least $\frac{d\cdot\sqrt{2}}{8}$. Since $y_a < -\frac{a}{4}$, it holds $b \geq \frac{3}{4}a$. Let ℓ be the line segment parallel to y that is completely contained in $\triangle_{ABC}$. Note that c_i does not need to be the midpoint of ℓ, see also Figure 4.8 for a depiction. Let $I = (x_i, y_i)$ be the intersection of ℓ and c. We conclude $x_i \geq \frac{x_a}{2}$ since p lies on the perpendicular bisector centered in the midpoint of c with x-coordinate $\frac{x_a}{2}$ and ℓ is parallel to b. Hence, $|BI| \geq \frac{c}{2}$. Applying the intercept theorem yields $\frac{|BI|}{c} = \frac{\ell}{b}$ and $\frac{\ell}{b} \geq \frac{1}{2} \iff \ell \geq \frac{b}{2}$. Since $b \geq \frac{3}{4}a$, we conclude $\ell \geq \frac{3}{8}a$.

It remains to estimate the position of c_i on c to give a final bound for α. We use the line segment starting in B, leading through c_i and intersecting b to split b and ℓ into b_1 and b_2 as well as ℓ_1 and ℓ_2. See also Figure 4.8 for a visualization. As $f(x_{c_i}) < \frac{a}{8}$, we obtain $b_1 \geq \frac{3}{8}a$ and $b_2 \leq \frac{5}{8}a$. Hence, also $b_1 \geq \frac{3}{8}b$. The intercept theorem yields $\ell_1 \geq \frac{3}{8}\ell$ and hence, we can center a line segment of length $\frac{2}{3}\ell$ in p that is parallel to b. Finally, we conclude $\frac{2}{3}\ell \geq \frac{2}{3} \cdot \frac{3}{8}a = \frac{a}{4} \geq \frac{d}{\sqrt{\frac{d}{2\cdot(d+1)}}\cdot 8} \geq \frac{d\cdot\sqrt{2}}{8}$ since $\lim_{d\to\infty} \sqrt{\frac{d}{2\cdot(d+1)}} = \frac{1}{\sqrt{2}}$. All in all, we obtain that we can always center a line segment of length at least $\frac{d\cdot\sqrt{2}}{8}$ in c_i that is completely contained in Q. ■

Lemma 4.24 A robot r_i that is at distance d_{target} from the center c of its SEC moves at least a distance of $\frac{d_{\text{target}}}{2}$ towards c.

Proof. Let c be the center of r_i's SEC C. We rotate and translate the coordinate system such that $c = (0,0)$ and r_i is located at $(x_i, 0)$, i.e., r_i is at a distance of x_i of c. Additionally, we define a to be the radius of C. Observe first that $a \leq 1$ since there must be at least one robot $r_j = (x_j, y_j)$ with $x_j \leq 0$ on the boundary of C and r_j is at a distance of at most 1 of r_i.

Now, let $r_k = (x_k, y_k)$ be a robot in r_i's neighborhood and m_k be the midpoint between r_i and r_k. We will prove that m_k is at a distance of at most $\frac{1}{2}$ of the point $\left(\frac{x_i}{2}, 0\right)$. First of all, we calculate the coordinates of m_k: $m_k = \left(\frac{1}{2} \cdot (x_i + x_k), \frac{1}{2} \cdot y_k\right)$. The distance between $\left(\frac{x_i}{2}, 0\right)$ and m_k is $\sqrt{\frac{1}{4} \cdot x_k^2 + \frac{1}{4} y_k^2}$. Basic calculus yields $\sqrt{\frac{1}{4} \cdot x_k^2 + \frac{1}{4} y_k^2} \leq \frac{1}{2} \iff -1 \leq x_k \leq 1$ and $-\sqrt{1 - x_k^2} \leq y_k \leq \sqrt{1 - x_k^2}$. Since $a \leq 1$ holds, the inequalities for x_k and y_k are fulfilled. Hence, r_i can move at least half its distance towards c. ■

As a consequence, we derive that d-GTC is (α, β)-contracting and thus, also λ-contracting in any dimension d.

Theorem 4.12 d-GTC is $\left(\sqrt{2}/8, 1/2\right)$-contracting.

4.5 Conclusion & Outlook

In this chapter, we introduced the classes of contracting and λ-contracting protocols considering robots in the $\mathcal{OBLOT}_1^C$ and $\mathcal{OBLOT}_1^F$ models. For both models, these were the first high-dimensional protocols for disoriented robots with limited visibility and provable runtime guarantees. The class of contracting protocols gathers in time $\mathcal{O}\left(n^{\log(d)} \cdot \Delta\right)$ while a lower bound of $\Omega(n \cdot \Delta)$ is known. Further research could investigate the gap between the upper and the lower time bound. Our intuition is that the real answer is a time bound of $\Theta(n \cdot \Delta)$ for every dimension d. Intuitively, the dependence on the dimension d is an artifact from the analysis ($d-2$ projections) and the important parameters in the runtime are only the number of robots and the diameter Δ. Furthermore, the two-dimensional contracting protocol MOVE-ON-BISECTOR (MOB) gathers in time $\mathcal{O}(n)$ which is in many cases severely faster than the bound of $\mathcal{O}(n \cdot \Delta)$ [49]. As a consequence, a further research question is: What are the properties of continuous robot formation protocols that gather in time $\mathcal{O}(n)$? Is even a bound of $\mathcal{O}(\Delta)$ achievable? Our research about λ-contracting protocols in the $\mathcal{OBLOT}_1^F$ model gives a hint that a dependence only on the diameter might be achievable since the class of λ-contracting protocols solves GATHERING in $\Theta\left(\Delta^2\right)$ rounds. Also regarding λ-contracting protocols and GATHERING of disoriented robots with limited visibility in the $\mathcal{OBLOT}_1^F$ model in general, several open questions remain. First of all, we did not aim to optimize the constants in the runtime. Thus, the upper runtime bound of $\frac{256 \cdot \pi \cdot \Delta^2}{\lambda^3}$ seems to be improvable. Moreover, one major open question remains unanswered: Is it possible to solve GATHERING of disoriented robots with limited visibility in the $\mathcal{OBLOT}_1^F$ model in $\mathcal{O}(\Delta)$ rounds? We could get closer to the answer: if there is such a protocol, it must compute target points regularly outside of the convex hulls of robots' neighborhoods. All λ-contracting protocols are slow in the configuration where the positions of the robots form a regular polygon with a side length equal to the viewing range. In Chapter 6, we will show that this configuration can be gathered in time $\mathcal{O}(\Delta)$ by a protocol where each robot moves as far as possible along the angle bisector between its neighbors (leaving the local convex hull). However, this protocol cannot perform well in general. See Figure 4.9 for the *alternating star*, a configuration where this protocol is always worse compared to any protocol that computes target points inside of local convex hulls. Figure 4.9 gives a hint that every protocol that performs well for the regular polygon cannot perform equally

well in the alternating star. Thus, we conjecture that $\Omega\left(\Delta^2\right)$ is a lower bound for every protocol that considers oblivious and disoriented robots with limited visibility in the $\mathcal{OBLOT}_1^F$ model.

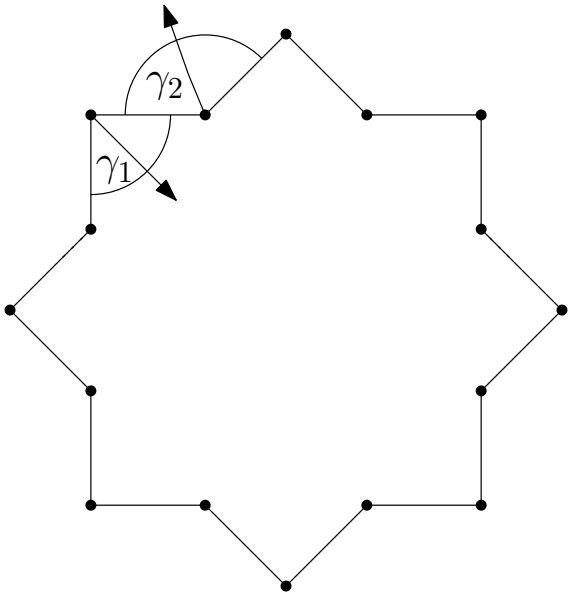

Figure 4.9: The robots at γ_1 observe a regular square, the robots at γ_2 a regular octagon. Given that each robot moves along the angle bisector between its neighbors and leaves its local convex hull, the radius of the global SEC decreases slower than in any λ-contracting protocol.

5. CHAIN-FORMATION in the $\mathcal{LUMI}$ Model

The following two chapters deal with formation problems for robots in the $\mathcal{LUMI}$ model combined with a (closed) chain. In this chapter, we study the CHAIN-FORMATION problem. The goal of the CHAIN-FORMATION problem is to arrange all robots in a straight line between the two stationary outer robots of the chain. In other words, the length of the chain should be equal to the distance between two stationary outer robots. Earlier work on this problem introduced the HOPPER protocol [91] in a setting comparable to the $\mathcal{LUMI}$ model (visible states seen by neighboring robots). The protocol can be easily implemented in the $\mathcal{LUMI}$ model. The HOPPER protocol can achieve a $\sqrt{2}$-approximation of the optimal chain length in $\mathcal{O}(n)$ rounds under the $\mathcal{F}$SYNC scheduler. One drawback of the HOPPER protocol is that the outer robots r_0 and r_{n-1} need to be distinguishable. One robot remains idle, while the other one generates certain states that are swapped along the chain. We study improvements of the HOPPER protocol with better approximation guarantees and we show how the assumption of distinguishable outer robots can be removed. While these are already improvements for the CHAIN-FORMATION problem, our modifications are also a basis for a linear time gathering protocol for closed chains we present afterward in Chapter 6. The results of this chapter are based on the first part of the following journal article.

> **2023** (with J. Harbig, D. Jung, T. Knollmann and F. Meyer auf der Heide) "Gathering a Euclidean Closed Chain of Robots in Linear Time and Improved Algorithms for Chain-Formation" In: *Theoretical Computer Science*, cf. [28].

5.1 Contribution

We show improved protocols for the CHAIN-FORMATION problem inspired by the HOPPER protocol. The HOPPER protocol uses so-called *run sequences* that allow a locally sequential movement of the robots. Such run sequences are realized by forwarding a state, called *run state*, along the chain. Only robots with a run state perform a movement, all other robots stay idle. If the robot r_i has a run state in round t, it will execute its movement and forward the run state either to r_{i+1} or r_{i-1} (depending on the direction of the run sequence) such that in round $t+1$ one of the neighboring robots has the run state and executes a movement. Pipelining of the run sequences, i.e., starting a new run sequence every constant number of rounds, leads to a fast runtime. The run sequences can be easily implemented in the $\mathcal{LUMI}$ mode with help of two lights: one light to indicate an active run state and a second light to remember a run state of the previous round such that a run sequence can keep a unique direction along the chain. Our ε-HOPPER protocol achieves a $(1+\varepsilon)$-approximation of the optimal configuration for an arbitrary constant $\varepsilon \in (0,1]$ in $\mathcal{O}(n/\varepsilon)$

epochs. Prior to this work, only a $\sqrt{2}$-approximation was known [91]. Both the HOPPER and the ε-HOPPER protocol consider outer robots that can be distinguished since run sequences start only at one outer robot while the other remains stationary. We show that this restriction can be removed by increasing the viewing range of each robot to 2. This way, the ε-2-HOPPER protocol starts run sequences at both outer robots. As soon as two run sequences started at different ends of the chain meet, the viewing range of 2 allows them to coordinate their movements. The ε-2-HOPPER protocol achieves a $(1+\varepsilon)$-approximation of the optimal configuration while having a runtime of $\mathcal{O}(n/\varepsilon)$ epochs. We first present the protocols designed for the $\mathcal{F}$SYNC scheduler in Section 5.4. Afterward, we show a two-step synchronization approach to transfer the protocols to the $\mathcal{S}$SYNC and $\mathcal{A}$SYNC schedulers with help of a constant number of additional lights in Section 5.6. Besides the improved approximation factors, these are the first protocols with provable runtime guarantees regarding CHAIN-FORMATION under the $\mathcal{A}$SYNC scheduler. Table 5.1 contains a comprehensive comparison of our results to previous ones about the CHAIN-FORMATION problem.

Model	Protocol	Identical Outer Robots	Approximation	Runtime
$\mathcal{OBLOT}_1^F$, $\mathcal{OBLOT}_1^S$	GTM [25, 60, 88]	yes	$(1+\varepsilon)$ (converging)	$\Omega\left(n^2 \log\left(1/\varepsilon\right)\right)$ $\mathcal{O}\left(n^2 \log\left(n/\varepsilon\right)\right)$
$\mathcal{OBLOT}_1^C$	MOB [49], GTM [20]	yes	optimal	$\Theta(n)$
$\mathcal{LUMI}_1^F$	HOPPER [91]	no	$\sqrt{2}$ (non converging)	$\Theta(n)$
$\mathcal{LUMI}_1^A$	ε-HOPPER	no	$(1+\varepsilon)$ (non converging)	$\Theta(n/\varepsilon)$
$\mathcal{LUMI}_2^A$	ε-2-HOPPER	yes	$(1+\varepsilon)$ (non converging)	$\Theta(n/\varepsilon)$

Table 5.1: Results about CHAIN-FORMATION of this chapter compared to existing work. The results of this chapter are marked in gray.

5.2 Model Recap and Preliminaries

Next, we briefly recap the robot model and formalize the problem statement. We study the $\mathcal{LUMI}_1^A$ (ε-HOPPER) and the $\mathcal{LUMI}_2^A$ (ε-2-HOPPER) models in combination with an open chain of robots. The most important features of the model required for this chapter are summarized in Table 5.2, and more details can be found in Chapter 2.

Protocol	Time	Dimension	Viewing Range	Orientation	Chain
ε-HOPPER	$\mathcal{A}$SYNC	≥ 1	1	disoriented	yes (open)
ε-2-HOPPER	$\mathcal{A}$SYNC	≥ 1	2	disoriented	yes (open)

Table 5.2: A summary of the most important model details for the ε-HOPPER and ε-2-HOPPER protocols.

Problem Statement. In the CHAIN-FORMATION problem, the outer robots r_0 and r_{n-1} are stationary (they do not move at all). The inner robots are mobile and aim to reduce length of the chain. More formally, the inner robots have to move such that $L(t) = \|p_0(t) - p_{n-1}(t)\|_2$, i.e., the inner robots have to arrange themselves on the line segment connecting r_0 and r_{n-1}. We say that, for a constant $\varepsilon > 0$, a $(1+\varepsilon)$-approximation of the optimal configuration is reached at time t, if $L(t) \leq (1+\varepsilon) \cdot \|p_0(t) - p_{n-1}(t)\|_2$.

5.3 Run Sequences and Movement Operations

A run state (introduced first in [91]) is a visible state (implemented by lights) that is passed along the chain in a fixed direction associated with it. Robots with a run state perform a movement operation while robots without do not. The movement is sequentialized in a way that in round t the robot r_i executes a movement operation (and neither r_{i-1} nor r_{i+1}), the robot r_{i+1} in round $t+1$ and so on. The movement of a run state along the chain is denoted as a *run sequence* and is visualized in Figure 5.1.

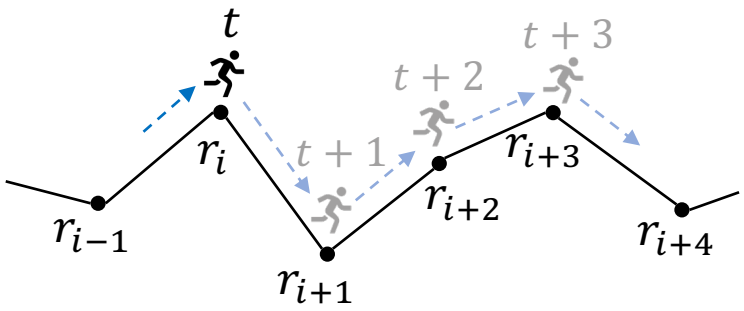

Figure 5.1: A run state at r_i in round t is passed in its direction along the chain, i.e., it is located at r_{i+x} in round $t+x$.

Since robots are moving sequentially, any robot with a run state does not have to care about the movements of its neighbors as they do not change their positions. There is one exception: two run states with opposite directions might be located at two neighboring robots. In this case, the two robots move simultaneously (see Section 5.3.1). All movements must ensure that the distance between neighbors stays less or equal to 1 (the connectivity range). A run sequence can be implemented with two lights: ℓ_{run} and ℓ_{prev}. The light ℓ_{run} indicates that a robot has a run state in the current round and ℓ_{prev} is active if a robot had a run state in the last round. Thus, the run sequence keeps a fixed direction along the chain; robots that have not activated the light ℓ_{prev} and see one neighbor with an active light ℓ_{run} will take over the run state in the next round by activating ℓ_{run}. After completing the movement based on the run state, ℓ_{run} is switched off, and ℓ_{prev} is activated such that the robot does not take over the same run state in the next round. In case run sequences have different directions along the chain, a few more lights are needed for the synchronization. For instance, a robot may simultaneously have two run states of run sequences heading in opposite directions. The existence of two run states at a single robot is indicated via an additional light ℓ_{double}. Robots that have not activated ℓ_{prev} and detect that both neighbors have activated ℓ_{run} will take over both run states by activating ℓ_{double}. Hence, also robots with an active light ℓ_{prev} need to take over a run state if they detect that a neighbor has activated ℓ_{double}. For ease of description and due to space constraints, we use the high-level concept of a run state to describe our protocols and do not describe precisely how the handling with lights works. We use the following notation to speak about run sequences. For a robot r_i, $run(r_i,t) = true$ if r_i has a run state in round t. Additionally, $run(N_i(t)) = \{r_j \in N_i(t) |\ run(r_j,t) = true\}$. Let κ denote an arbitrary run sequence. $r(\kappa,t)$ denotes the robot that has the run state of the run sequence κ in round t and $r(\kappa,t+1)$ denotes the robot that will have the run state of the run sequence κ in round $t+1$ (κ's direction).

5.3.1 Movement Operations

This section describes the basic movement operations that robots can execute when having a run states. The concrete protocols are composed of these movement operations. All protocols ensure that at most two directly neighboring robots move in the same round to maintain the connectivity of the chain. This is done by allowing the existence of only two patterns of run states at neighboring robots: Either r_i and neither r_{i-1} nor r_{i+1} has a run state (*isolated run sequence*) or r_i and r_{i+1} have run states heading in each other's direction while r_{i-1} and r_{i+2} do not have run states (*joint run pair*). All other patterns, especially sequences of length at least 3 of neighboring robots having run states, are avoided by all protocols. More formally:

Definition 5.1 A run sequence κ is called an *isolated run sequence* in round t if $r(\kappa,t) = r_i$ and $run(r_{i-1},t) = run(r_{i+1},t) = false$. Two run sequences κ_1 and κ_2 with $r(\kappa_1,t) = r_i$ and $r(\kappa_2,t) = r_{i+1}$ are called a *joint run pair* in round t in case $r(\kappa_1,t+1) = r_{i+1}$, $r(\kappa_2,t+1) = r_i$ and $run(r_{i-1},t) = run(r_{i+2},t) = false$.

For robots with a run state, there are three kinds of movement operations, the *merge*, the *shorten* and the *hop*. The purpose of the merge is to reduce the number of robots in the chain. It is executed by a robot r_i if its neighbors have a distance of at most 1. In this case, r_i is not necessary for the connectivity of the chain and can be safely removed. Removing r_i means that it moves to the position of its next neighbor in the direction of the run sequence, the robots merge their neighborhoods, and both continue to behave as a single robot. The execution of a merge stops a run sequence. The goal of a shorten is to reduce the length of the chain by moving a robot to the midpoint of its neighbors. Intuitively, if the angle between vectors of r_i pointing to its neighbors is not too large, the shorten can reduce the length of the chain by a constant. The execution of a shorten also stops a run sequence. In case no progress (in terms of reducing the number of robots or the length of the chain) can be made locally, a hop is executed. The purpose of a hop is to exchange two neighboring vectors in the chain. Thus, run sequences are associated with run vectors that are swapped along the chain until they find a position at which progress (in terms of shortens or merges) can be made. For each of the three operations, there is also a *joint* one (joint hop, joint shorten and joint merge), which is a similar operation executed by a joint run pair. We continue by introducing the formal definitions of all movement operations. For the ease of notation, we assume for an isolated run sequence κ that $r(\kappa,t) = r_i$ and $r(\kappa,t+1) = r_{i+1}$.

Hop and Joint hop

Consider the isolated run sequence κ. If r_i executes a *hop*, it swaps the vectors $w_i(t)$ and $w_{i+1}(t)$ with its movement. Formally, $p_i(t+1) = p_{i+1}(t) - w_i(t)$. The run sequence continues in its direction. A *joint hop* is a similar operation executed by a joint run pair κ_1, κ_2 located at robots r_i and r_{i+1}. The vectors $w_i(t)$ and $w_{i+2}(t)$ are swapped such that the new positions are $p_i(t+1) = p_{i-1}(t) + w_{i+2}(t)$ and $p_{i+1}(t+1) = p_{i+2}(t) - w_i(t)$. Due to the different viewing ranges of the protocols in this thesis (the hop will also be used in Chapter 6), a joint hop is handled slightly differently. In the first variant, both run sequences continue in their directions and skip the next robot, i.e., in round $t+1$, $r(\kappa_1,t+1) = r_{i+2}$ and $r(\kappa_2,t+1) = r_{i-1}$. See Figure 5.2 for a visualization. In the way the concrete protocols are designed, this is only possible if the viewing range is at least 3, as the neighboring robots, r_{i-1} and r_{i+2} need to be able to detect that r_i and r_{i+1} execute a joint hop. The operation is split into two rounds if the viewing range is at most 2 (such as in the ε-HOPPER and ε-2-HOPPER protocols). First, run sequences continue without skipping the next robot: $r(\kappa_1,t+1) = r_{i+1}$ and $r(\kappa_2,t+1) = r_i$. Additionally, r_i and r_{i+1} activate a light ℓ_{joint} to remember this situation. In the following round $t+1$, r_i and r_{i+1} do not execute a movement and the run sequences are passed to r_{i-1} and r_{i+2}.

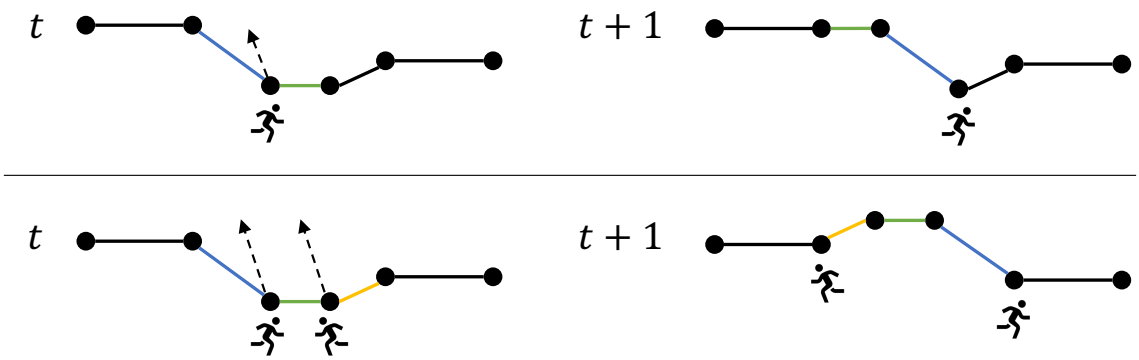

Figure 5.2: Visualization of a hop (above) and a joint hop (below) in the variant with sufficient viewing range.

Shorten and Joint shorten

In the *shorten*, a robot r_i with an isolated run sequence moves to the midpoint between its neighbors: $p_i(t+1) = \frac{1}{2} \cdot p_{i-1}(t) + \frac{1}{2} \cdot p_{i+1}(t)$. The run sequence stops. In a *joint shorten* executed by two robots r_i and r_{i+1} with a joint run pair, the vector $v = p_{i+2}(t) - p_{i-1}(t)$ is subdivided into three parts of equal length. The new positions are $p_i(t+1) = p_{i-1}(t) + \frac{1}{3} \cdot v$ and $p_{i+1}(t+1) = p_{i+2}(t) - \frac{1}{3} \cdot v$. Both run sequences are stopped after executing a joint shorten. See Figure 5.3 for a visualization of both operations.

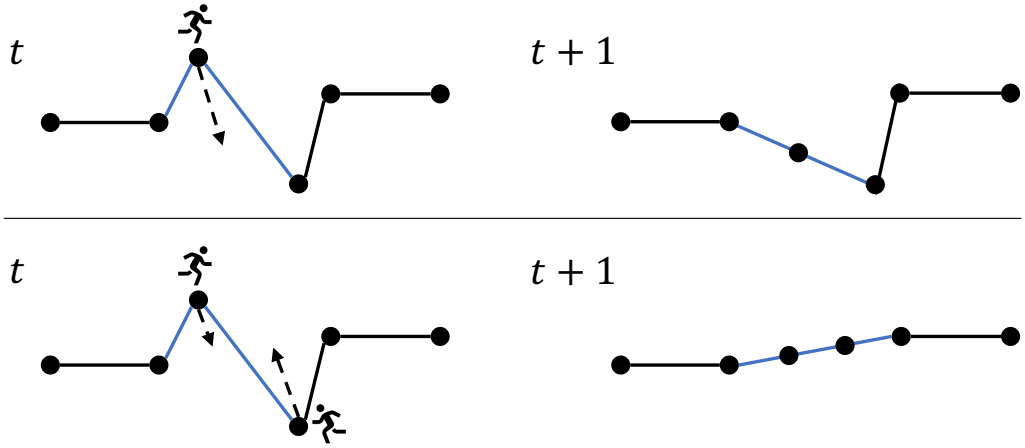

Figure 5.3: A shorten (above) and a joint shorten (below).

Merge and Joint merge

Consider an isolated run sequence κ with $r(\kappa,t) = r_i$ and $r(\kappa,t+1) = r_{i+1}$. If r_i executes a *merge*, it moves to $p_{i+1}(t)$. Afterward, the robots r_i and r_{i+1} merge such that their neighborhoods are identical, and they continue to behave like a single robot. In the *joint merge*, the robots r_i and r_{i+1} both move to $\frac{1}{2}p_i(t) + \frac{1}{2}p_{i+1}(t)$. Afterward, the robots merge there such that they behave as a single robot in the future. All run sequences that participate in a merge or a joint merge are immediately stopped. See Figure 5.4 for a visualization of both operations.

5.4 Protocols for the $\mathcal{F}$SYNC Scheduler

This section is dedicated to the ε-HOPPER and the ε-2-HOPPER protocols. Section 5.4.1 deals with the ε-HOPPER protocol: it uses the same assumptions as the original HOPPER protocol, i.e., robots have a viewing range of 1 and new run sequences are only generated at r_0 while r_{n-1} does nothing at all. The ε-2-HOPPER protocol is presented in Section 5.4.2. In contrast to ε-HOPPER, both r_0

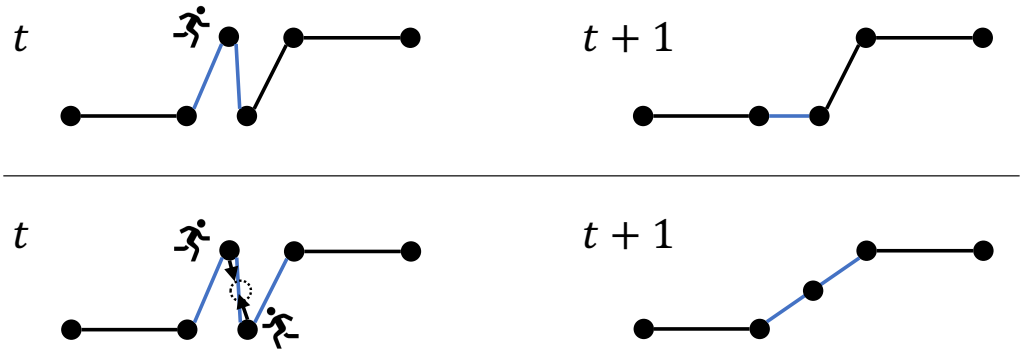

Figure 5.4: A merge (above) and a joint merge (below).

and r_{n-1} participate such that run sequences are generated at both ends of the chain. To handle run sequences with opposite movement directions, a viewing range of 2 is required. Section 5.5 contains the combined analysis of both protocols, proving the linear runtime and the approximation guarantees.

5.4.1 Description of the ε-HOPPER Protocol

The ε-HOPPER protocol consists of three operations for robots with run states. For ease of notation, we define $\psi := 2 \cdot \sin^{-1}\left(\frac{1}{1+\varepsilon}\right)$.

1. If $\|p_{i-1}(t) - p_{i+1}(t)\|_2 \leq 1$, r_i: merge.
2. Else, if $\alpha_i(t) \leq \psi$, r_i: shorten.
3. Else, r_i: hop.

The difference to the original HOPPER protocol is marginal. Only the angle size on which decisions are based differs. While in the original HOPPER protocol a shorten is executed if $\alpha_i(t) \leq \frac{\pi}{2}$, this angle is replaced by $2 \cdot \sin^{-1}\left(\frac{1}{1+\varepsilon}\right)$ in the ε-HOPPER protocol. Observe that for $(1+\varepsilon) = \sqrt{2}$ both protocols coincide: $2 \cdot \sin^{-1}(\frac{1}{\sqrt{2}}) = \frac{\pi}{2}$. New run sequences are started all 3 rounds by r_0.

Next, we give pseudocode for the ε-HOPPER protocol that explains how the sequential movement can be implemented in the $\mathcal{LUMI}$ model, see Algorithm 4. We assume that each robot has 4 lights, $\ell_{run}, \ell_{prev}, \ell_{stop}$ and ℓ_c. The purpose of the lights ℓ_{prev} and ℓ_{run} is to move run sequences along the chain as described in Section 5.3. The additional light ℓ_{stop} is needed to stop run sequences after merges and shortens. Suppose that r_i has a run state and r_{i+1} is supposed to take over the run state. Due to the viewing range of 1, r_{i+1} cannot see which operation r_i executes. Hence, it has to take over the run state optimistically. If r_i executes a shorten or a merge, it activates the light ℓ_{stop} such that r_{i+1} gets to know in the next round that the run sequence is supposed to stop. The lights ℓ_{run}, ℓ_{prev} and ℓ_{stop} have two colors (they are either activated or switched off). The third light, ℓ_c has the color set $C_c = \{0,1,2\}$ and is used as a round counter. Every round, the color of ℓ_c is incremented (after color 2, it starts again with color 0). For the light ℓ_c we use the simplified notation $\ell_c = i$ to indicate that ℓ_c has color i and the notation ℓ_c++ is used to indicate that the color is incremented. For the other three lights, we use the notation $\ell_{run} = 1$ to indicate that the light ℓ_{run} is activated and $\ell_{run} = 0$ for an inactive light (similar for ℓ_{prev} and ℓ_{stop}). Note that in the original $\mathcal{LUMI}$ model, each robot only has one light; our model can be easily emulated in the $\mathcal{LUMI}$ model by using $3 \cdot 2 \cdot 2 \cdot 2 = 24$ colors. Even better, not all of these colors are needed because the lights ℓ_{run}, ℓ_{prev} and ℓ_{stop} are never activated at the same time, and the light ℓ_c also has only a single color in every round. Thus, only 12 colors are needed to simulate the 4 lights used in the ε-HOPPER in the $\mathcal{LUMI}$ model. We denote by $p_i(t) \leftarrow x$ that the target point x is computed and the robot r_i

moves to x in its MOVE operation. For ease of description, we represent the handling of the lights ℓ_{prev}, ℓ_{run} and ℓ_{stop} by an automaton. See Figure 5.5.

Algorithm 4 ε-HOPPER (executed from the view of robot r_i)

if $\ell_{run} = 1$ **then**
  if $r_{i\pm1}$ has activated ℓ_{stop} **then** ▷ The run sequence has stopped
    runStopped $\leftarrow$ true
  else
    if $\|p_{i-1}(t) - p_{i+1}(t)\|_2 \leq 1$ **then**
      Merge with the next robot in the direction of the run sequence ▷ Merge
      runEnds $\leftarrow$ true
    else if $\alpha_i(t) \leq \sin^{-1}\left(\frac{1}{1+\varepsilon}\right)$ **then**
      $p_i(t) \leftarrow \frac{1}{2}p_{i-1}(t) + \frac{1}{2}p_{i+1}(t)$ ▷ Shorten
      runEnds $\leftarrow$ true
    else
      $p_i(t) \leftarrow p_{i+1}(t) - w_i(t)$ ▷ Hop
if $r_i = r_0$ **and** $\ell_c = 2$
  or $\ell_{prev} = 0$ **and** $\ell_{stop} = 0$ **and** $r_{i\pm1}$ has set $\ell_{run} = 1$ **then**
  takeRun $\leftarrow$ true ▷ Start new or take over run sequence if $\ell_{prev} = \ell_{stop} = 0$
UpdateLights-ε-HOPPER(), see Figure 5.5
ℓ_c++ ▷ Increment light for round counting

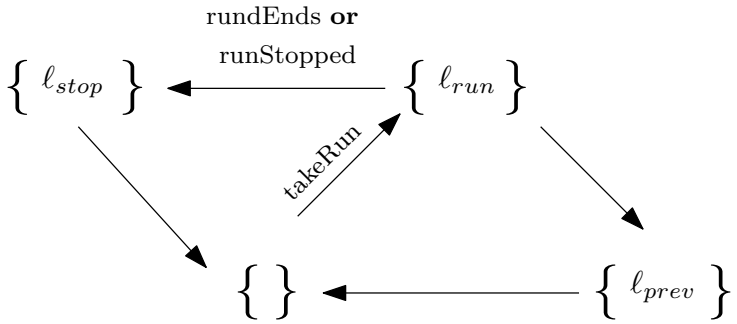

Figure 5.5: The figure describes the handling of ℓ_{run}, ℓ_{prev} and ℓ_{stop} in UPDATELIGHTS-ε-HOPPER(). The states represent active lights. The transitions refer to variables in Algorithm 4.

5.4.2 Description of the ε-2-HOPPER Protocol

The following section is dedicated to the ε-2-HOPPER protocol in which robots have a viewing range of 2. In contrast to the ε-HOPPER protocol, the ε-2-HOPPER does not distinguish the robots r_0 and r_{n-1}. Instead, both are regularly generating new run sequences which adds new movement operations that are necessary to handle joint run pairs to the protocol: joint hops, joint shortens and joint merges. The ε-2-HOPPER protocol consists of two parts: The movement based on run states and the generation of new run sequences. For better readability, we present the protocol incrementally and omit the details about the handling of lights.

Movement

The idea of the movement is as follows. Robots that have an isolated run sequence behave identically to the ε-HOPPER protocol (Section 5.4.1). For better readability, define $\psi := 2 \cdot \sin^{-1}\left(\frac{1}{1+\varepsilon}\right)$. The movement of a robot r_i with an isolated run sequence can be summarized as follows:

1. If $\|p_{i-1}(t) - p_{i+1}(t)\|_2 \leq 1$, r_i: merge.
2. Else, if $\alpha_i(t) \leq \psi$, r_i: shorten.
3. Else, r_i: hop.

After some time, run sequences of opposite ends of the chain meet at two neighboring robots (joint run pairs). Robots that are part of a joint run pair execute similar operations as robots with isolated run sequences in the respective joint variant. In some cases, a priority rule is used (instead of a joint shorten only one robot executes a shorten and the other run sequence stops). In summary, two robots r_i and r_{i+1} that are part of a joint run pair move as follows:

1. If $\|p_{i-1}(t) - p_{i+2}(t)\|_2 < 2$, both: joint merge.
2. Else, if $\alpha_i(t) \leq \psi$ and $\alpha_{i+1}(t) \leq \psi$ both: joint shorten
3. Else, if $\alpha_i(t) \leq \psi$, r_i: shorten.
4. Else, if $\alpha_{i+1}(t) \leq \psi$, r_{i+1}: shorten.
5. Else, if $\measuredangle(w_i(t), -w_{i+2}(t)) \leq \psi$ both: joint shorten.
6. Else, both: joint hop.

Generation of run sequences

The generation of new run sequences is implemented as follows. Initially, both r_0 and r_{n-1} generate new run sequences every 4 rounds. Robots can count the number of rounds with an additional light ℓ_c and color set $C_c = \{0,1,2,3\}$. The color is incremented in every round. Furthermore, the role of generating new run sequences is passed along the chain with the help of an additional light ℓ_{exp}. Initially, only r_0 and r_{n-1} have activated the light ℓ_{exp}. As soon as r_1 executes a hop based on a run sequence that has been started at the other end of the chain, r_1 activates the light ℓ_{exp} and takes the responsibility to generate new run sequences. The intuition for this procedure is that the vector belonging to the run sequence is *non-conflicting*, i.e., it has a small angle to all other vectors and will not cause any shorten in the future. Henceforth, this approach prevents the vector from being passed along the chain again. Finally, all robots have activated the light ℓ_{exp}, and the $(1+\varepsilon)$-approximation is achieved.

5.5 Analyses

First, we state an upper bound on the number of shortens.

Lemma 5.1 There are at most $2n-1$ shorten operations such that one participating vector has a length of less than $\frac{1}{2}$.

Proof. A robot r_i only executes a shorten in case $\|p_{i-1}(t) - p_{i+1}(t)\|_2 > 1$. Otherwise, a merge is executed. Hence, a shorten in which a participating vector has a length of less than $\frac{1}{2}$ always results in two vectors with a length of at least $\frac{1}{2}$. Initially, there are at most $n-1$ vectors of length less than $\frac{1}{2}$. New vectors of length less than $\frac{1}{2}$ can only be created by merges. Since a merge is executed at most n times, the number of such shortens is upper bounded by $2n-1$. ■

Lemma 5.2 Assume that a robot r_i executes a shorten and both $\|w_i(t)\|_2 \geq \frac{1}{2}$ and $\|w_{i+1}(t)\|_2 \geq \frac{1}{2}$. Then $L(t+1) \leq L(t) - \frac{\varepsilon}{2}$. Additionally, if a joint run pair executes a joint shorten in round t, it holds also $L(t+1) \leq L(t) - \frac{\varepsilon}{\varepsilon+1} \leq L(t) - \frac{\varepsilon}{2}$.

Proof. Denote $a := \|w_i(t)\|_2$, $b := \|w_{i+1}(t)\|_2$ and $c := \|w_i(t) + w_{i+1}(t)\|_2$. As r_i moves to $\frac{1}{2}p_{i-1}(t) + \frac{1}{2}p_{i+1}(t)$, the length of the chain decreases with the movement of r_i by $a+b-c$. By the law of cosines $c = \sqrt{a^2+b^2-2ab\cdot\cos(\alpha_i(t))}$. Therefore, for fixed a,b the value of c is maximized for $\alpha_i(t) = 2\cdot\sin^{-1}\left(\frac{1}{1+\varepsilon}\right)$. Thus,

$$\begin{aligned} a+b-c &\geq a+b-\sqrt{a^2+b^2-2ab\cdot\cos\left(2\cdot\sin^{-1}\left(\frac{1}{1+\varepsilon}\right)\right)} \\ &= a+b-\sqrt{a^2+b^2-2ab\cdot\left(1-2\cdot\frac{1}{(1+\varepsilon)^2}\right)} \end{aligned} \tag{5.1}$$

Equation (5.1) holds since $\cos\left(2\cdot\sin^{-1}(x)\right) = 1-2x^2$. Using that $\frac{1}{2} \leq a,b \leq 1$ and $a+b-\sqrt{a^2+b^2-2ab\cdot\left(1-2\cdot\frac{1}{(1+\varepsilon)^2}\right)}$ is minimized for $a=b=\frac{1}{2}$, we conclude

$$\begin{aligned} a+b-c &\geq a+b-\sqrt{a^2+b^2-2ab\cdot\left(1-2\cdot\frac{1}{(1+\varepsilon)^2}\right)} \\ &\geq a+b-\sqrt{\frac{1}{4}+\frac{1}{4}-2\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\left(1-2\cdot\frac{1}{(1+\varepsilon)^2}\right)} \\ &= a+b-\sqrt{\frac{1}{2}-\frac{1}{2}+\frac{1}{(1+\varepsilon)^2}} \\ &= a+b-\frac{1}{1+\varepsilon} \\ &\geq 1-\frac{1}{1+\varepsilon} \\ &= \frac{\varepsilon}{\varepsilon+1} \\ &\geq \frac{\varepsilon}{2} \end{aligned}$$

For a joint shorten, we observe: Let κ_1 and κ_2 be the two run sequences with $r(\kappa_1,t) = r_i$ and $r(\kappa_2,t) = r_{i+1}$. The involved vectors are $w_i(t)$, $w_{i+1}(t)$ and $w_{i+2}(t)$. For simplicity, $a := \|w_i(t)\|_2$, $b := \|w_{i+1}(t)\|_2$, $c := \|w_{i+2}(t)\|_2$ and $d := \|w_i(t) + w_{i+1}(t) + w_{i+2}(t)\|_2$. The length of the chain decreases by $a+b+c-d$. By the triangle inequality, it follows $d \leq \|w_i(t)+w_{i+2}(t)\|_2 + \|w_{i+1}(t)\|_2$. Thus, $a+b+c-d \geq a+b-\|w_i(t)+w_{i+2}(t)\|_2$. Now, we can apply the same calculations as above (since both $\|w_i(t)\|_2 \geq \frac{1}{2}$ and $\|w_{i+2}(t)\|_2 \geq \frac{1}{2}$ because all run vectors have a length of the least $\frac{1}{2}$) and obtain $L(t+1) \leq L(t) - \frac{\varepsilon}{2}$. ■

Next, we introduce the notion of *conflicting* vectors, i.e., vectors that still could cause shortens.

Definition 5.2 A vector $w_i(t)$ is called to be *conflicting* if $\measuredangle(w_i(t), w_j(t)) \leq 2\cdot\sin^{-1}\left(\frac{1}{1+\varepsilon}\right)$ for any $j \in \{1,\ldots,n-1\} \setminus i$. Otherwise, the vector $w_i(t)$ is called to be *non-conflicting*.

Definition 5.3 Define by $r_{nc}(t)$ the largest index of a robot such that $w_{r_{nc}(t)}(t)$ is a conflicting vector and by $\ell_{nc}(t)$ the smallest index of a robot such that $w_{\ell_{nc}(t)}(t)$ is a conflicting vector.

The following lemma is crucial, stating that the number of the non-conflicting vectors at the end of the chain (close to r_n) does never decrease.

Lemma 5.3 Let u be a spatial vector and $V = u_1, \ldots, u_k$ a set of spatial vectors. If $\measuredangle(u,u_i) > 2 \cdot \sin^{-1}\left(\frac{1}{1+\varepsilon}\right)$ for all $i = 1, \ldots, k$, then, for any positive linear combination s of vectors from V, $\measuredangle(u,s) > 2 \cdot \sin^{-1}\left(\frac{1}{1+\varepsilon}\right)$ holds.

Proof. Note that $\measuredangle(u,u_i)$ determines the smaller of the two angles created by anchoring u_i at the terminal point of u. In the following, we use $\measuredangle'(u,u_i)$ to determine the standard angle between the two vectors u and u_i. Now, observe that $\measuredangle(u,u_i) \geq 2 \cdot \sin^{-1}\left(\frac{1}{1+\varepsilon}\right)$ translates to $\measuredangle'(u,u_i) \leq \pi - 2 \cdot \sin^{-1}\left(\frac{1}{1+\varepsilon}\right)$.

$$\begin{aligned} \measuredangle'(u,u_i) &\leq \pi - 2 \cdot \sin^{-1}\left(\frac{1}{1+\varepsilon}\right) \\ \Leftrightarrow \frac{u \cdot u_i}{\|u\|_2 \cdot \|u_i\|_2} &\geq \cos\left(\pi - 2 \cdot \sin^{-1}\left(\frac{1}{1+\varepsilon}\right)\right) \\ \Leftrightarrow u \cdot u_i &\geq -\cos\left(2 \cdot \sin^{-1}\left(\frac{1}{1+\varepsilon}\right)\right) \cdot \|u\|_2 \cdot \|u_i\|_2 \end{aligned}$$

Next, assume that $s = c_1 \cdot u_i + c_2 \cdot u_j$. We need to check

$$\begin{aligned} \frac{u \cdot s}{\|u\|_2 \cdot \|s\|_2} &\geq -\cos\left(2 \cdot \sin^{-1}\left(\frac{1}{1+\varepsilon}\right)\right) \\ \Leftrightarrow u \cdot (c_1 \cdot u_i + c_2 \cdot u_j) &\geq -\cos\left(2 \cdot \sin^{-1}\left(\frac{1}{1+\varepsilon}\right)\right) \|u\|_2 \cdot \|c_1 \cdot u_i + c_2 \cdot u_j\|_2 \end{aligned}$$

This can be verified by the following observations:

$$\begin{aligned} u \cdot (c_1 \cdot u_i + c_2 \cdot u_j) &= c_1 \cdot u \cdot u_i + c_2 \cdot u \cdot u_j \\ &\geq -\cos\left(2 \cdot \sin^{-1}\left(\frac{1}{1+\varepsilon}\right)\right) \cdot \|u\|_2 \cdot (c_1 \cdot \|u_i\|_2 + c_2 \cdot \|u_j\|_2) \end{aligned}$$

and by the triangle inequality, it follows

$$c_1 \cdot \|u_i\|_2 + c_2 \cdot \|u_j\|_2 \geq \|c_1 \cdot u_i + c_2 \cdot u_j\|_2.$$

■

Lemma 5.4 In the ε-HOPPER protocol, neither the merge nor the shorten operation increase $r_{nc}(t)$. In the ε-2-HOPPER protocol, the merge, joint merge, shorten and the joint shorten operations do not decrease $\ell_{nc}(t)$ or increase $r_{nc}(t)$.

Proof. This follows immediately from Lemma 5.3 since (joint) shortens and (joint) merges are positive linear combinations of the vectors $w_1(t), \ldots, w_{n-1}(t)$. ■

Lemma 5.5 At most $3n - 1 + \frac{2(n-1)}{\varepsilon}$ run sequences are needed until all vectors are non-conflicting.

Proof. The sum of merges and joint merges is upper bounded by n. Additionally, the number of shortens and joint shortens is upper bounded by $2n - 1 + \frac{2 \cdot (n-1)}{\varepsilon}$ (Lemmata 5.1 and 5.2). Lastly, no new conflicting vectors are created according to Lemma 5.4. ■

Lemma 5.6 Consider a configuration, where all vectors are non-conflicting in round t. Then, $L(t) \leq (1+\varepsilon) \cdot \|p_0(t) - p_{n-1}(t)\|_2$.

Proof. For ease of notation, define $d := \|p_0(t) - p_{n-1}(t)\|_2$. Sort the vectors $w_1(t), \ldots, w_{n-1}(t)$ ascending with respect to their angle to the line connecting r_0 and r_{n-1}. This results in the configuration $C' = (a_0(t), \ldots, a_k(t))$ with length $|C'|$. $|C'| = L(t)$; however, the shapes of the configurations might differ significantly. The polygon defined by the configuration C' and the line segment connecting the two stationary endpoints is convex (due to the sorting of the vectors). Now, take $a_0(t)$ and $a_k(t)$ and enlarge both of them such that they form a triangle together with the line segment connecting r_0 and r_{n-1}. The two other legs of the triangle are denoted as a and b, and γ is the angle between a and b. Since C' is convex, it holds $|C'| \leq a+b$. As $\measuredangle(a_0(t), a_k(t)) > 2 \cdot \sin^{-1}\left(\frac{1}{1+\varepsilon}\right)$, it holds $\gamma \geq 2 \cdot \sin^{-1}\left(\frac{1}{1+\varepsilon}\right)$. Now, observe that for a fixed d, the sum $a+b$ is maximized for $\gamma = 2 \cdot \sin^{-1}\left(\frac{1}{1+\varepsilon}\right)$ and $a = b$. By observing that in an isosceles triangle it holds $d = 2 \cdot a \cdot \sin\left(\frac{\gamma}{2}\right)$, we obtain:

$$\begin{aligned} d &= 2 \cdot a \cdot \sin\left(\frac{2 \cdot \sin^{-1}\left(\frac{1}{1+\varepsilon}\right)}{2}\right) \\ &= \frac{a+b}{1+\varepsilon} \end{aligned}$$

All in all, we can conclude $a+b \leq (1+\varepsilon) \cdot d$. ■

Theorem 5.1 For an arbitrary constant $\varepsilon \in (0,1]$, both the ε-HOPPER and the ε-2-HOPPER protocol achieve a $(1+\varepsilon)$-approximation of the optimal configuration regarding the CHAIN-FORMATION problem in $\mathcal{O}(n/\varepsilon)$ rounds in the $\mathcal{LUMI}_1^F$ and $\mathcal{LUMI}_2^F$ models. The light of each robot requires a constant number of colors.

Proof. By Lemma 5.5, at most $3n - 1 + \frac{2(n-1)}{\varepsilon}$ run sequences are required to make all vectors non-conflicting. Since every 4 rounds a new run sequence is generated and each run sequence stops after at most $n-1$ rounds, the total number of rounds is $4 \cdot (3n - 1 + \frac{2 \cdot (n-1)}{\varepsilon}) + n - 1 \in \mathcal{O}(n/\varepsilon)$. Lemma 5.6 gives us that $L(t) \leq (1+\varepsilon) \cdot \|p_0(t) - p_{n-1}(t)\|_2$ holds. ■

5.6 Synchronization for the $\mathcal{S}$SYNC and $\mathcal{A}$SYNC Schedulers

Next, we present a two-stage synchronization approach to transfer the ε-HOPPER and the ε-2-HOPPER protocol to the $\mathcal{A}$SYNC scheduler. First, we present a synchronization approach for the $\mathcal{S}$SYNC scheduler in Section 5.6.1. Afterward, we add a second synchronization for the $\mathcal{A}$SYNC scheduler in Section 5.6.4.

5.6.1 Basic Synchronization Procedure for the $\mathcal{S}$SYNC Scheduler

The main synchronization idea is to simulate a 1-fair execution of the $\mathcal{S}$SYNC scheduler *locally*, i.e., in each neighborhood, each robot is activated at most once before the first robot becomes active a second time. In practice, however, robots might become active multiple times, but they wait if they detect that a robot in their neighborhood has not yet executed its protocol. A synchronization can guarantee this based on the α-synchronizer [8], a synchronization protocol for asynchronous message passing systems. As the robots do not exchange messages, we only need the main synchronization idea of the protocol that can be implemented as follows: the robots currently count rounds in each protocol with a light ℓ_c. The number of colors $|C_c|$ differs between the protocols but is always a small constant. The light ℓ_c is so far only used to keep the distance between newly started run sequences: New run sequences are started all $|C_c|$ rounds. We assign an additional role

to the light ℓ_c. For ease of notation, we say each robot r_i has a counter c_i with a numerical value out of the set C_c. We refer to the counter of robot r_i with c_i. The counters are used to synchronize the robots as follows: A robot r_i only executes any operation (movement and switching of lights) if $c_{i-1} \geq c_i$ and $c_i \leq c_{i+1}$. Otherwise, it waits until it wakes up in the future and the condition is fulfilled. Observe that this procedure ensures $|c_i - c_{i+1}| \leq 1$ for any two neighbors r_i and r_{i+1} [8]. However, this does not prevent a robot from becoming active twice before the direct neighbor becomes active. Hence, it still can happen that a run sequence gets lost before the next robot takes it over. The information that the previous robot had a run state is, however, still encoded in ℓ_{prev}. Assume a robot r_i checks whether it should take a run state of r_{i-1}. Two cases need to be checked: either $c_{i-1} = c_i$ and r_{i-1} has activated ℓ_{run} or $c_{i-1} = c_i + 1$ and r_{i-1} has activated ℓ_{prev}. In both cases, r_i activates ℓ_{run}. This procedure can be generalized: For every light, ℓ_i that needs to be seen by a neighboring robot, a second light ℓ_{i-prev} is added. After switching ℓ_i off, the light ℓ_{i-prev} is activated such that the neighboring robot is always able to see the information transported by ℓ_i. The synchronization ensures the following: If a robot increments its counter c for the k-th time, its position and lights are equivalent to those obtained by an execution of the same protocol under a 1-fair $\mathcal{S}$SYNC scheduler. Thus, the position and lights are equivalent to those after the k-th round of execution under the $\mathcal{F}$SYNC scheduler.

5.6.2 ε-HOPPER in $\mathcal{S}$SYNC

A robot simultaneously shows its active lights of the current and the last epoch. We introduce for every light ℓ_i except those only used for the synchronization (ℓ_c) a second light with the suffix $-prev$ (i.e., ℓ_{i-prev}). When Update-Lights is called, it assigns $\ell_{i-prev} \leftarrow \ell_i$ for all such lights before updating them. Whenever $\ell_i(r)$ (the light ℓ_i of robot r) is accessed by r', it accesses the value $\ell_i(r) \wedge (\ell_c(r) = \ell_c(r')) \vee \ell_{i-prev}(r) \wedge (\ell_c(r) = \ell_c(r') + 1)$ instead. Note, that $\ell_{run-prev}$ is equivalent to ℓ_{prev} and $\ell_{prev-prev}$ is never accessed, therefore these two lights can be removed. The implementation is contained in Algorithm 5. For the sake of clarity, we omit the concrete description of the handling of ℓ_{i-prev} lights.

Algorithm 5 ε-HOPPER-SSYNC (executed from the view of robot r_i)

if $\exists r_j \in$ Neighborhood with $c_j = c_i - 1$ **then**
  return ▷ Neighbor is in previous epoch
if $\ell_{run} = 1$ **then**
  if $r_{i\pm1}$ has activated ℓ_{stop} **then** ▷ The run sequence has stopped
    runStopped $\leftarrow$ true
  else
    if $\|p_{i-1}(t) - p_{i+1}(t)\|_2 \leq 1$ **then**
      Merge with next robot in direction of the run sequence ▷ Merge
      runEnds $\leftarrow$ true
    else if $\alpha_i(t) \leq \sin^{-1}\left(\frac{1}{1+\varepsilon}\right)$ **then**
      $p_i(t) \leftarrow \frac{1}{2}p_{i-1}(t) + \frac{1}{2}p_{i+1}(t)$ ▷ Shorten
      runEnds $\leftarrow$ true
    else
      $p_i(t) \leftarrow p_{i+1}(t) - w_i(t)$ ▷ Hop
if $r_i = r_0$ **and** $\ell_c = 2$
  or $\ell_{prev} = 0$ **and** $\ell_{stop} = 0$ **and** $r_{i\pm1}$ has set $\ell_{run} = 1$ **then**
  takeRun $\leftarrow$ true ▷ Start new or take over run sequence if $\ell_{prev} = \ell_{stop} = 0$
Update $-prev$ lights
UpdateLights-ε-HOPPER(), see Figure 5.5
ℓ_c++ ▷ Increment light for round counting

Next, we analyze the ε-HOPPER-SSYNC protocol and prove that the lights and the position of a robot after incrementing c the k-th time are always identical to its position and lights under a 1-fair execution of the $\mathcal{S}$SYNC scheduler.

Lemma 5.7 Consider a 1-fair execution of the $\mathcal{S}$SYNC scheduler. After epoch k, the lights and the positions of all robots match exactly with the lights and positions of the robots executing the related ε-HOPPER protocol under the $\mathcal{F}$SYNC scheduler after round k.

Proof. We show that all actions are solely based on the global snapshot taken in the LOOK phase when the first robot became active the k-th time, which is the same information a robot has in the related $\mathcal{F}$SYNC protocol. No robot will return in line 2 of Algorithm 5 such that $c(r)$ denotes the number of times r was active. The way lights are accessed is based on the state of the lights at the beginning of epoch k. When a robot r_i was active before $r_{i\pm1}$ and performed a hop, $r_{i\pm1}$ cannot access the previous position of r_i when it becomes active. However, this does not change any actions because this information is only used when movement operations are executed, and two direct neighbors never move in the same epoch. ■

Lemma 5.8 When r_i increments its epoch counter the k-th time under a (not necessarily 1-fair) $\mathcal{S}$SYNC scheduler, it is located at the same position and has activated the same lights as under the 1-fair $\mathcal{S}$SYNC scheduler after epoch k.

Proof. We prove the claim by induction over the value of the counter c of robot r_i. For $c = 0$, the claim holds. Let us assume it holds for the first (arbitrary but fixed) k increments of c. When a robot r_i becomes active but a neighbor $r_{i\pm1}$ did not increment its epoch timer the k-th time, $\ell_c(r_{i\pm1}) = \ell_c(r_i) - 1$ and r_i returns in line 2 of Algorithm 5 without performing any action. In this case, the claim is true because r_i does not increment its epoch counter. Otherwise, both neighbors are in a state they can have during epoch e of Lemma 5.7, therefore r_i performs the same action compared to Lemma 5.7 and this lemma holds still after incrementing the $k+1$-th time. ■

Lemma 5.9 The value of the epoch counter c_i is always in the interval $[k, k+n]$ for all i in $\{0, \ldots, n-1\}$ where k denotes the current epoch.

Proof. This follows directly from the fact that neighbors always have a difference of at most 1. ■

Theorem 5.2 For an arbitrary constant $\varepsilon \in (0, 1]$, the following holds: The ε-SSYNC-HOPPER protocol achieves a $(1+\varepsilon)$-approximation of the optimal configuration (regarding CHAIN-FORMATION) in $\mathcal{O}(n/\varepsilon)$ epochs in the $\mathcal{LUMI}_1^S$ model. The protocol can be implemented with a constant number of colors.

Proof. With the introduced synchronization, our robots act as they would under a 1-fair $\mathcal{S}$SYNC scheduler (Lemma 5.8). The newly introduced lights and their handling ensure that robots perform the same actions with the 1-fair $\mathcal{S}$SYNC scheduler as in the $\mathcal{F}$SYNC setting (Lemma 5.7). Hence, after each robot performed $\mathcal{O}(n)$ actions, the ε-HOPPER-SSYNC reached the same or a better configuration compared to ε-HOPPER regarding the CHAIN-FORMATION problem. From Lemma 5.9 we know that after $f(n) \in \mathcal{O}(n)$ epochs each robot performed at least $f(n)$ actions. ■

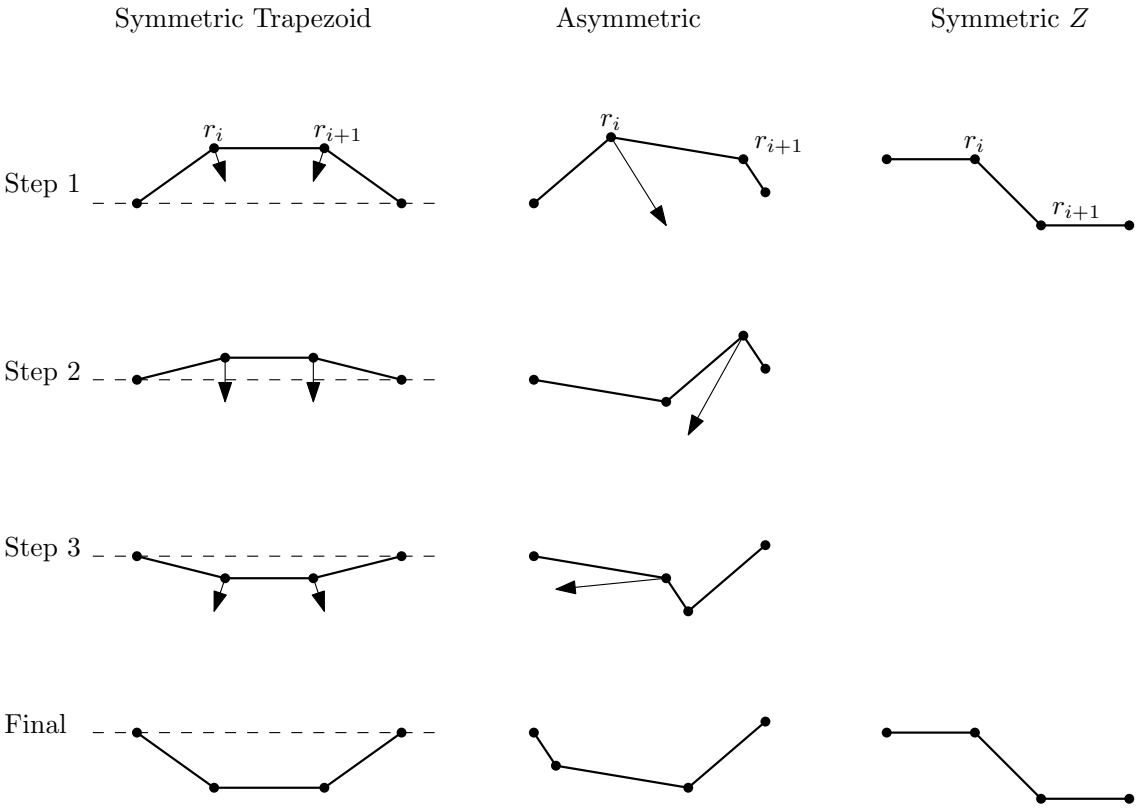

Figure 5.6: Visualization of modified joint hop operations.

5.6.3 ε-2-HOPPER in $\mathcal{S}$SYNC

The additional challenge when transferring the ε-2-HOPPER protocol to the $\mathcal{S}$SYNC scheduler is the handling of joint run pairs in which two robots together need to execute a movement operation. The joint shorten and joint merge do not need further synchronization: The result of a simultaneous movement is equivalent to a sequential movement, and the operations keep the connectivity of the chain. In a joint hop, however, two robots may have a distance larger than 1 when one performs its hop while the other is not active. This can be prevented via a three-step execution of joint hops. Consider two robots r_i and r_{i+1} that are part of a joint run pair and want to execute a joint hop. There are three kinds of configurations to distinguish, the symmetric trapezoid, the symmetric Z, and an asymmetric configuration (see Figure 5.6).

Asymmetric Configuration: In asymmetric configurations, one robot can be determined as a leader, either with a smaller angle $\alpha_i(t)$ or the robot with a smaller vector length $\|w_i(t)\|_2$. After determining the leader, the two robots perform hops (the operation for isolated run sequences) alternately. In steps 1 and 3, the leader performs a hop; in step 2, the other robot. This leads to the same outcome as a joint hop in $\mathcal{F}$SYNC.

Symmetric Z: No movement is necessary since the participating vectors are identical. Both run sequences continue without movement of the participating robots.

Symmetric Trapezoid: In the symmetric trapezoid, no leader can be determined since $\alpha_i(t) = \alpha_{i+1}(t)$ and $w_i(t) = w_{i+2}(t)$. Therefore, the operation is executed as follows (each substep ensures connectivity): Let l be the line segment connecting r_{i-1} and r_{i+2}. In the first step, each robot computes the intersection of its angle bisector (intersecting the angle of size at most π) and l. Afterward, it moves half the distance along its angle bisector towards the intersection point. The second step moves onto the mirrored position (mirror axis l). The third step moves on the outer angle bisector (intersecting the angle of size at least π) until it doubles its distance to l. When one robot (e.g. r_{i+1}) performs a step before the other one (r_i), r_i needs to compute the initial position of r_{i+1}. This is possible because all steps are reversible operations.

We need to prove that the three-step execution of the joint hop still maintains the connectivity of the chain.

Lemma 5.10 Two robots performing a joint hop in a symmetric trapezoid configuration always stay at a distance of at most 1 from each other.

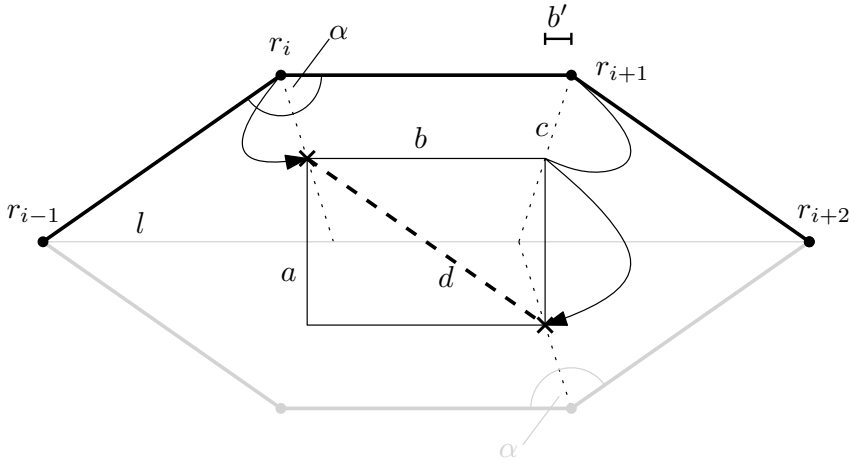

Figure 5.7: Symmetric Joint Hop - worst case distance d between r_i and r_{i+1}.

Proof. Let r_i and r_{i+1} be the robots which perform the joint hop, let l be the line between r_{i-1} and r_{i+2}. In a worst-case configuration, both robots have a distance of 1 from their direct neighbors. Both robots perform their operation in 3 steps, but they can be up to one step apart. There are two relevant cases we need to consider. First, assume that r_i performs the first step and r_{i+1} remains in its initial position. Then, all points on the bisector between r_i and l are within a range of 1 to r_{i+1}. Second, assume that r_i performed the first step and r_{i+1} already the second. The resulting configuration is shown in Figure 5.7. The respective bisectors are drawn with dotted lines. The crosses mark the position of r_i and r_{i+1}, d denotes the distance between both robots. We derive d by computing the side length of the shown rectangle; its side lengths are denoted by a and b. Further, c denotes the length of the bisector segment between r_i and l.

We obtain the following formulas: $c = \frac{\sin(\alpha)}{\sin(\alpha/2)} = 2\cos(\alpha/2), a = \sin(\alpha) \iff \sin(\alpha/2) = \frac{a}{c}, b' = \cos^2(\frac{\alpha}{2}) \iff \cos(\frac{\alpha}{2}) = \frac{2b'}{c}, b = 1-2b' = 1-2\cos^2(\frac{\alpha}{2})$ and $d^2 = a^2+b^2 = \sin^2(\alpha)+(1-2\cos^2(\frac{\alpha}{2}))^2 = 1+\sin^2(\alpha)-4\cos^2(\frac{\alpha}{2})+4\cos^4(\frac{\alpha}{2})$. Next, we prove $d = 1$ for all α.

$$
\begin{aligned}
0 \stackrel{?}{=} & -4\cos^2(\alpha/2) && +4\cos^4(\alpha/2) && +\sin^2(\alpha)\\
= & -2-2\cos(\alpha) && +4\cos^4(\alpha/2) && +\sin^2(\alpha)\\
= & -2-2\cos(\alpha) && +1+2\cos(\alpha)+\cos^2(\alpha) && +\sin^2(\alpha)\\
= & -2-2\cos(\alpha) && +1+2\cos(\alpha)+\frac{\cos(2\alpha)}{2}+\frac{1}{2} && +\sin^2(\alpha)\\
= & -2-2\cos(\alpha) && +1+2\cos(\alpha)+\frac{\cos(2\alpha)}{2}+\frac{1}{2} && -\frac{\cos(2\alpha)}{2}+\frac{1}{2}\\
= & \ 0
\end{aligned}
$$

The derivations follow from the sine and cosine power reduction identities. ■

Lemma 5.11 The ε-2-HOPPER-SSYNC protocol reaches the same outcome as the ε-2-HOPPER protocol.

Proof. From Lemmata 5.7 and 5.8, it follows that the lights of other robots can be accessed without differences to the $\mathcal{F}$SYNC case. The protocol for joint hops presented above ensures that the outcome is the same as in the respective $\mathcal{F}$SYNC joint hop. The same holds for joint shortens and joint merges. ■

Based on the previous lemmas, we conclude the following theorem about the ε-2-HOPPER-SSYNC protocol.

Theorem 5.3 For an arbitrary constant $\varepsilon \in (0,1]$, the following holds: The ε-2-HOPPER-SSYNC protocol achieves a $(1+\varepsilon)$-approximation of the optimal configuration (regarding CHAIN-FORMATION) in $\mathcal{O}(n/\varepsilon)$ epochs in the $\mathcal{LUMI}_2^S$ model. The protocol can be implemented with a constant number of colors.

5.6.4 From $\mathcal{S}$SYNC to $\mathcal{A}$SYNC

Finally, we add a synchronization procedure to transfer the protocols to the $\mathcal{A}$SYNC scheduler. We use an already known synchronization procedure for robots with unlimited visibility introduced in [42]. The authors introduce a protocol to transfer the execution of any $\mathcal{A}$SYNC protocol into a 1-fair execution under an $\mathcal{S}$SYNC scheduler by using a light with 5 colors. Robots get an additional light with the colors M (Moving), T (Trying), W (Waiting), S (Stopped), and F (Finished). Initially, all lights are initialized with the color T. The light M indicates that a robot is currently moving. Thus, robots that see other robots in their snapshot (taken in the LOOK phase) currently moving switch their light to W and wait until they take a snapshot only with non-moving robots. If this is the case (all other robots have the light T or the light S activated), a robot switches its light to M and executes its MOVE-operation. Robots that have finished their MOVE-operation (that wake up with an active light M) only proceed to the next light S if they see no other robot with light T. Similarly, robots with active light W switch to T if no moving robot is in their snapshot. Finally, robots with light S move to F if all other robots have an active light S; in the same way, robots switch from F to T, and the procedure starts again. This procedure can take $\mathcal{O}(n)$ epochs of the $\mathcal{A}$SYNC scheduler to execute a single 1-fair epoch of the $\mathcal{S}$SYNC scheduler. In our case, the robots only have limited visibility: We use the same synchronization protocol, but robots only care for the lights in their neighborhood. Similar to before, this does not result in a global 1-fair $\mathcal{S}$SYNC execution of the protocols. However, for any two neighbors, the positions afterward are the same as under a 1-fair $\mathcal{S}$SYNC scheduler. Additionally, the synchronization is much faster compared to [42] because the robots only have to care for a constant number of neighbors. Hence, for any two neighbors, only a constant number of epochs under the $\mathcal{A}$SYNC scheduler are needed to simulate a 1-fair $\mathcal{S}$SYNC execution of their movements. We denote the resulting protocols with the additional synchronization by ε-HOPPER-ASYNC and ε-2-HOPPER-ASYNC. Then, we can formulate the final theorem.

Theorem 5.4 For an arbitrary constant $\varepsilon \in (0,1]$, both the ε-HOPPER-ASYNC and the ε-2-HOPPER-ASYNC protocol achieve a $(1+\varepsilon)$-approximation of the optimal configuration regarding the CHAIN-FORMATION problem in $\mathcal{O}(n/\varepsilon)$ epochs in the $\mathcal{LUMI}_1^A$ and $\mathcal{LUMI}_2^A$ models. The light of each robot requires a constant number of colors.

5.7 Conclusion & Outlook

In this chapter, we studied the CHAIN-FORMATION problem and provided new insights compared to the existing literature. We introduced the ε-HOPPER and ε-2-HOPPER protocols, that are able to achieve $(1+\varepsilon)$-approximations of the optimal configuration in $\mathcal{O}(n/\varepsilon)$ epochs. The protocols consider the the $\mathcal{LUMI}_1^A$ (ε-HOPPER) and $\mathcal{LUMI}_2^A$ (ε-2-HOPPER) models. The new aspects of

the protocols are the improved approximation guarantees ($(1+\varepsilon)$ compared to $\sqrt{2}$), the transfer to the $\mathcal{A}$SYNC scheduler and the study of identical outer robots in the ε-2-HOPPER protocol.

Still, there are lots of open research questions regarding the CHAIN-FORMATION problem. While a $(1+\varepsilon)$-approximation seems reasonably good, the most intriguing open question is: Is there a protocol that solves the CHAIN-FORMATION problem *optimally* considering *discrete* time in the $\mathcal{LUMI}$ (or even in the $\mathcal{OBLOT}$) model? For both $\mathcal{LUMI}$ and $\mathcal{OBLOT}$, neither an impossibility result nor a protocol reaching an optimal solution is known. Furthermore, the current protocols for the $\mathcal{LUMI}$ model do not converge to the optimal solution. After reaching the approximations, no further improvements happen. Current solutions for the $\mathcal{OBLOT}$ model, in contrast, converge to the optimal solution in $\mathcal{O}(n^2 \cdot \log(n/\varepsilon))$ epochs [60, 88]. Hence, a further research question is: Are there protocols for the $\mathcal{LUMI}$ model that converge to the optimal configuration in $\mathcal{O}(n \cdot \log(n/\varepsilon))$ epochs (or at least reasonably faster compared to $\mathcal{OBLOT}$)?

The last question, we want to emphasize deals with the viewing ranges. Our ε-2-HOPPER protocol considers identical outer robots but uses a viewing range of 2. The viewing range of 2 is essential here to coordinate the movements of run sequences starting at opposite ends of the chain. However, it is not clear whether the viewing range of 2 is needed in general. Are there protocols that achieve a $(1+\varepsilon)$-approximation of the optimal configuration in linear time that consider identical outer robots but require a viewing range of only 1?

All in all, our conjecture is that there is a runtime difference between the $\mathcal{OBLOT}$ and the $\mathcal{LUMI}$ models. While the time bounds in the $\mathcal{OBLOT}$ model seem to depend quadratically on n, we only observe a linear dependence when considering the $\mathcal{LUMI}$ model. However, a formal exploration of the boundaries between the two models with a focus on time bounds remains an open problem.

6. GATHERING in the $\mathcal{LUMI}$ Model

One of the intriguing open research questions raised from Chapter 4 is about the existence of a linear time protocol to solve GATHERING of disoriented robots with limited visibility in the Euclidean plane. While there is a strong conjecture that such a protocol cannot exist for oblivious robots ($\mathcal{OBLOT}$), there is some evidence that such a protocol could be designed for luminous robots ($\mathcal{LUMI}$). The papers [2, 40] introduce linear time protocols for disoriented robots with limited visibility *located on a two-dimensional grid* that make use of locally visible states (and are hence implementable in the $\mathcal{LUMI}$ model). The protocol presented in [2] considers a closed chain of robots while [40] studies the standard limited visibility model, where robots can observe all nearby other robots. Both protocols use the ideas of run sequences, which we have already seen in Chapter 5. In this chapter, we show that a linear time protocol exists in the $\mathcal{LUMI}_4^A$ model for a closed chain of robots in the Euclidean plane. The results are based on the second part of the following journal article.

> **2023** (with J. Harbig, D. Jung, T. Knollmann and F. Meyer auf der Heide) "Gathering a Euclidean Closed Chain of Robots in Linear Time and Improved Algorithms for Chain-Formation" In: *Theoretical Computer Science*, cf. [28].

A preliminary version appeared in the conference proceedings of ALGOSENSORS 2021:

> **2021** (with J. Harbig, D. Jung, T. Knollmann and F. Meyer auf der Heide) "Gathering a Euclidean Closed Chain of Robots in Linear Time" In: *Proceedings of the 17th International Symposium on Algorithms and Experiments for Wireless Sensor Networks (ALGOSENSORS)*, Best Paper & Best Student Paper Award, cf. [27].

6.1 Contribution

We present a linear time protocol to solve GATHERING of a closed chain of disoriented robots with limited visibility in the Euclidean plane. More precisely, we study the $\mathcal{LUMI}_4^A$ model. The main idea of our protocol, called CLOSED-CHAIN-HOPPER (CC-HOPPER), is similar to the ε-HOPPER and ε-2-HOPPER protocols: to use run sequences for a locally sequential movement of the robots. In both the ε-HOPPER and the ε-2-HOPPER protocol, *open* chains are considered that have uniquely identifiable outer robots (based on their neighborhood). This asymmetry of the neighborhoods is used to start run sequences at the ends of the chain. Such outer robots do not exist in closed

chains. The CC-HOPPER protocol determines (whenever possible) locally unique robots (based on geometric properties of their neighborhoods) to generate new run sequences. One of the challenges in the closed chain is handling highly symmetric configurations. While it is possible to identify locally unique robots in every connected configuration on the grid (as in [40]), this is impossible in the Euclidean plane. We identify the class of isogonal configurations based on isogonal polygons by Grünbaum [74] and show that no locally unique robots can be determined in these configurations. We believe this characterization is of independent interest because highly symmetric configurations often cause a large runtime. For instance, the $\Omega(\Delta^2)$ lower bound of the GTC protocol as well as the lower bound for any λ-contracting protocol (Chapter 4) hold for an isogonal configuration [50]. See Table 6.1 for comparing our results to previous results about GATHERING with limited visibility.

Model	Space	Protocol	Orientation	Runtime
$\mathcal{OBLOT}_1^F$	$\mathbb{R}^2$	GTC [50]	disoriented	$\Theta(n^2)$
$\mathcal{OBLOT}_1^C$	$\mathbb{R}^2$	MOB [49]	disoriented	$\Theta(n)$
$\mathcal{OBLOT}_1^C$	$\mathbb{R}^2$	Contracting Protocols [97]	disoriented	$\Theta(n \cdot \Delta)$
$\mathcal{OBLOT}_{\sqrt{10}}^A$	$\mathbb{R}^2$	[109]	one-axis agreement	$\Theta(\Delta)$
$\mathcal{OBLOT}_7^F$	$\mathbb{Z}^2$	[24]	disoriented	$\mathcal{O}(n^2)$
$\mathcal{LUMI}_{19}^F$ (closed chain)	$\mathbb{Z}^2$	[2]	disoriented	$\mathcal{O}(n)$
$\mathcal{LUMI}_{11}^F$	$\mathbb{Z}^2$	[40]	disoriented	$\mathcal{O}(n)$
$\mathcal{LUMI}_4^A$ (closed chain)	$\mathbb{R}^2$	CC-HOPPER	disoriented	$\mathcal{O}(n)$

Table 6.1: Results about GATHERING compared to existing work.

6.2 Model Recap and Preliminaries

We study the $\mathcal{LUMI}_4^A$ model in combination with a closed chain of robots. The most important features of the model required for this chapter are summarized in Table 6.2, and more details can be found in Chapter 2.

Protocol	Time	Dimension	Viewing Range	Orientation	Chain
CC-HOPPER	ASYNC	2	4	disoriented	yes (closed)

Table 6.2: A summary of the most important model details for the CC-HOPPER protocol.

Problem Statement. In the GATHERING problem of a closed chain of robots, all robots have to move to the same position. More formally, we aim for a time t such that $L(t) = 0$.

6.3 Protocol for the $\mathcal{F}$SYNC Scheduler

Next, we present the CC-HOPPER protocol. Each robot has a viewing range of 4, i.e., it can always see the positions of the next 4 neighbors in each direction along the chain. Our approach consists of two protocols – one for asymmetric configurations and one for highly symmetric (isogonal) configurations.

6.3.1 Intuition

The general behavior of the CC-HOPPER protocol is quite similar to the ε-2-HOPPER protocol. However, the nature of the closed chain introduces additional challenges. In the following, we discuss the three main challenges that are handled by the CC-HOPPER protocol: (1) the generation of run sequences, (2) stopping of run sequences after at most n rounds and (3) handling of highly symmetric configurations.

Generation of run sequences. In contrast to an open chain, no outer robots exist that might start new run sequences regularly. Thus, one of the main questions to solve is *where* run sequences should be started. For this, we identify geometrically unique robots in their local neighborhood. These robots are assigned an init state (implemented with a light ℓ_{init}) allowing them to regularly generate new run sequences. Formally, we define for a robot r_i, $init(r_i,t) = true$ if r_i has an init state in round t and $init(N_i(t)) = \{r_j \in N_i(t) | \, init(r_j,t) = true\}$.

To maintain the connectivity of the chain, it is essential that not too many neighbors have a run state simultaneously. The generation of new run sequences must maintain this property. Therefore, the protocol ensures that at most two neighboring robots have an init state. Additionally, a robot r_i with an init state only generates new run sequences in case no other run sequence is present in its neighborhood.

Stopping of run sequences after at most n rounds. One core feature of the ε-2-HOPPER protocol is that each run sequence stops after at most n rounds either by making some progress (in terms of shorten or merge operations) or reaching the end of the chain. In a closed chain, however, no end of the chain exists. Nevertheless, it is crucial for the linear runtime of the CC-HOPPER protocol that each run sequence stops after at most n rounds since otherwise, the run sequence could cycle multiple times around the chain and hinder other robots from generating new run sequences that potentially lead to progress.

The CC-HOPPER protocol uses the following ideas to ensure this behavior. Robots with init states generate two run sequences at the same time: one run sequence heading in each direction of the chain. Additionally, the robot with the init state moves to the midpoint between its two direct neighbors before generating the run sequences. This ensures that both run sequences start with opposite run vectors. Furthermore, a hop is only executed if the angle between neighboring vectors is larger than $\frac{7}{8}\pi$. Suppose that the robot starting the two run sequences lies in the origin of a global coordinate system, and after moving to the midpoint, one of its neighbors lies on the positive x-axis while the other one lies on the negative x-axis. The angle of $\frac{7}{8}\pi$ ensures that the run sequence that starts along the positive x-axis can only move to the right in case of a hop or joint hop while the other run sequence can only move to the left. Hence, the two run sequences cannot meet each other again and at least one run sequence must stop via a merge, a joint merge, a shorten or a joint shorten. See Figure 6.1 for a visualization of this behavior.

To guarantee that also the second run sequence stops after at most n rounds, we need to ensure that the chain structure cannot change too much in n rounds. Therefore, we add a rule: Each run sequence that leads to a merge or a joint merge stops *all* run sequences in its neighborhood. Additionally, no new run sequence is generated in this neighborhood for a constant number of rounds. This way, it is ensured that not too many (joint) merges happen during n rounds such that the second run sequence is also guaranteed to stop.

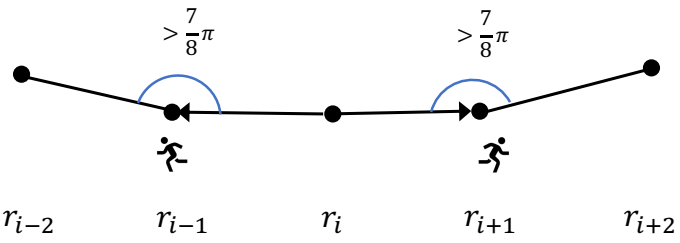

Figure 6.1: Two run sequences generated by r_i with opposite run vectors. Due to the threshold of $\frac{7}{8}\pi$, the run sequence at r_{i-1} can only execute a hop if r_{i-2} is positioned to the left of r_{i-1}. The same holds mirrored for r_{i+1}.

Handling of Highly Symmetric Configurations. It is impossible to identify locally unique robots in a class of highly symmetric configurations, denoted as *isogonal configurations*. An example of an isogonal configuration is when the positions of the robots form a regular polygon with a side length of 1. Then, the local coordinate systems of the robots could be aligned such that every robot observes exactly the same. To overcome the impossibility of finding locally unique robots, we introduce an additional protocol for these configurations that works without run sequences. Isogonal configurations have in common that all robots lie on the boundary of a common circle. We exploit this fact by letting the robots move towards the center of the surrounding circle in every round until they finally gather in its center. Additional care has to be taken in case both protocols interfere with each other. This can happen if some parts of the chain are isogonal while others are asymmetric. Since a robot can only decide how to move based on its local view, the robots behave according to different protocols in this case. We show how to handle such a case and ensure that the two protocols do not hinder each other later.

6.3.2 Asymmetric Protocol

The asymmetric protocol consists of two parts: The generation of new run sequences and the movement depending on such a run sequence. We start with explaining the movement depending on run sequences. Assume that the number of robots in the chain is at least 6 and consider an isolated run sequence κ in round t with $r(\kappa,t) = r_i$ and $r(\kappa,t+1) = r_{i+1}$. Then, r_i moves as follows:

1. If $\|p_{i-1}(t) - p_{i+1}(t)\|_2 \leq 1$, r_i: merge.
2. Else, if $\|p_i(t) - p_{i+2}(t)\|_2 \leq 1$, r_i: Pass the run sequence to r_{i+1}.
3. Else, if $\alpha_i(t) \leq \frac{7}{8}\pi$, r_i: shorten.
4. Else, r_i: hop.

Given a joint run pair at robots r_i and r_{i+1}, the robots r_i and r_{i+1} move according to the following description.

1. If $\|p_{i-1}(t) - p_{i+2}(t)\|_2 \leq 2$, both: joint merge.
2. Else, if $\alpha_i(t) \leq \frac{7}{8}\pi$ and $\alpha_{i+1}(t) \leq \frac{7}{8}\pi$, both: joint shorten.
3. Else, if $\alpha_i(t) \leq \frac{7}{8}\pi$, r_i: shorten.
4. Else, if $\alpha_{i+1}(t) \leq \frac{7}{8}\pi$, r_{i+1}: shorten.
5. Else, if $\measuredangle(w_i(t), -w_{i+2}(t)) \leq \frac{7}{8}\pi$. both: joint shorten.
6. Else, both: joint hop.

If the number of robots in the chain is at most 5 (robots can detect this since they can see 4 chain neighbors in each direction), the robots move towards the center of the smallest enclosing circle of their neighborhood while ensuring connectivity. More precisely, the robots execute the GTC protocol which ensures GATHERING after $\mathcal{O}(1)$ rounds [50] (and Chapter 4).

Additional Rule for Merges and Joint Merges. To ensure that all started run sequences stop, we add a rule that stops nearby run sequences after (joint) merges to ensure that the chain structure does not change too much within n rounds (see Section 6.3.1 for an intuition). Suppose the robot r_i (and r_{i+1}) executes a merge (or a joint merge). Then, all run sequences in the neighborhood of r_i and r_{i+1} are immediately stopped and all robots in $N_i(t)$ do not start any further run sequences within the next 4 rounds (the robots are *blocked* and the counting of blocked rounds is implemented with a light $\ell_{blocked}$). Special care has to be taken of init states. Suppose that a robot r_i executes a merge into the direction of r_{i+1} while having an init state. The init state is handled as follows: In case $init(r_{i+2}) = false$ and r_{i+2} does not execute a merge in the same round and $init(r_{i+3}) = true$, the init state of r_i is passed to r_{i+1}. Otherwise, the state is removed.

Where to start run sequences? New run sequences are created by robots with init states. To generate new init states, we aim to discover asymmetric structures in the chain. When the surrounding robots observe such a structure, the robot closest to the structure is assigned a so-called init state. Such a robot can thus remember that it was at a point of asymmetry to generate runs in the future. To keep the distance between run sequences (essential for maintaining the connectivity), our rules ensure that at most two neighboring robots have an init state. Sequences of length at least 3 of robots having an init state are avoided. Intuitively, there are three sources of asymmetry in the chain: Sizes of angles, orientations of angles, and lengths of vectors. To detect an asymmetry, we introduce *patterns* depending on the size of angles, the orientation of angles, and the vector lengths. The next class of patterns is only checked if a complete symmetry regarding the previous pattern is identified to avoid too many fulfilled patterns. More precisely, a robot only checks orientation patterns if all angles $\alpha_i(t)$ in its neighborhood are identical. Similarly, a robot only checks vector length patterns if all angles in its neighborhood have the same size and orientation. Whenever a pattern holds, the robot observing the pattern assigns itself an init state if there is no other robot already assigned an init state in its neighborhood. If two direct neighbors are assigned an init state, they fulfilled the same type of pattern and formed a joint init state together.

Angle Patterns. A robot r_i is assigned an init state if

$$\alpha_{i-1}(t) > \alpha_i(t) \leq \alpha_{i+1}(t)$$
$$\textbf{or } \alpha_{i-1}(t) \geq \alpha_i(t) < \alpha_{i+1}(t).$$

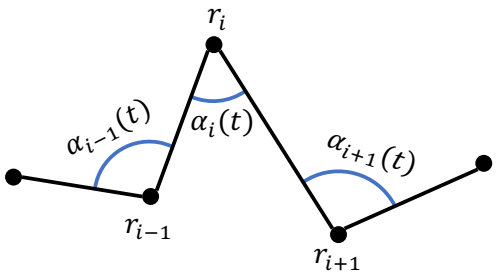

Figure 6.2: A configuration in which r_i fulfills the first *Angle Pattern*, i.e., $\alpha_i(t)$ is a local minimum.

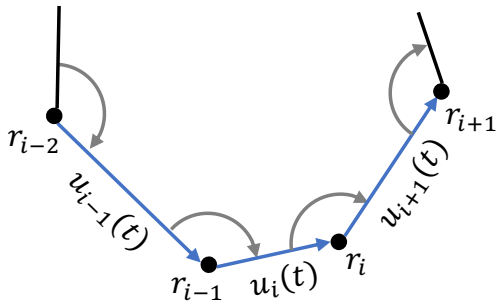

Figure 6.3: A configuration in which r_i (and also r_{i-1}) fulfills the first *Vector Length Pattern*, i.e., the length of $w_i(t)$ is a local minimum.

Orientation Patterns. A robot r_i gets an init state if one of the following patterns is fulfilled.

1. r_i is between three angles that have a different orientation than $\alpha_i(t)$:

$$\begin{aligned} &\operatorname{sgn}_i(\alpha_{i-1}(t)) = \operatorname{sgn}_i(\alpha_{i+1}(t)) = \operatorname{sgn}_i(\alpha_{i+2}(t)) \neq \operatorname{sgn}_i(\alpha_i(t)) \\ \textbf{or } &\operatorname{sgn}_i(\alpha_{i-2}(t)) = \operatorname{sgn}_i(\alpha_{i-1}(t)) = \operatorname{sgn}_i(\alpha_{i+1}(t)) \neq \operatorname{sgn}_i(\alpha_i(t)). \end{aligned}$$

2. r_i borders a sequence of at least two angles with the same orientation next to a sequence of at least three angles with the same orientation:

$$\begin{aligned} &\operatorname{sgn}_i(\alpha_{i-1}(t)) = \operatorname{sgn}_i(\alpha_i(t)) \neq \operatorname{sgn}_i(\alpha_{i+1}(t)) = \operatorname{sgn}_i(\alpha_{i+2}(t)) = \operatorname{sgn}_i(\alpha_{i+3}(t)) \\ \textbf{or } &\operatorname{sgn}_i(\alpha_{i+1}(t)) = \operatorname{sgn}_i(\alpha_i(t)) \neq \operatorname{sgn}_i(\alpha_{i-1}(t)) = \operatorname{sgn}_i(\alpha_{i-2}(t)) = \operatorname{sgn}_i(\alpha_{i-3}(t)). \end{aligned}$$

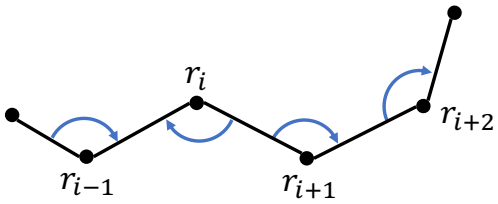

Figure 6.4: A configuration in which r_i fulfills the first *Orientation Pattern*.

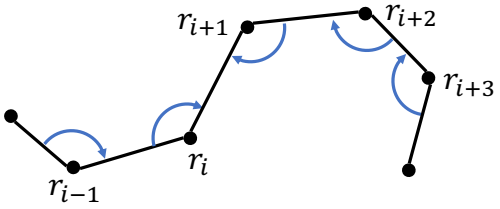

Figure 6.5: A configuration in which r_i fulfills the second *Orientation Pattern*.

Vector Length Patterns. If Angle and Orientation Patterns fail, we consider vector lengths. A robot r_i is assigned an init state if one of the following patterns is fulfilled. In the patterns, the term *locally minimal* occurs. $\|w_i\|_2$ is locally minimal means that all other vectors that can be seen by r_i are either larger or have the same length.

1. The robot is located at a locally minimal vector next to two succeeding larger vectors, i.e., $\|w_i\|_2$ is locally minimal **and** $\|w_{i-1}\|_2 > \|w_i\|_2 < \|w_{i+1}\|_2$ **and** $\|w_i\|_2 < \|w_{i+2}\|_2$ **or** $\|w_{i+1}\|_2$ is locally minimal **and** $\|w_i\|_2 > \|w_{i+1}\|_2 < \|w_{i+2}\|_2$ **and** $\|w_{i+1}\|_2 < \|w_{i+3}\|_2$.
2. The robot is at the boundary of a sequence of at least two locally minimal vectors, i.e., $\|w_{i-1}\|_2 = \|w_i\|_2 < \|w_{i+1}\|_2$ **or** $\|w_i\|_2 > \|w_{i+1}\|_2 = \|w_{i+2}\|_2$.

How to start run sequences? Robots with (joint) init states try every 9 rounds (counted with a light ℓ_c) to start new run sequences. A robot r_i with $\mathit{init}(r_i) = \mathit{true}$ only starts a new run sequence if $\mathit{run}(N_i(t)) = \emptyset$ to ensure sufficient distance between run sequences. The constant 9 is chosen to ensure that a robot (potentially) observes different run sequences in its neighborhoods each time it tries to start a new run sequence (since a robot can observe 9 robots including itself and each run sequences moves at least one robot forward per round). Additionally, r_i only starts new run sequences provided $\|p_{i-1}(t) - p_{i+1}(t)\|_2 > 1$. Otherwise, it executes a merge. Given $\|p_{i-1}(t) - p_{i+1}(t)\|_2 > 1$, r_i generates two new run sequences at its direct neighbors with opposite directions as follows: r_i executes a shorten and generates two new run sequences κ_1 and κ_2 with $r(\kappa_1, t+1) = r_{i+1}$ and $r(\kappa_2, t+1) = r_{i-1}$. Two robots r_i and r_{i+1} with a joint init state proceed similarly: given $\|p_{i-1}(t) - p_{i+2}(t)\|_2 \leq 2$, they directly execute a joint merge. Otherwise, r_i and r_{i+1} execute a joint shorten and induce two new run sequences at their direct neighbors with opposite directions. Observe that this procedure ensures that every vector assigned to a run sequence has a length of at least $\frac{1}{2}$.

6.3.3 Symmetric Protocol

Due to the patterns in Section 6.3.2, there is a set of configurations where no init state is generated. Intuitively, such configurations have no local criterion that identifies some robots as different from their neighbors, i.e., they are *symmetric*. We start by defining precisely the class of isogonal configurations in which no run sequence can be generated by our protocol. Afterward, we show how we can still gather such configurations. Intuitively, a configuration is isogonal if all angles have the same size and orientation and either all vectors have the same length or there are two alternating vector lengths. For some round t, the set of all vectors $w_i(t)$ describes a polygon denoted as the *configuration polygon* of round t. A configuration is then called an *isogonal configuration* in case its configuration polygon is isogonal. A polygon P is *isogonal* if and only if for each pair of vertices, there is a symmetry of P that maps the first onto the second [74]. Examples of such polygons can be seen in Figures 6.6 and 6.7. Grünbaum [74] classified the set of isogonal polygons. This set of polygons consists of the set of *regular star polygons* and polygons that can be obtained from them by a small translation of the vertices.

Definition 6.1 — [74]. The *regular star polygon* $\{n/d\}$ $(n,d \in \mathbb{N}, d \leq n)$ is constructed as follows: Consider a circle C and fix an arbitrary radius R of C. Place n points $A_1,\ldots,A_n$ such that A_j is placed on C and forms an angle of $\frac{2\cdot\pi\cdot d}{n\cdot j}$ with R and connect A_j to A_{j+1} mod n by a segment. A configuration is called a *regular star configuration* in case the configuration polygon is a regular star polygon.

Next, we describe, how isogonal polygons with two alternating edge lengths can be constructed out of regular star polygons. The construction is based on [74]. For n odd, every isogonal polygon is a regular star polygon. For n even, isogonal polygons that are not regular star polygons can be constructed as follows: Take any regular star polygon $\{n/d\}$ based on the circle C of radius R. Now, choose a parameter $0 < t < \frac{n}{2}$ and locate the vertex A_j such that its angle to R is $\frac{2\pi}{n} \cdot (j \cdot d + (-1)^j \cdot t)$. Choosing $t = \frac{n}{2}$ yields the polygon $\{n/d\}$ again. Larger values for t obtain the same polygons as in the interval $[0, \frac{n}{2}]$.

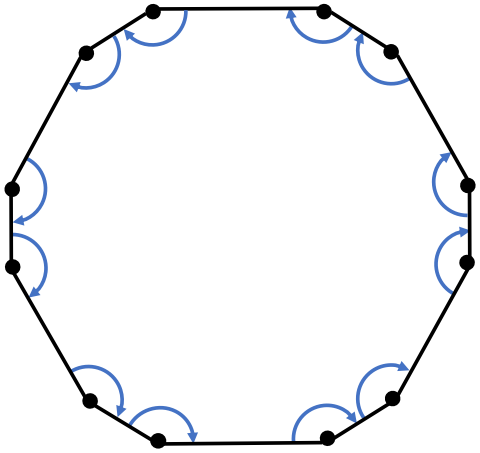

Figure 6.6: An isogonal configuration that has two alternating vector lengths and all angles are equal with $n = 12$.

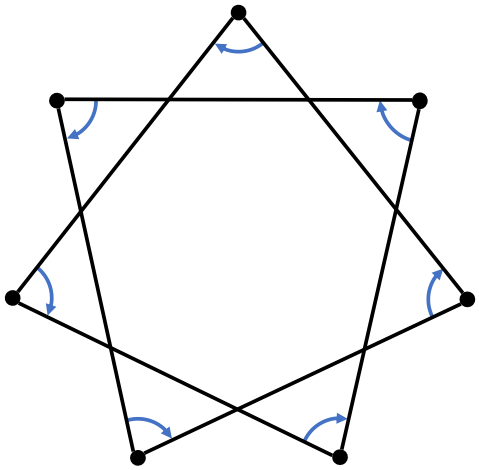

Figure 6.7: An isogonal configuration of which the configuration polygon is a regular star polygon $n = 7$.

Movement. A robot r_i moves according to the symmetric protocol if all $\alpha_i(t)$ in its neighborhood have the same size and orientation, either all vectors $w_i(t)$ have the same length or have two alternating lengths, $init(N_i(t)) = \emptyset$ and $run(N_i(t)) = \emptyset$. Then, r_i performs one of the two following symmetrical operations. In case all vectors $w_i(t)$ have the same length, it performs a *bisector-operation*. The purpose of the bisector-operation is to move all robots towards the center of the

surrounding circle. Otherwise (in the case of two alternating vector lengths), the robot executes a *star-operation*. The goal of the star-operation is to transform an isogonal configuration with two alternating vector lengths into a regular star configuration. Afterward, bisector-operations are executed.

Bisector-Operation & Star-Operation. In the *bisector-operation*, a robot r_i computes the angle bisector of vectors pointing to its direct neighbors (bisecting the angle of size less than π) and jumps to the point p on the bisector such that $\|p_{i-1}(t) - p\|_2 = \|p_{i+1}(t) - p\|_2 = 1$. If $\|p_i(t) - p\|_2 > \frac{1}{5}$, the robot moves only a distance of $\frac{1}{5}$ towards p. Additionally, the *star-operation* works as follows: Let C be the circle induced by r_i's neighborhood and R its radius. If the diameter of C is at most 2, r_i jumps to the midpoint of C. Otherwise, the robot r_i observes the two circular arcs $L_\alpha = \alpha \cdot R$ and $L_\beta = \beta \cdot R$ connecting itself to its direct neighbors. The angles α and β are the corresponding central angles measured from the radius R_i connecting r_i to the midpoint of C. Without loss of generality, we assume that $L_\alpha < L_\beta$. Then, r_i jumps to the point on L_β such that L_α is enlarged by $R \cdot ((\beta - \alpha)/4)$ and L_β is shortened by the same value.

6.3.4 Combination

The asymmetric and symmetric protocols are executed in parallel. More precisely, robots whose neighborhoods fulfill the property of an isogonal configuration move according to the symmetric protocol, while others follow the asymmetric protocol. To ensure that the two protocols do not hinder each other, we need one additional rule: the *exceptional generation of init states*. Intuitively, if some robots follow the asymmetric protocol while others execute the symmetric protocol, there are borders at which a robot r_i moves according to the symmetric protocol while its neighbor does not move at all (the neighbor cannot have a run state; otherwise r_i would not move according to the symmetric protocol). At these borders, the length of the chain may increase. We use an additional visible state to prevent this from happening too often (implemented with the light ℓ_{symm}). Robots that move according to the symmetric protocol store this via activating ℓ_{symm}. If any robot detects in the next round that ℓ_{symm} is activated, but its local neighborhood does not fulfill the criterion of being an isogonal configuration, it concludes that the chain is not entirely symmetric. To ensure this does not occur again, the robot closest to the asymmetry is assigned an init state and thus adds a source of asymmetry to the chain. Note that this exceptional generation is not needed to find robots that start run sequences. Instead, the exceptional generation ensures that the symmetric operations are not executed too often in case the global configuration is asymmetric.

6.3.5 High-level Analysis

This section presents the main course of the analysis devoted to proving the following main theorem about the CC-HOPPER protocol under the $\mathcal{F}$SYNC scheduler.

Theorem 6.1 For any initially connected closed chain of disoriented robots in the Euclidean plane with a viewing range of 4 and a connectivity range of 1, GATHERING can be solved with the CC-HOPPER protocol in $\mathcal{O}(n)$ epochs assuming the $\mathcal{F}$SYNC scheduler and a light with a constant number of colors. The number of rounds is asymptotically optimal.

One of the crucial properties of the correctness of our protocol is that it maintains the connectivity of the chain. For the proof, we show that the operations of isolated run sequences, joint run pairs, and the operations of the symmetrical protocol do not break the connectivity as well as no other pattern of run states exists (for instance, a sequence of three neighboring robots having a run state).

Lemma 6.1 A configuration that is connected in round t stays connected in round $t+1$.

The asymmetric protocol depends on the generation of run sequences. We prove that at least one pattern is fulfilled in every asymmetric configuration in which no robot has an init state.

Lemma 6.2 A configuration without any init state in round t is either isogonal or at least one init state exists in round $t+1$.

The following lemma is crucial for the asymmetric protocol: Every run sequence that is started at robot r_i will never visit r_i again in the future. The run sequence either stops by a merge, a joint merge, a shorten, a joint shorten or it is stopped by a merge or a joint merge of a different run sequence. Consequently, a run sequence cannot execute n succeeding hops or joint hops.

Lemma 6.3 A run sequence does not visit the same robot twice.

Next, we count the number of run sequences needed to gather all robots on a single point. Lemma 6.3 states that each run sequence stops either by a shorten, joint shorten, merge or joint merge or it is stopped via a merge or joint merge of a different run sequence. There can be at most $n-1$ merges and joint merges. To count the number of shortens and joint shortens, we consider two cases: Either the two vectors involved in a shorten have both a length of at least $\frac{1}{2}$ or one vector is smaller, and the other one is larger than $\frac{1}{2}$ (the case that both vectors are smaller than $\frac{1}{2}$ would lead to a merge). Due to the threshold of $\frac{7}{8}\pi$, we can prove that the chain length reduces by at least a constant in case both vectors have a length of at least $\frac{1}{2}$. If one vector is smaller and the other larger than $\frac{1}{2}$, the chain length does not necessarily decrease by a constant. Instead, the smaller vector increases and has a length of at least $\frac{1}{2}$ afterward. Hence, either the chain length or the number of small vectors decreases. New small vectors can only be created upon the execution of merge, joint merges, star-operations or bisector-operations. For each of the mentioned operations, we can prove that it is only executed a linear number of times. We conclude that a linear number of run sequences is needed to gather all robots on a single point.

Lemma 6.4 At most $143n$ run sequences are required to gather all robots.

To conclude a final runtime for the asymmetric protocol, we need to prove that sufficiently many run sequences are generated. We apply a witness argument. Consider an init state. Each 9 rounds, this state either creates a new run sequence or observes other run sequences in its neighborhood and waits. This way, we can count each 9 rounds a new run sequence: Either the robot with the init state starts a new run sequence or it waits because of a different run sequence. Since a robot can observe 9 neighbors, we count a new run sequence after 9 rounds. Roughly said, we can prove that in k rounds $\approx \frac{k}{9}$ run sequences exist. This holds until the init state is removed due to a merge or joint merge. Afterward, we can continue counting at the next init state in the direction of the run sequence causing the merge or joint merge.

Lemma 6.5 A configuration that does not become isogonal gathers after at most $5129n$ rounds.

It remains to prove a linear runtime for the symmetric protocol. The symmetric protocol consists of two parts: First, an isogonal configuration is transformed into a regular star configuration, and afterward, the robots move towards the center of the surrounding circle. The transformation to a regular star configuration requires a single round in which all robots execute a star-operation.

Lemma 6.6 If the configuration is isogonal but not a regular star configuration in round t, the configuration is a regular star configuration in round $t+1$.

To prove a linear runtime for regular star configurations, we analyze the runtime for the regular polygon $\{n/1\}$. In all other regular star configurations, the inner angles are smaller; thus, the robots can move larger distances towards the center of the surrounding circle. We use the radius of the circumcircle as a progress measure. Although the radius decreases very slowly initially, it decreases by a constant in every round after a linear number of rounds. The linear runtime follows.

Lemma 6.7 Regular star configurations gather in at most $30n$ rounds.

6.3.6 Detailed Analysis

Next, we present the detailed analysis of the CC-HOPPER protocol. Section 6.3.6.1 contains a detailed proof that all operations maintain the connectivity of the chain. Afterward, Section 6.3.6.2 deals with the analysis of the asymmetric protocol, and Section 6.3.6.3 analyses the protocol for isogonal configurations.

6.3.6.1 Connectivity

We first prove that the individual movement operations do not violate the connectivity of the chain.

Lemma 6.8 The movement operations of isolated run sequences and joint run pairs keep the chain connected.

Proof. A robot r_i executes only a merge if $\|(p_{i-1}(t) - p_{i+1}(t)\|_2 \leq 1$. Since r_i moves either to $p_{i-1}(t)$ or to $p_{i+1}(t)$ (depending on the direction of the run sequence) and neither r_{i-1} nor r_{i+1} move in the same round, the chain remains connected.

Consider a robot r_i that executes a shorten. It moves to the midpoint between its neighbors, more formally $p_i(t+1) = \frac{1}{2}p_{i-1}(t) + \frac{1}{2}p_{i+1}(t)$. Since the configuration is connected in round t, it holds $\|p_{i-1}(t) - p_{i+1}(t)\|_2 \leq 2$ and thus it follows $\|p_{i-1}(t+1) - p_i(t+1)\|_2 \leq 1$ and $\|p_i(t+1) - p_{i+1}(t+1)\|_2 \leq 1$ as $p_{i-1}(t+1) = p_{i-1}(t)$ and $p_{i+1}(t+1) = p_{i+1}(t)$.

Now, suppose r_i executes a hop in the direction of r_{i+1}. The hop exchanges the vectors $w_i(t)$ and $w_{i+1}(t)$. Since both vectors have a length of at most 1, the connectivity is ensured in round $t+1$. The arguments for a joint merge, joint shorten and joint hop are analogous. ■

Lemma 6.9 Star-operations and bisector-operations keep the chain connected.

Proof. For bisector-operations, this follows directly from the definition: two neighboring robots that execute a bisector-operation jump towards the center of the same circle, and thus their distance decreases. If a robot executes a bisector-operation while its neighbor does not move, the bisector-operation ensures by definition that their distance is at most 1 in the next round. Other cases cannot occur: Since bisector-operations are only executed in case no run sequence is visible in the neighborhood of a robot, it cannot happen that a neighbor executes a different operation. Thus, bisector-operations maintain the connectivity of the chain.

The last arguments also apply to the star-operation. A robot only executes a star-operation if its neighbors also execute a star-operation or do not move at all. Suppose two neighboring robots execute a star-operation. In round t they are connected via a circular arc $L_\alpha = \alpha \cdot r$ or $L_\beta = \beta \cdot r$, where r denotes the radius of the circumcircle of the neighborhood. Assume that $\alpha < \beta$. If the robots are connected with L_β, the robots move closer to each other and maintain connectivity. Otherwise, they move away from each other, but the new circular arc connecting the two robots is $L_\alpha + r \cdot \left(\frac{\beta-\alpha}{2}\right)$ which is less than L_β. Hence, the two neighboring have a distance of at most 1 in the next round. ■

Next, we prove that either isolated run sequences or joint run pairs exist. For that, we show that it is impossible that other patterns of run states exist. The next lemma focuses on the generation of new run sequences.

Lemma 6.10 In every round t, there exists no sequence of neighboring robots of length at least 3, all having an init state.

Proof. Robots that already have neighbors with an init state do not generate new ones. Thus, we only have to look at sequences of robots in which no robot has an init state and show that a pattern is fulfilled for at most two neighbors. First, consider the angle patterns. Suppose for a robot r_i the first angle pattern is fulfilled (the argumentation for the second angle pattern is analogous because it describes the mirrored version.) Thus, it holds, $\alpha_{i-1}(t) > \alpha_i(t) \leq \alpha_{i+1}(t)$. For r_{i-1}, no angle pattern can hold because $\alpha_{i-1}(t)$ is no local minimum. Additionally, r_{i-1} does not check any further patterns because $\alpha_{i-1}(t) \neq \alpha_i(t)$ and hence no full symmetry is given. The only case in which an angle pattern for r_{i+1} can be fulfilled is the case that $\alpha_i(t) = \alpha_{i+1}(t)$. If $\alpha_{i+1}(t) \geq \alpha_{i+2}(t)$, no pattern for r_{i+1} holds, since $\alpha_{i+1}(t)$ is either no local minimum or $\alpha_i(t) = \alpha_{i+1}(t) = \alpha_{i+2}(t)$. Thus, the only case that a pattern for both r_i and r_{i+1} holds is the case that $\alpha_{i-1}(t) > \alpha_i(t) = \alpha_{i+1}(t) < \alpha_{i+2}(t)$. Since $\alpha_{i+2}(t)$ is also no local minimum, no pattern for r_{i+2} holds. As a consequence, an angle pattern for both r_i and r_{i+1} may be fulfilled, but then, neither r_{i-1} nor r_{i+2} fulfills a pattern.

We continue with the orientation patterns. There are two classes of orientation patterns; we start with the first class. Assume that for r_i, an orientation pattern is fulfilled. Thus, it holds $\text{sgn}(\alpha_{i-1}(t)) = \text{sgn}(\alpha_{i+1}(t)) = \text{sgn}(\alpha_{i+2}(t)) \neq \text{sgn}(\alpha_i(t))$ (the other pattern in this class is a mirrored version and the same argumentation can be applied). No orientation pattern of the first class is fulfilled for r_{i+1} because the neighboring angles have a different orientation. Additionally, no pattern of the second class can be fulfilled, because $\text{sgn}(\alpha_i(t)) \neq \text{sgn}(\alpha_{i-1}(t))$. For r_{i-1} no pattern of the second class can be fulfilled because $\text{sgn}(\alpha_i(t)) \neq \text{sgn}(\alpha_{i+1}(t))$. A pattern of the first class can only be fulfilled if $\text{sgn}(\alpha_{i-3}(t)) = \text{sgn}(\alpha_{i-2}(t)) = \text{sgn}(\alpha_i(t))$. In this case, no pattern for r_{i-2} can be fulfilled because its neighboring angles have a different orientation, and it is not located at the boundary of two sequences of angular orientations of the length of at least two. Hence, a pattern for r_i and r_{i-1} may be fulfilled but then neither a pattern for r_{i-2} nor for r_{i+1} is fulfilled.

Lastly, we consider the vector length patterns. Suppose that a vector length pattern is fulfilled for r_i. We give the arguments for the first class; the second class is analogous. Arguments for the third class are given afterward. Assume that the first vector pattern is fulfilled for r_i and thus $\|w_i(t)\|_2$ is locally minimal and $\|w_{i-1}(t)\|_2 > \|w_i(t)\|_2 < \|w_{i+1}(t)\|_2$ and $\|w_i(t)\|_2 < \|w_{i+2}(t)\|_2$. In this case, no pattern can be fulfilled for r_{i+1} because $\|w_{i+1}(t)\|_2$ and $\|w_{i+2}(t)\|_2$ are not locally minimal ($\|w_i(t)\|_2$ is smaller). For r_{i-1} at most the second pattern can be fulfilled (because $\|w_{i-1}(t)\|_2$ is not locally minimal. The second pattern can only be fulfilled if $\|w_{i-2}(t)\|_2 > \|w_i(t)\|_2$. In this case, no pattern for r_{i-2} can hold because neither $\|w_{i-2}(t)\|_2$ nor $\|w_{i-1}(t)\|_2$ are locally minimal. Thus, a pattern for r_{i-1} and r_i can be fulfilled, but no patterns for r_{i-2} and r_{i+1}. The arguments for the second class are analogous.

Now, suppose that the third pattern is fulfilled for a robot r_i. Thus, it holds $\|w_{i-1}(t)\|_2 = \|w_i(t)\|_2 < \|w_{i+1}(t)\|_2$ (the arguments for the other pattern are analogous because it describes the mirrored version). In this case, no vector pattern holds for r_{i-1} since both neighboring vectors have the same length. For r_{i+1}, the third pattern can be fulfilled. It must hold $\|w_{i+2}(t)\|_2 = \|w_{i+3}(t)\|_2 < \|w_{i+1}(t)\|_2$. However, the third pattern cannot hold for r_{i+2} because both neighboring vectors have the same length. In addition, neither the first nor the second pattern can hold for r_{i-1} and r_{i+1} (due to the definition of the third pattern) and thus both r_i and r_{i+1} can generate an init state but neither r_{i-1} nor r_{i+2}. ■

In the following, we define undesired patterns of run states formally.

Definition 6.2 Three run states are called a *prohibited run sequence* if the three run states are located at three directly neighboring robots.

Definition 6.3 Consider two run sequences κ_1 and κ_2. Two run sequences κ_1, κ_2 are called an *opposite conflicting run pair* in case $r(\kappa_1, t)$ and $r(\kappa_2, t)$ are direct neighbors and $r(\kappa_1, t+1) \neq r(\kappa_2, t)$ and $r(\kappa_2, t+1) \neq r(\kappa_1, t)$. Two run sequences κ_1, κ_2 are called a *uni-directional conflicting run pair* in case $r(\kappa_1, t+1) = r(\kappa_2, t)$ and $r(\kappa_2, t+1) \neq r(\kappa_1, t)$ or vice versa.

Definition 6.4 A configuration is called to be *run sequence-valid* in round t if neither a prohibited run sequence nor a conflicting run pair exists.

We aim to show that a configuration always remains run sequence-valid. To do so, we state the following auxiliary lemma.

Lemma 6.11 Consider a run sequence-valid configuration in round t with a joint run pair of run sequences κ_1 and κ_2 with $r(\kappa_1,t) = r_i$ and $r(\kappa_2,t) = r_{i+1}$. Suppose the robots execute a joint hop. Then, $run(r_i,t+1) = run(r_{i+1},t+1) = false$.

Proof. Since the configuration is run sequence-valid, $run(r_{i-1},t) = run(r_{i+2},t) = false$. As the robots execute a joint hop it holds $r(\kappa_1,t+1) = r_{i+2}$ and $r(\kappa_2,t+1) = r_{i-1}$. Thus, it can only happen that run sequences different from κ_1 and κ_2 are located at r_i or r_{i+1} in round $t+1$. Next, we argue that this is impossible. We prove that no run sequence κ_3 that is located at a robot with an index larger than $i+2$ can be located at r_i or r_{i+1}, the arguments for run sequences located at robots with smaller indices are analogous. Assume that there exists an isolated run sequence κ_3 with $r(\kappa_3,t) = r_{i+3}$. Depending on its direction it either holds $r(\kappa_3,t+1) = r_{i+4}$ or $r(\kappa_3,t+1) = r_{i+2}$. It follows that κ_3 cannot be located at r_{i+1} or r_i. Similar arguments hold for isolated run sequences located at robots with larger indices. Now, assume that there is a joint run pair κ_3 and κ_4 with $r(\kappa_3,t) = r_{i+3}$ and $r(\kappa_4,t) = r_{i+4}$. It holds $r(\kappa_3,t+1) = r_{i+5}$ and $r(\kappa_4,t+1) = r_{i+2}$. Again, the same arguments hold for joint run pairs located at robots with higher indices. It follows that $run(r_i,t+1) = run(r_{i+1},t+1) = false$. ■

Lemma 6.12 A configuration that is run sequence-valid in round t is also run sequence-valid in round $t+1$.

Proof. Lemma 6.10 states that starting of new run sequences always ensures that no prohibited run sequence exists. Beyond that, a merge or a joint merge stops all run sequences in its neighborhood such that run sequences do not come too close such that these operations also ensure that no prohibited run sequence exists. Shortens and joint shortens stop the involved run sequences and do not change the number of robots in the chain. Hence, no prohibited run sequences can be generated. Hops are only executed by isolated run sequences and continue in their direction and thus can also not create prohibited run sequences. The only operation we have to consider in more detail is the joint hop because the involved run sequences skip the next robot in their direction and move to the next but one robot. Let κ_1 and κ_2 denote a joint run pair with $r(\kappa_1,t) = r_i$ and $r(\kappa_2,t) = r_{i+1}$. By the definition of a joint run pair it holds $run(r_{i-1},t) = run(r_{i+2},t) = false$. Moreover, Lemma 6.11 states that $run(r_i,t+1) = run(r_{i+1},t+1) = false$. In the following, we prove that κ_1 is not part of a prohibited run sequence in round $t+1$; the arguments for κ_2 are analogous. By definition, it holds $r(\kappa_1,t+1) = r_{i+2}$. Lemma 6.11 gives us also that a prohibited run sequence cannot be generated by a joint run pair located at r_{i+3} and r_{i+4} or r_{i+4} and r_{i+5} since in both cases no run sequences is located at r_{i+4} in round $t+1$. Joint run pairs with larger indices are too far away to generate a prohibited run sequence.

This cannot happen by isolated run sequences because no neighboring robots have run states, and they move to the next. Hence, no prohibited run sequence can exist in round $t+1$.

Next, we argue that no conflicting run pair exists in round $t+1$. The start of new run sequences never creates new conflicting run pairs. Thus, if an opposite or a conflicting run pair exists in round $t+1$, both involved run sequences have already existed in round t. Consider a uni-directional run pair at round $t+1$. In round t, both run sequences must have had a distance of at least 2 (one robot without a run state in between). The distance between the two robots can only decrease based on a merge or a joint hop. A merge stops all run sequences in the neighborhood and cannot create such a run pair. A joint hop can also not create a uni-directional run pair since the configuration has

been run sequence-valid in round t and both neighboring robots have not had a run state. Thus, no uni-directional run pair can exist in round $t+1$.

Assume now that an opposite run pair exists at round $t+1$. This can only be the case if the two run sequences have been heading towards each other in round t. However, joint hops, joint shortens and joint merges ensure that no opposite run pair exists in round $t+1$. Hence, the configuration remains run sequence-valid. ■

Finally, we conclude that the connectivity is always maintained.

Lemma 6.1 A configuration that is connected in round t stays connected in round $t+1$.

Proof. By Lemma 6.12 it holds that only isolated run sequences or joint run pairs exist (or no run sequence at all). Lemma 6.8 states that isolated run sequence and joint run pairs keep the connectivity of the chain and by Lemma 6.9 this also holds for all operations of the symmetric protocol. The lemma follows. ■

6.3.6.2 Asymmetric Case

For the asymmetric case, we start by proving that in every asymmetric configuration, at least one init state exists.

Lemma 6.2 A configuration without any init state in round t is either isogonal or at least one init state exists in round $t+1$.

Proof. Assume that the configuration is not isogonal. Now, suppose that not all angles $\alpha_i(t)$ are identical and consider the globally minimal angle $\alpha_{min}(t)$ at the robot r_{min} (or any of them if the angle is not unique). The robot r_{min} generates an init state if at least one of the neighboring angles is larger. Since $\alpha_{min}(t)$ is minimal, the only situation in which r_{min} does not generate an init state is that $\alpha_{min-1}(t) = \alpha_{min}(t) = \alpha_{min+1}(t)$. In this case, follow the chain in any direction until a robot r'_{min} is reached such that the next robot has a larger angle. This robot exists since we have assumed that not all angles are identical. For this robot, an angle pattern is fulfilled. Consequently, given a configuration in which not all angles are identical, at least one init state is generated.

Next, we consider that all angles are identical, but not all have the same orientation. Observe first that the chain must contain two more angles of one orientation than the other because the chain is closed. This implies that the orientations cannot be alternating along the entire chain, and it also cannot happen that alternating sequences of two angular orientations exist. More formally, the chain cannot consist only of the following two sequences:

1. $\text{sgn}_i(\alpha_i) \neq \text{sgn}_i(\alpha_{i+1}) \neq \text{sgn}_i(\alpha_{i+2}), \ldots$
2. $\text{sgn}_i(\alpha_i) = \text{sgn}_i(\alpha_{i+1}) \neq \text{sgn}_i(\alpha_{i+2}) = \text{sgn}_i(\alpha_{i+3}) \neq \text{sgn}_i(\alpha_{i+4}), \ldots$

As a consequence, at least one of the orientation patterns must be fulfilled: Either there exists a sequence of at least three angles with the same orientation, and a pattern is fulfilled at the boundary of the sequence, or there exists a robot r_i that lies between three angles with a different orientation than $\alpha_i(t)$. Hence, given that all angles have the same size but not the same orientation, there must be at least one fulfilled orientation pattern.

Lastly, we take a look at the vector length patterns. Now, we assume that all angles in the chain have the same size and orientation. Since we assume that the configuration is not isogonal, not all vectors can have the same length, and it also cannot be the case that there exist only two different vector lengths that are alternating along the chain. Consider the vector of global minimal length w_{min} (or any of them if the length is not unique). Two cases can occur: Either a sequence of at least two neighboring vectors of length $\|w_{min}\|_2$ exists. At the end of such a sequence, the third vector pattern is fulfilled. In case no such sequence exists, all vectors having the length of $\|w_{min}\|_2$ have larger direct neighbors. Since the configuration is not isogonal, there must be a vector of length $\|w_{min}\|_2$ such that the direct neighboring vectors are larger and at least one of the next but one vector is also larger (otherwise, the configuration is isogonal with two alternating vector

lengths). In such a robot, the first or second pattern is fulfilled. All in all, we have proven that for configurations that are not isogonal, at least one pattern is fulfilled. ■

Next, we count the number of occurrences of shortens, joint shortens, merges and joint merges until all robots are gathered. Since each merge and joint merge reduces the number of robots in the chain, the following lemma trivially holds.

Lemma 6.14 There are at most $n-1$ merges and joint merges.

There are two types of occurrences of shortens. Either both involved vectors have a length of at least $\frac{1}{2}$ or one vector is larger than $\frac{1}{2}$ while the other one is shorter. We prove that the chain length decreases by a constant in the first case. The second case reduces the number of vectors of length less than $\frac{1}{2}$ in the chain. The following two lemmas can be derived from lemmas we have seen in the context of the CHAIN-FORMATION problem in Chapter 5. More precisely, the lemmas can be derived from Lemma 5.2 by choosing $\varepsilon = \frac{2}{\sqrt{2+\sqrt{2+\sqrt{2}}}} - 1$, such that $2 \cdot \sin^{-1}\left(\frac{1}{1+\varepsilon}\right) = \frac{7}{8}\pi$.

Lemma 6.15 Assume an isolated run sequence κ with $r(\kappa,t) = r_i$ and $r(\kappa,t+1)$ executes a shorten and both $\|w_i(t)\|_2 \geq \frac{1}{2}$ and $\|w_{i+1}(t)\|_2 \geq \frac{1}{2}$. Then $L(t+1) \leq L(t) - 0.019$.

Joint shortens are different in the sense that every joint shorten reduces the length of the chain by at least a constant. This holds because every run vector has a length of at least $\frac{1}{2}$ and thus, two involved vectors in the joint shorten have a length of at least $\frac{1}{2}$.

Lemma 6.16 Assume that a joint run pair executes a joint shorten in round t. Then, $L(t+1) \leq L(t) - 0.019$.

Next, we count the total number of shortens in which both involved vectors have a length of at least $\frac{1}{2}$ and joint shortens. We have to take care that different operations might increase the chain length here. However, this can only happen in a single case: A robot executes a bisector-operation while its neighbor does not move. The exceptional generation of init states ensures that this happens at most n times.

Lemma 6.17 There are at most $\frac{n\cdot\left(1+\frac{1}{5}\right)}{0.019}$ executions of shortens and joint shortens in which both involved vectors have a length of at least $\frac{1}{2}$.

Proof. There is only one case in which $L(t)$ can increase: If a robot executes a bisector-operation while its direct neighbor does not. Since the maximal distance moved in a bisector-operation is $\frac{1}{5}$, $L(t)$ can increase by at most $\frac{1}{5}$ in this case. This, however, can happen at most n times. To see this, observe that if a robot r_i executes a bisector-operation in round t and one of its neighbors does not, $init(r_i) = true$ in round $t+1$. The robot r_i will not execute any further bisector-operation until it executes a merge since an init state in the neighborhood of a robot prevents the robot from executing a bisector-operation and the init state is only removed after a merge, Thus, such a case can happen for at most n times and thus, $L(t)$ is upper bounded by $n \cdot \left(1+\frac{1}{5}\right)$. Lemmata 6.15 and 6.16 state that each shorten in which both involved vectors have a length of at least $\frac{1}{2}$ and each joint shorten decrease the length of the chain by at least 0.019. Consequently, the total number of such operations can be upper bounded by $\frac{n\cdot\left(1+\frac{1}{5}\right)}{0.019}$. ■

To count the total number of shortens required to gather all robots, we count the number of shortens in which one vector has a length of at most $\frac{1}{2}$ as the last step.

Lemma 6.18 There are at most $4n$ executions of shortens such that one of the participating vectors has a length less than $\frac{1}{2}$.

Proof. There are at most n vectors of length at most $\frac{1}{2}$ in the beginning. Every shorten in which one vector of size at most $\frac{1}{2}$ and the other vector of length at least $\frac{1}{2}$ is involved increases the length of the smaller vector to at least $\frac{1}{2}$. The only way to create new vectors of length less than $\frac{1}{2}$ is via a merge, a joint merge, a bisector-operation or a star-operation. Merges and joint merges are executed at most n times and thus at most n vectors of length less than $\frac{1}{2}$ can be generated.

In the following, we consider the bisector-operation and the star-operation. A robot may execute a bisector-operation or a star-operation while its neighbor does not. In this case, the robot generates a new init state. Thus, this happens at most n times since init states prevent the robots in the neighborhood from executing bisector-operations or star-operations and an init state is only removed after a merge. Therefore, at most n vectors of length less than $\frac{1}{2}$ can be created by bisector-operations or star-operations if a neighboring robot does not execute the same operation.

It remains to consider that a robot and its direct neighbors execute a bisector-operation and a star-operation. Consider now the bisector-operation. This operation only takes place at a robot r_i in case $\|w_{i-1}(t)\|_2 = \|w_i(t)\|_2 = \|w_{i+1}(t)\|_2 = \|w_{i+2}(t)\|_2$. Assume now that $\|w_i(t)\|_2 > \frac{1}{2}$. Thus, in case $\|w_i(t+1)\|_2 \leq \frac{1}{2}$ it also holds $\|w_{i+1}(t+1)\|_2 \leq \frac{1}{2}$. If the configuration is completely isogonal, then no shorten will be executed at r_i anymore, or some parts of the chain still generate run sequences. In the latter case, r_i (and maybe also r_{i-1} and r_{i+1}) can execute a merge (since $\|p_{i-1}(t+1) - p_{i+1}(t+1)\|_2 \leq 1$ as $\|w_i(t+1)\|_2 \leq \frac{1}{2}$ and $\|w_{i+1}(t+1)\|_2 \leq \frac{1}{2}$) such that the next run sequence that comes close either executes a merge at r_{i-1}, r_i or r_{i+1}. Thus, also this case can happen at most n times such that at most $2n$ vectors of length at most $\frac{1}{2}$ can be generated of which at most n can be part of a future shorten.

Similar arguments apply for the star-operation: If both $\|w_i(t+1)\|_2 < \frac{1}{2}$ and $\|w_{i+1}(t+1)\|_2 < \frac{1}{2}$ after the star-operation at least one of them had a length of less than $\frac{1}{2}$ in round t. Hence, at most one vector of length less than $\frac{1}{2}$ can be generated by a star-operation. However, the same arguments as for the bisector-operation hold now: r_i (and maybe also r_{i-1} and r_{i+1}) can execute a merge (since $\|p_{i-1}(t+1) - p_{i+1}(t+1)\|_2 \leq 1$ as $\|w_i(t+1)\|_2 \leq \frac{1}{2}$ and $\|w_{i+1}(t+1)\|_2 \leq \frac{1}{2}$) such that the next run sequence that comes close either executes a merge at r_{i-1}, r_i or r_{i+1}. Thus, before r_i can generate a further vector of length at most $\frac{1}{2}$ via a further star-operation, either r_{i-1}, r_i or r_{i+1} execute a merge such that this can happen also at most n times.

In total, we obtain at most $4n$ shortens in which one participating vector has a length of at most $\frac{1}{2}$: n initial vectors that can have a length of at most $\frac{1}{2}$, n vectors that can be generated via merges, n vectors that can be generated via star-operations and n vectors that can be generated via bisector-operations. ■

Next, we prove that every run sequence is stopped after at most n rounds. More precisely, no run sequence visits the same robot twice. For the proof, we state some auxiliary lemmata that analyze how the position of a robot changes in a global coordinate system based on the movement operations of the protocol. We use the notation $w_i(t) = (x_i(t), y_i(t))$ to describe the components of a vector $w_i(t)$.

Lemma 6.19 Assume that a robot r_i executes a merge or a joint merge in round t. Then, $y_i(t+1) \geq y_i(t) - 1$.

Proof. Consider a run sequence κ with $r(\kappa, t) = r_i$ and $r(\kappa, t+1) = r_{i+1}$. In the worst case, it holds $\alpha_i(t) = 0$ and $y_{i+1}(t) = y_i(t) - 1$ such that r_i moves to the position of r_{i+1} and executes a merge there. In a joint merge, the distance is even less since the robots merge in the midpoint between their neighbors. ■

Lemma 6.20 Assume that a robot r_i executes a shorten or a joint shorten in round t. Then, $y_i(t+1) \geq y_i(t) - \frac{\sqrt{3}}{2}$.

Proof. Observe first that $\alpha_i(t) > \frac{\pi}{3}$, otherwise a merge can be executed. Thus, to maximize the distance covered in the vertical direction, consider the three involved robots to be the vertices of an equilateral triangle with a side length of 1. The height of this triangle is $\frac{\sqrt{3}}{2}$. Thus, $y_i(t+1) \geq y_i(t) - \frac{\sqrt{3}}{2}$. The same holds for a joint shorten. ■

Lemma 6.21 Assume that a robot r_i executes a hop or a joint hop in round t. Then, $y_i(t+1) \geq y_i(t) - \frac{1}{2}$.

Proof. Observe first that every run vector has a length of at least $\frac{1}{2}$. Otherwise, the robot that initiated the run sequence would have immediately executed a merge. Assume now that r_i has a run state of run sequence κ with $r(\kappa, t+1) = r_{i+1}$ and r_i executes a hop. The largest distance to cover in vertical direction for r_i is $\frac{1}{2}$, in case $\|v_\kappa\|_2 = \frac{1}{2}$and $y_{i+1}(t) = y_i(t) - 1$. Larger vectors v_κ lead to a smaller distance moved in the vertical direction, in case $\|v_\kappa\|_2 > \|w_{i+1}(t)\|_2$, r_i moves even upwards. Smaller vectors $w_i(t+1)$ have the same effect. Thus, $y_i(t+1) \geq y_i(t) - \frac{1}{2}$. The same holds for a joint hop. ■

Lemma 6.3 A run sequence does not visit the same robot twice.

Proof. Assume that the robot r_i starts two run sequences κ_1 and κ_2 in round t_0. In round t_0+1, r_i is located on the midpoint between r_{i-1} and r_{i+1}. Without loss of generality, we assume that r_i is located in the origin of a global coordinate system with r_{i+1} to be located on the y-axis above r_i and r_{i-1} to be located on the y-axis below r_i. The two run vectors are denoted as $v_{\kappa_1} = w_{i+1}(t_0+1)$ and $v_{\kappa_2} = -w_i(t_0+1)$ with $v_{\kappa_1} = -v_{\kappa_2}$. We prove exemplary for κ_2 that it stops before reaching r_i again. The arguments for κ_1 are analogous.

Suppose κ_2 does not stop due to a shorten, joint shorten, a merge or a joint merge. This implies that κ_2 can only execute hops or joint hops in every round. Consider a round t' with $r(\kappa_2, t') = r_j$ and $r(\kappa_2, t'+1) = r_{j-1}$. If r_j executes a hop or joint hop, it holds $\alpha_j(t) > \frac{7}{8}\pi$. Thus, $y_{j-1}(t') < y_j(t')$. Since this holds for every hop and joint hop executed by κ_2 it also holds $y_{j-1}(t') < y_i(t) = 0$.

We now bound $y_{\kappa_2}(t'+2)$: Since only hops are performed, it must hold that the length of the next two vectors together is at least 1 (otherwise, the run sequence stops, and a merge is executed). It follows that $y_{\kappa_2}(t'+2) \leq y_{\kappa_2}(t') - \cos\left(\frac{\pi}{8}\right) = y_{\kappa_2}(t') - \frac{\sqrt{2+\sqrt{2}}}{2}$. Thus, in every second round $y_{\kappa_2}(t)$ decreases by at least $\frac{\sqrt{2+\sqrt{2}}}{2}$.

We now compare the movements of r_i: At the same time, r_i can at most execute every second round of an operation based on a run sequence (since run sequences have a distance of 2). In case of shortens or joint shortens, $y_i(t) \geq y_i(t) - \frac{\sqrt{3}}{2} > y_i(t) - \frac{\sqrt{2+\sqrt{2}}}{2}$ (Lemma 6.20). In case of hops or joint hops, $y_i(t) \geq y_i(t) - \frac{1}{2} > y_i(t) - \frac{\sqrt{2+\sqrt{2}}}{2}$ (Lemma 6.21). Thus, in case r_i executes shortens, joint shortens, hops or joint hops, $\|y_i(t) - y_{\kappa_2}(t)\|_2$ decreases every second round by a constant. The same holds for bisector-operations and star-operations: By its definition, $y_i(t+1) \geq y_i(t) - \frac{1}{5}$ in case of a bisector-operation. For a star-operation it holds $y_i(t+1) \geq y_i(t) - \frac{1}{2}$ (since a robot jumps to the midpoint of two circular arcs). However, no two consecutive star-operations can be executed since either the configuration is a regular star afterward or the robot detects the asymmetry and does not execute a further star-operation (and generates an init state instead). As a consequence, also in case of bisector-operations and star-operations, $\|y_i(t) - y_{\kappa_2}(t)\|_2$ decreases by a constant.

It remains to argue about merges and joint merges. It can happen that $y_i(t)$ decreases by 1 (Lemma 6.19). However, since all run sequences in the distance 4 are stopped, no further run

sequence in the neighborhood of r_i is started within the next 4 rounds; this can at most happen every fourth round. In the same time, however, $y_{\kappa_2}(t)$ decreases by at least $2 \cdot \frac{\sqrt{2+\sqrt{2}}}{2} = \sqrt{2+\sqrt{2}} > 1.8$. Thus, the only time in which $y_{\kappa_2}(t) > y_i(t)$ can hold is within the first 4 rounds after the run sequences κ_1 and κ_2 have been started, more precisely only in the rounds t_0+3 and t_0+4 since r_i can execute its first merge or joint merge earliest in round t_0+2. For every round $t' > t_0+4$ it always holds $y(\kappa_2,t') < y_i(t')$. Hence, κ_2 cannot reach r_i in a round $t' > t_0+4$ since κ_2 can only execute hops or joint hops and to execute a further hop when located at r_{i+1} it must hold $y_i(t') < y_{\kappa_2}(t')$ which is a contradiction. In case κ_2 reaches r_i again in round t_0+3 or t_0+4, the chain only has 5 robots left. Hence, the robots move towards the center of the smallest enclosing circle of all robots, and all run sequences are stopped. ■

Lemma 6.4 At most $143n$ run sequences are required to gather all robots.

Proof. At most n merges or joint merges can happen (Lemma 6.14). Additionally, as stated by Lemma 6.15, there can be at most $\frac{n\cdot\left(1+\frac{1}{5}\right)}{0.019}$ run sequences that stop with a shorten or joint shorten and two vectors of length at least $\frac{1}{2}$. The number of shortens in which one vector has a smaller length is bounded by $4n$ (Lemma 6.17).

A run sequence may be stopped via the progress of a different run sequence or two run sequences together create progress (in joint shortens or joint merges). Every merge or joint merge can stop at most 4 other run sequences. A joint shorten can stop one other run sequence (in case the two run sequences form a joint run pair and only one of them executes the shorten). Thus, every shorten stops at most one additional run sequence. Every joint shorten and joint merge also stops one additional run sequence because two run sequences together have progress. Lastly, in the case of merge and joint merges, at most 4, other run sequences are stopped. Hence, we count at most $8n$ run sequences for merges and joint merges, $2 \cdot \frac{n\cdot\left(1+\frac{1}{5}\right)}{0.019}$ for shortens and joint shortens with vectors of length at least $\frac{1}{2}$, and $8n$ run sequences for shortens and joint shortens in which one vector has a length of at most $\frac{1}{2}$. The total number of run sequences is hence upper bounded by $16n + 2 \cdot \frac{\left(1+\frac{1}{5}\right)}{0.019} \leq 143n$. ■

Lemma 6.5 A configuration that does not become isogonal gathers after at most $5129n$ rounds.

Proof. We make use of a witness argument here. Consider an arbitrary robot r_i with $init(r_i) = true$. The robot r_i tries every 9 rounds to start two new run sequences. In case it does not start new run sequences, it either sees another run sequence or it is blocked by a merge of another run sequence within the last 4 rounds. Since every run states moves to the next robot in every round, $|N_i(t)| = 9$ and no run sequence visits the same robot twice (Lemma 6.3), r_i can at most twice be prevented from starting new run sequences by the same run sequence. Thus, for every 9 rounds, r_i either starts two new run sequences or is hindered by a run sequence. However, it can only be hindered by the same run sequence twice. We say that in $9k$ rounds, for an integer k, r_i is a witness of k run sequences.

It can however happen that r_i executes a merge in some round t. Without loss of generality, we assume that r_i merges with r_{i+1}. Now, we have to consider three cases. Either $init(r_{i+1},t+1) = true$, $init(r_{i+2},t+1) = true$ or $init(r_{i+1},t+1) = init(r_{i+2},t+1) = false$. Assume that $init(r_{i+1},t+1) = true$. In the next iteration when r_{i+1} tries to generate new run sequences it can happen that it is hindered by run sequence that we have already seen at r_i. However, 9 rounds later, this cannot be the case anymore because r_i and r_{i+1} have been direct neighbors. The same argument holds if $init(r_{i+2},t+1) = true$.

What remains is a single case where neither $init(r_{i+1},t+1) = true$ nor $init(r_{i+2},t+1) = true$. This can only happen if r_{i+2} has an init state in round t and executes a merge in the same round.

Then, either r_{i+3} or r_{i+4} has an init state or none of them if r_{i+4} has an init state in round t and executes a merge. However, at the end of such a sequence, there either must be a robot that has an init state or all run sequences along the chain execute a merge in round t. We either continue counting at the robot that has an init state at the end of such a sequence, or we continue counting at an arbitrary new init state that will be generated in the next round. We do not count any run sequence that we have already counted at r_i again in both cases.

All in all, after at most 18 rounds, either a new run sequence is generated, or we count a run sequence while counting the same run sequence at most twice. Thus, after at most $2 \cdot 18 \cdot n \cdot \left(16 + 2 \cdot \frac{\left(1+\frac{1}{5}\right)}{0.019}\right) \leq 5124n$ rounds we have counted enough run sequences that are necessary to gather all robots in a single point (Lemma 6.4). After at most $n-1$ additional rounds, every run sequence has stopped, and the configuration is gathered. As soon as only 5 robots are remaining in the chain, all robots move according to the GO-TO-THE-CENTER protocol which ensures the gathering of 5 robots after 5 more rounds [50]. The lemma follows. ■

6.3.6.3 Symmetric Case

Lemma 6.6 If the configuration is isogonal but not a regular star configuration in round t, the configuration is a regular star configuration in round $t+1$.

Proof. Given that the entire configuration is isogonal, every robot executes a star-operation. All robots lie on the same circle in round t, and the star-operation ensures that all robots stay on the same circle because only target points on the circle's boundary are computed. If the circle's diameter is at most 2, the robots gather in round $t+1$. Now, assume that the diameter has a size of at least 2. This implies that no pair of robots can be connected via a vector that describes the circle's diameter. Thus, the circular arcs L_α and L_β are unique. Consider now a robot r_i and its neighbors r_{i-1} and r_{i+1}. Without loss of generality, assume that $\|w_i(t)\|_2 < \|w_{i+1}(t)\|_2$ and thus L_α connects r_i and r_{i-1} and L_β connects r_i and r_{i+1}. In the star-operation, the robots r_i and r_{i+1} move towards each other and the robots r_i and r_{i-1} move away from each other. Moreover, r_i enlarges L_α by $r \cdot \left(\frac{\beta-\alpha}{4}\right)$ and reduces L_β by the same value. Since r_{i-1} enlarges L_α by the same distance and r_{i+1} reduces L_β by the same value, the new circular arcs $L_\alpha(t+1)$ and $L_\beta(t+1)$ can be computed as follows:

$$L_\alpha(t+1) = L_\alpha + 2 \cdot r \cdot \left(\frac{\beta-\alpha}{4}\right) = r \cdot \alpha + r \cdot \left(\frac{\beta-\alpha}{2}\right) = r \cdot \left(\frac{\alpha+\beta}{2}\right)$$
$$L_\beta(t+1) = L_\beta - 2 \cdot r \cdot \left(\frac{\beta-\alpha}{4}\right) = r \cdot \beta - r \cdot \left(\frac{\beta-\alpha}{2}\right) = r \cdot \left(\frac{\alpha+\beta}{2}\right)$$

Thus, $L_\alpha(t+1) = L_\beta(t+1)$. Since this holds for every robot, the configuration is a regular star configuration at time $t+1$. ■

The next step is to prove that bisector-operations executed in regular star configurations lead to a linear gathering time. We analyze the regular star configuration represented by the regular polygon $\{n/1\}$ with edge length 1. In $\{n/1\}$ and edge length 1, the inner angles are of maximal size, and the distances robots are allowed to move towards the center of the surrounding circle are minimal. Thus, it is enough to prove a linear gathering time for $\{n/1\}$ with edge length 1. Afterward, we can conclude a linear gathering time for any regular star configuration. In the following, we remove for simplicity the assumption that a robot moves at most a distance of $\frac{1}{5}$ in a bisector-operation. We multiply the resulting rounds by 5 and get the same result. For the proof, we introduce some additional notation. Observe first that all angles $\alpha_i(t)$ have the same size in a regular star configuration. Thus we simplify the notation to $\alpha(t)$ in this context. Let $h(t) := \|p_i(t+1) - p_i(t)\|_2$ for any index i be the distance of a robot to its target point. This is

again the same distance for every robot r_i. In addition, define $d(t) := \|p_i(t) - p_{i+1}(t)\|_2$ for any index i. The positions $p_i(t), p_{i+1}(t)$ and $p_i(t+1)$ form a triangle. The angle between $h(t)$ and $d(t)$ has a size of $\frac{\alpha(t)}{2}$. Let $\gamma(t)$ denote the angle between $d(t)$ and the line segment connecting $p_{i+1}(t)$ and $p_i(t+1)$ and $\beta(t)$ be the angle between $h(t)$ and the line segment connecting $p_{i+1}(t)$ and $p_i(t+1)$. See Figure 6.8 for a visualization of these definitions.

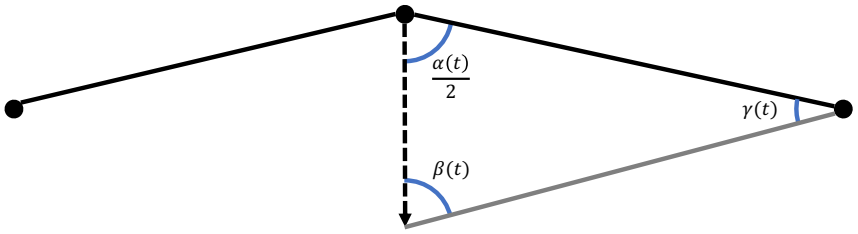

Figure 6.8: The notation for the analysis of regular star configurations.

Next, we state lemmas that analyze how the radius of the surrounding circle decreases. Let $r(t)$ be the radius of the surrounding circle in round t and t_0 the round where the protocol's execution starts.

Lemma 6.22 $r(t+1) \leq r(t) - \sqrt{1 - \frac{4\pi^2 \cdot r(t)^2}{n^2}}$.

Proof. Since we are considering a regular polygon, $\alpha(t) = \frac{(n-2)\pi}{n}$ in every round t. By the law of sines, $h(t) = \frac{\sin(\gamma(t))}{\sin(\frac{\alpha(t)}{2})}$. Also, $\beta(t) = \arcsin(d(t) \cdot \sin(\frac{\alpha(t)}{2}))$ and $\gamma(t) = \pi - \frac{\alpha(t)}{2} - \beta(t)$. Thus, $\sin(\gamma(t)) = \sin(\pi - \frac{\alpha(t)}{2} - \beta(t)) = \sin(\frac{\alpha(t)}{2} + \beta(t))$. Together with $\frac{r(t)}{r(t_0)} = \frac{d(t)}{d(t_0)}$ (intercept theorem), we obtain the following formula for $h(t)$:

$$
\begin{aligned}
h(t) = \frac{\sin(\gamma(t))}{\sin\left(\frac{\alpha(t)}{2}\right)} &= \frac{\sin\left(\frac{\alpha(t)}{2} + \beta(t)\right)}{\sin\left(\frac{\alpha(t)}{2}\right)} \\
&= \frac{\sin\left(\frac{\alpha(t)}{2} + \arcsin\left(d(t) \cdot \sin\left(\frac{\alpha(t)}{2}\right)\right)\right)}{\sin\left(\frac{\alpha(t)}{2}\right)} \\
&= \frac{\sin\left(\frac{(n-2)\pi}{2n} + \arcsin\left(\frac{r(t)}{r(t_0)} \cdot \sin\left(\frac{(n-2)\pi}{2n}\right)\right)\right)}{\sin\left(\frac{(n-2)\pi}{2n}\right)} \\
&= \frac{\sin\left(\frac{(n-2)\pi}{2n} + \arcsin\left(\frac{2\pi r(t)}{n} \cdot \cos\left(\frac{\pi}{n}\right)\right)\right)}{\cos\left(\frac{\pi}{n}\right)} \\
&= \frac{\sin\left(\frac{\pi}{2} - \frac{\pi}{n} + \arcsin\left(\frac{2\pi r(t)}{n} \cdot \cos\left(\frac{\pi}{n}\right)\right)\right)}{\cos\left(\frac{\pi}{n}\right)} \\
&= \frac{\cos\left(\frac{\pi}{n} - \arcsin\left(\frac{2\pi r(t)}{n} \cdot \cos\left(\frac{\pi}{n}\right)\right)\right)}{\cos\left(\frac{\pi}{n}\right)}
\end{aligned}
$$

Next, we apply the trigonometric identities $\cos(x-y) = \sin(x) \cdot \sin(y) + \cos(x) \cdot \cos(y)$ and $\cos(\arcsin(x)) = \sqrt{1-x^2}$ and obtain:

$$
\begin{aligned}
h(t) &= \frac{\cos\left(\frac{\pi}{n} - \arcsin\left(\frac{2\pi r(t)}{n} \cdot \cos\left(\frac{\pi}{n}\right)\right)\right)}{\cos\left(\frac{\pi}{n}\right)} \\
&= \sqrt{1 - \frac{4\pi^2 \cdot r(t)^2}{n^2} \cdot \cos^2\left(\frac{\pi}{n}\right)} + \frac{2\pi \cdot r(t)}{n} \cdot \sin\left(\frac{\pi}{n}\right) \\
&\geq \sqrt{1 - \frac{4\pi^2 \cdot r(t)^2}{n^2}}
\end{aligned}
$$

Finally, we can prove the lemma:

$$
r(t+1) = r(t) - h(t) \leq r(t) - \sqrt{1 - \frac{4\pi^2 \cdot r(t)^2}{n^2}}
$$

■

Now, define by $\Delta r(t) = r(t_0) - r(t)$. For $\Delta r(t+1)$, we can derive the formula stated by the following lemma.

Lemma 6.23 For $\Delta r(t) \geq 1$: $\Delta r(t+1) \geq \Delta r(t) + \frac{\sqrt{\Delta r(t)}}{\sqrt{n}}$

Proof. First of all, observe $\Delta r(t+1) = \Delta r(t) + h(t)$.

$$
\begin{aligned}
\Delta r(t+1) &\geq \Delta r(t) + \sqrt{1 - \frac{4\pi^2 \cdot (r(t_0) - \Delta r(t))^2}{n^2}} \\
&= \Delta r(t) + \sqrt{1 - \frac{4\pi^2 \cdot \left(\frac{n}{2\pi} - \Delta r(t)\right)^2}{n^2}} \\
&= \Delta r(t) + \sqrt{\frac{4\pi \cdot \Delta r(t)}{n} - \frac{4\pi^2 \cdot \Delta r(t)^2}{n^2}} \\
&= \Delta r(t) + \frac{2\sqrt{\pi}\sqrt{\Delta r(t)}}{\sqrt{n}} \sqrt{1 - \frac{\pi \cdot \Delta r(t)}{n}} \\
&\geq \Delta r(t) + \frac{2\sqrt{\pi}\sqrt{\Delta r(t)}}{\sqrt{2}\sqrt{n}} \\
&= \Delta r(t) + \frac{\sqrt{2}\sqrt{\pi}\sqrt{\Delta r(t)}}{\sqrt{n}} \\
&\geq \Delta r(t) + \frac{\sqrt{\Delta r(t)}}{\sqrt{n}}
\end{aligned}
$$

■

Lemma 6.24 After $2n$ rounds, $\Delta r(t) \geq 1$.

Proof. In the proof of Lemma 6.22, we identified $h(t) \geq \frac{2\pi \cdot r(t)}{n} \cdot \sin\left(\frac{\pi}{n}\right)$. Now, assume that $r(t) \geq \frac{r(t_0)}{2} = \frac{n}{4\pi}$ (which holds for sufficiently large n since $\Delta r(t) < 1$). Then, $h(t) \geq \frac{2\pi \cdot n}{4\pi} \cdot \sin\left(\frac{\pi}{n}\right) = \frac{1}{2} \cdot \sin\left(\frac{\pi}{n}\right)$. For $n \geq 2$ it now holds $h(t) \geq \frac{\pi}{4n}$. Thus, after $2n$ rounds it holds $\Delta r(t) \geq 1$. ■

Lemma 6.25 After at most $6n$ rounds, $\Delta r(t) \geq \frac{n}{2\pi}$ and thus $r(t) = 0$.

Proof. We fix the first time step t' such that $\Delta r(t') \geq 1$ (note that $\Delta r(t') \leq 2$ in this case since no robot moves more than a distance of 1 per round). By Lemma 6.24 this holds after at most $2n$ rounds. Furthermore, $\Delta r(t)$ doubles every $\sqrt{n} \cdot \sqrt{\Delta r(t)}$ rounds (Lemma 6.23). After at most $\log n$ doublings, it holds $\Delta r(t) \geq n$, and the robots are gathered. The first doubling requires $\sqrt{n} \cdot \sqrt{\Delta r(t')}$ rounds, the next doublings $\sqrt{n} \cdot \sqrt{2 \cdot \Delta r(t')}$, $\sqrt{n} \cdot \sqrt{4 \cdot \Delta r(t')}$, ... rounds. Thus, the number of rounds for $\log n$ doublings can be counted as follows:

$$\sqrt{\Delta r(t')} \cdot \sqrt{n} \sum_{k=1}^{\log n} \sqrt{2^k} = \sqrt{\Delta r(t')} \cdot \sqrt{n} \cdot \sqrt{2} \left(1 + \sqrt{2}\right) \left(\sqrt{n} - 1\right) \leq 4n$$

Therefore, the total number of rounds can be upper bounded by $6n$. ■

Lemma 6.7 Regular star configurations gather in at most $30n$ rounds.

Proof. Generally, the bisectors of regular star configurations intersect in the center of C. Thus, with every bisector-operation, the distance of a robot to the center of C decreases until finally, all robots gather. We prove the runtime for regular star configurations $\{n/1\}$, which are also denoted as regular polygons. These polygons have inner angles $\alpha_i(t)$ of maximal size; for higher values of d (referring to the Schläfli symbol $\{n/d\}$), the inner angles become smaller. Thus, for the regular polygon $\{n/1\}$, the distance a robot is allowed to move within a bisector-operation is minimal among all regular star polygons. Lemma 6.25 states that the regular star configuration $\{n/1\}$ with side length 1 is gathered in $6n$ rounds. Since we did not consider the assumption that a robot moves a distance of at most $\frac{1}{5}$ per round, we multiply the result with 5 and get an upper bound of $30n$ on the number of required rounds. Hence, all other regular star configurations can be gathered in linear time as the robots can move larger distances per round. ■

6.4 Synchronization for the $\mathcal{S}$SYNC and $\mathcal{A}$SYNC Schedulers

The asymmetric part of the CC-HOPPER protocol uses the same movement operations as the ε-2-HOPPER protocol for the CHAIN-FORMATION problem. Those movements can be transferred to the $\mathcal{S}$SYNC and $\mathcal{A}$SYNC schedulers, as already seen in Chapter 5.

We need one additional synchronization procedure for small chains (at most 5 robots are remaining). In this case, all robots move towards the center of the smallest enclosing circle of the remaining robots. Without additional synchronization, GATHERING is impossible in $\mathcal{S}$SYNC. The slight change is as follows: As long as the radius of the smallest enclosing circle of all remaining robots is larger than $\frac{1}{2}$, the robots continue to execute GTC. Otherwise, (the radius is at most $\frac{1}{2}$, and thus all robots are in the distance at most 1 of each other), robots move towards the center of the smallest enclosing circle in case all other robots have the same value of the epoch counter. If at least one epoch counter is larger, robots move simply to a robot with a larger epoch counter. This ensures GATHERING as either all robots are activated simultaneously and gather immediately or some robots have activated that move to the center of the smallest enclosing circle and all other robots follow as soon as they are activated.

Lastly, the CC-HOPPER protocol also consists of a protocol for isogonal configurations. However, this does not require further synchronization as all robots are activated simultaneously, or some robots move according to the symmetric protocol, and some do not move. In the latter case, the exceptional generation of init states ensures that this can happen at most n times, and thus, GATHERING is done after $\mathcal{O}(n)$ epochs. We denote the CC-HOPPER protocol extended by the synchronization techniques as the CC-HOPPER-ASYNC protocol.

Theorem 6.2 The CC-HOPPER-ASYNC protocol gathers a closed chain of robots in $\mathcal{O}(n)$ epochs in the $\mathcal{LUMI}_4^A$ model. The protocol can be implemented with a constant number of colors.

6.5 Conclusion & Outlook

We introduced and analyzed the CC-HOPPER protocol that gathers a closed chain of disoriented robots with limited visibility in the Euclidean plane in linear time. The linear runtime originates in a combination of run sequences implemented with help of the $\mathcal{LUMI}$ model and a complete characterization of symmetries, where the ideas of run sequences cannot be applied. By understanding the symmetries, we designed an additional protocol that gathers these configurations in linear time. The CC-HOPPER protocol is the first linear time GATHERING protocol for disoriented robots with limited visibility in the Euclidean plane.

Nevertheless, there are lots of questions left to answer for future research. First of all, there are some questions regarding the CC-HOPPER protocol itself. The protocol uses a viewing range of 4 which is necessary for the vector length patterns to detect asymmetries to start new run sequences. However, it is not clear whether the viewing range of 4 is needed in general. Hence, we raise the following question: Is there a linear time protocol to gather a Euclidean closed chain of disoriented robots with limited visibility that requires a viewing range of 3 or less? Moreover, we have analyzed an upper runtime bound of $5129n$ epochs. Although the runtime is *asymptotically* optimal, the constant 5129 could be reduced with a more sophisticated analysis technique. Currently, we bound for each operation (e.g., (joint) shorten and (joint) merge) how often it could be executed in the worst case without taking into account other operations. For instance, if initially $n-3$ merges are executed, it cannot happen that still, $\Omega(n)$ (joint) shortens are executed since the remaining chain has only a constant length. Such bounds, however, depend heavily on the order of the operations that occur and, thus, are non-trivial to establish.

Furthermore, it remains open whether a linear time protocol for disoriented robots with limited visibility in the Euclidean plane also exists in the standard connectivity model, i.e., without an a priori given closed chain. The main idea would be to apply (ideas of) the CC-HOPPER protocol to the boundaries of the swarm since the boundary itself forms a closed chain. Similarly, this has been done on the grid [2, 40] (although the concrete protocol is much more involved compared to simply applying the closed chain protocol to the boundaries). However, in our first approach, we identified severe challenges. First of all, it is not clear where the boundary exactly is. Robots can locally guess whether they could be part of the global boundary but can never be sure. Moreover, they cannot necessarily determine which of their neighbors are also part of the boundary. Additionally, assuming that robots can determine that they are part of the boundary, applying the CC-HOPPER protocol could destroy the connectivity of the swarm to other robots that are not part of the boundary. Hence, we currently observe several challenges that need to be carefully analyzed in future research.

The last open question is about a general lower bound: Is the $\mathcal{LUMI}$ model needed to reach a linear time GATHERING protocol or does such a protocol also exist in the $\mathcal{OBLOT}$ model? We elaborated on this question already in Section 4.5.

Expanding Problems

7. The MAX-CHAIN-FORMATION Problem

In Part I, we studied formation problems where robots have to move closer together (GATHERING) or to minimize a target structure (CHAIN-FORMATION). Such tasks are basic subroutines for recollecting spread-out robots or optimizing a communication interface realized by a chain of relay robots. The following two chapters shift the focus to formation tasks that have an opposite goal: to spread the robots out to cover a large area. Algorithmically, contracting formation problems seem to be easier to solve since robots can locally guess where the inside of the swarm is. The main challenge of expanding formation problems is that robots locally have to spread out not knowing in which direction other robots potentially move. Probably because of these additional challenges and since the research area is relatively young, there is not much known about expanding formation problems for robots with limited visibility in the literature yet. Therefore, we focus on a conceptually very simple formation: a line of maximal length (concerning the viewing ranges of the robots). As we will observe, also this simple structure adds lots of challenges both on the algorithmic as well as on the analysis side. In this chapter, we study the MAX-CHAIN-FORMATION problem, where the robots are connected in an open chain and the outer robots can move. The goal is to maximize the length of the chain. The results in this chapter are based on the following journal article.

> **2022** (with P. Kling, T. Knollmann and F. Meyer auf der Heide) "A Discrete and Continuous Study of the Max-Chain-Formation Problem" In: *Information and Computation*, cf. [30].

A preliminary version appeared in the conference proceedings of SSS 2020:

> **2020** (with P. Kling, T. Knollmann and F. Meyer auf der Heide) "A Discrete and Continuous Study of the Max-Chain-Formation Problem – Slow down to Speed Up" In: *Proceedings of the 22nd International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, Best Paper Award, cf. [29].

7.1 Contribution

We initiate the study of the MAX-CHAIN-FORMATION in two different time models: the $\mathcal{F}$SYNC scheduler and the continuous time model. More precisely, we study the $\mathcal{OBLOT}_1^F$ and $\mathcal{OBLOT}_1^C$ models with an open chain of robots. On the algorithmic side, we adapt the known (contracting) CHAIN-FORMATION protocols GTM [60] ($\mathcal{F}$SYNC) and MOB [49] (continuous time) such that they still straighten the chain but, at the same time, keep extending its length. In the GTM protocol,

inner robots move in every round to the midpoint between their neighbors. The MOB protocol moves inner robots with a speed of 1 along the angle bisector between vectors pointing to their neighbors. Our basic idea for the MAX-CHAIN-FORMATION problem is to let inner robots perform the contracting protocol while the two outer robots extend the chain by moving away from their respective neighbor. While this seems to be a minor modification of the contracting protocols on a conceptual level, we identify a much more complex behavior of the robots caused by the extension. Also, the analysis is much more complex – we use several different techniques: among others, we use discrete Fourier transforms, the mixing time of Markov Chains, and the stability theory of dynamical systems.

Section 7.3 considers the $\mathcal{OBLOT}_1^F$ model, for which we distinguish the one-dimensional case (all robots are initially collinear) and the general two-dimensional case. In the one-dimensional case, we see that symmetric configurations are problematic for any (deterministic) protocol. This is obvious for the trivial configuration, where all robots start in the same spot. However, also from less contrived starting positions (e.g., when the initial chain is symmetrical around the origin), any deterministic protocol results in a non-maximal chain (that potentially keeps moving) (see Theorem 7.4). In the following, we denote the desired configuration, a straight line of length $n-1$ by a *max-chain*. On the contrary, an undesired configuration that keeps moving through the plane forever is called a *marching chain* (and is formally defined in Section 7.3.2).

Theorem 7.1 Under the MAX-GTM protocol on the line, the robot movement reaches in time $\Omega\left(n^2 \cdot \log(1/\varepsilon)\right)$ and $\mathcal{O}\left(n^2 \cdot \log(n/\varepsilon)\right)$ an ε-approximation of a stationary, max-chain of length $n-1$, if the outer robots move in different directions after $n-2$ rounds or a chain of non-maximal length moving at speed $1/n$ (*marching chain*) if the outer robots move in the same direction after $n-2$ rounds.

While this gives a nearly complete picture of the one-dimensional case, the two-dimensional case exhibits a much more complex behavior. We can still prove convergence in finite time and derive a lower bound (which now depends also on the outer robots' initial distance) but an upper bound remains elusive.

Theorem 7.2 Under the MAX-GTM protocol, the robot movement reaches an ε-approximation either of the max-chain or a one-dimensional marching chain. There are configurations for which this takes $\Omega(n^2 \cdot \log(1/\delta))$ rounds, where δ denotes the initial distance between the outer robots.

Interestingly, however, fixing the position of one of the two outer robots enables us to employ tools from Markov Chain theory (as used in previous results [88]), yielding the same almost tight runtime bound as in the one-dimensional case (see Theorem 7.11). In combination with simple experimental evaluations, our results indicate that the worst-case lower bound stated in Theorem 7.2 is tight.

Section 7.4 considers the $\mathcal{OBLOT}_1^C$ model. As in the discrete setting, very symmetric configurations lead to unavoidable problems for deterministic strategies. Moreover, a naïve translation of the MOB protocol results in the same dependency on the outer robots' initial distance δ. However, the continuous model allows for an interesting tweak which, as we show in Section 7.5, cannot be done in the discrete model. Namely, it turns out that decreasing the speed of outer robots by a small constant gets rid of the dependency on δ and yields an optimal, linear runtime bound. As a byproduct, this also causes symmetrical initial positions to collapse to a single point instead of becoming a marching chain. Summarized, we get the following result for our protocol MAX-MOB designed for the $\mathcal{OBLOT}_1^C$.

Theorem 7.3 MAX-MOB reaches in worst-case optimal time $\Theta(n)$ a stationary, maximum chain of length $n-1$ or the chain collapses to a single point.

Our results show that the idealized continuous model yields a linear speed-up for the MAX-CHAIN-FORMATION problem, similar to contracting robot formation problems. The major open problem is to find an upper runtime bound for MAX-GTM in the discrete setting where both endpoints move. Moreover, while very symmetrical initial configurations pose a problem for deterministic protocols, both experiments with a simple, custom simulator and looking at our protocols from the perspective of dynamical systems suggest that such configurations are few and unstable. Thus, minor, random perturbations usually yield a configuration where the robots reach the desired maximal chain. We analyze this observation formally by proving that the marching chain is an unstable fixed point of the related dynamical system. We discuss this in more detail in Section 7.3.3.2.

7.2 Model Recap and Preliminaries

We study the $\mathcal{OBLOT}_1^F$ (MAX-GTM) and $\mathcal{OBLOT}_1^C$ (MAX-MOB) models in combination with an open chain of robots. The most important model parameters are summarized in Table 7.1, and a complete model definition can be found in Chapter 2.

Protocol	Time	Dimension	Viewing Range	Orientation	Chain
MAX-GTM	$\mathcal{F}$SYNC	1 & 2	1	disoriented	yes (open)
MAX-MOB	Continuous	1 & 2	1	disoriented	yes (open)

Table 7.1: A summary of the most important model details for the MAX-GTM and MAX-MOB protocols.

Problem Statement. Throughout the chapter, we use the notation $\Delta_{i,j}(t) := \|p_i(t) - p_j(t)\|_2$ to describe the Euclidean distance between r_i and r_j at time t. The goal of the MAX-CHAIN-FORMATION problem is to reach a configuration with $\Delta_{0,n-1}(t) = n-1$, i.e., the outer robots reach a distance of $n-1$ which can only be the case if the chain is connected and all robots are arranged on a straight line of maximal length. More precisely, each vector $w_i(t)$ (recall that $w_i(t) = p_i(t) - p_{i-1}(t)$) should have a length of 1 and $w_i(t) = w_{i+1}(t)$ for $1 \leq i \leq n-2$. We call this configuration a *max-chain*. We say that we have reached an ε-approximation of the max-chain if $\Delta_{0,n-1}(t) \geq (1-\varepsilon)\cdot(n-1)$ and $\|w_i(t)\|_2 > 1-\varepsilon$ for all $1 \leq i \leq n-1$.

Definitions & Notation. Next, we introduce a characterization of configurations that is relevant to our analyses. In *one-dimensional* configurations, the positions of all robots are collinear. In *two-dimensional* configurations, there exists a set of at least 3 robots whose positions are not collinear. Our analyses distinguish two special kinds of one-dimensional configurations: *Opposed configurations* and *marching configurations*. Recall that $\widehat{w}_i(t)$ represents the normalized vector $\frac{w_i(t)}{\|w_i(t)\|_2}$. In opposed configurations, the outer robots are on different sides of their neighbors, i.e., $\widehat{w}_1(t) = \widehat{w}_{n-1}(t)$. In marching configurations, the outer robots are on the same side of their neighbors, i.e., $\widehat{w}_1(t) = -\widehat{w}_{n-1}(t)$.

7.3 Protocols and Analyses in the $\mathcal{OBLOT}_1^F$ Model

In this section, we describe MAX-GTM for the $\mathcal{F}$SYNC time model. Intuitively, the protocol solves two tasks concurrently. The first task is to arrange all robots in a straight line, while the second task is to lengthen the chain by moving the outer robots away from each other. For the first task, we

adapt the GTM-protocol for CHAIN-FORMATION in which all inner robots move to the midpoint between their neighbors each round. The outer robots move as far as possible away from their neighbors for the second task while keeping the chain connected.

7.3.1 MAX-GO-TO-THE-MIDDLE (MAX-GTM)

MAX-GTM works as follows: Every inner robot moves to the midpoint between its neighbors. An outer robot moves as far as possible away from its neighbor. To do so, in round t, the outer robot r_0 normalizes the vector $w_1(t)$, imagines a virtual robot r_{-1} positioned at $p_0(t) - \widehat{w}_1(t)$ and moves to the midpoint between r_{-1} and r_1. The procedure works analogously for r_{n-1}. In the special case $p_0(t) = p_1(t)$ ($p_{n-2}(t) = p_{n-1}(t)$ respectively), r_0 (r_{n-1}) does not move. See Figure 7.1 for a visualization of the movements.

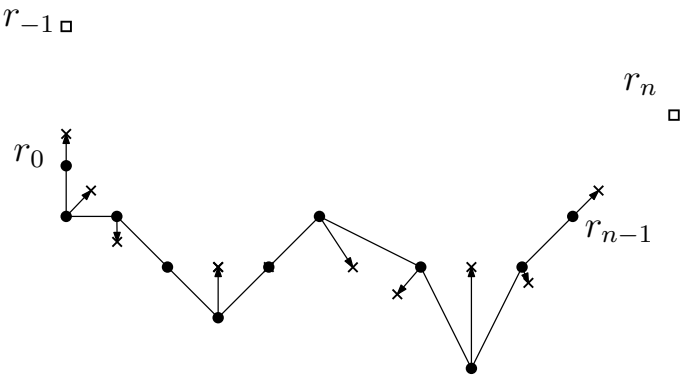

Figure 7.1: A visualization of MAX-GTM. The target point of each robot is marked by a cross. Both r_{-1} and r_n are virtual robots.

Formally, assuming a global coordinate system not known to the robots, the positions of all robots in the next round can be computed as follows.

$$p_0(t+1) = \frac{1}{2}p_0(t) + \frac{1}{2}p_1(t) - \frac{1}{2}\widehat{w}_1(t) \tag{7.1}$$

$$p_i(t+1) = \frac{1}{2}p_{i-1}(t) + \frac{1}{2}p_{i+1}(t) \qquad \text{for } 1 \leq i < n-1 \tag{7.2}$$

$$p_{n-1}(t+1) = \frac{1}{2}p_{n-2}(t) + \frac{1}{2}p_{n-1}(t) + \frac{1}{2}\widehat{w}_{n-1}(t) \tag{7.3}$$

Based on Equations (7.1) to (7.3) we can derive formulas for the vectors $w_i(t+1)$.

$$\begin{aligned} w_1(t+1) &= p_1(t+1) - p_0(t+1) \\ &= \frac{1}{2}p_0(t) + \frac{1}{2}p_2(t) - \frac{1}{2}p_0(t) - \frac{1}{2}p_1(t) + \frac{1}{2}\widehat{w}_1(t) \\ &= \frac{1}{2}\widehat{w}_1(t) + \frac{1}{2}w_2(t) \end{aligned} \tag{7.4}$$

$$\begin{aligned} w_i(t+1) &= p_i(t+1) - p_{i-1}(t+1) \\ &= \frac{1}{2}p_{i-1}(t) + \frac{1}{2}p_{i+1}(t) - \frac{1}{2}p_{i-2}(t) - \frac{1}{2}p_i(t) \\ &= \frac{1}{2}w_{i-1}(t) + \frac{1}{2}w_{i+1}(t) \\ w_{n-1}(t+1) &= p_{n-1}(t+1) - p_{n-2}(t+1) \\ &= \frac{1}{2}p_{n-2}(t) + \frac{1}{2}p_{n-1}(t) + \frac{1}{2}\widehat{w}_{n-1}(t) - \frac{1}{2}p_{n-3}(t) - \frac{1}{2}p_{n-1}(t) \\ &= \frac{1}{2}w_{n-2}(t) + \frac{1}{2}\widehat{w}_{n-1}(t) \end{aligned} \tag{7.5}$$

Next, define $w(t) = (w_1(t), w_2(t), \ldots, w_{n-1}(t))^T$ as the matrix where the i-th row corresponds to the vector $w_i(t)$. Simplified, we can compute $w(t+1)$ as a matrix-vector product: $w(t+1) = S(t) \cdot w(t) = \prod_{i=t}^{0} S(i) \cdot w(0)$ with the protocol matrix

$$S(t) = \begin{bmatrix} \frac{1}{2\cdot\|w_1(t)\|_2} & \frac{1}{2} & 0 & 0 & \ldots & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & \ldots & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & \ldots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & \ldots & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & \ldots & 0 & 0 & \frac{1}{2} & \frac{1}{2\cdot\|w_{n-1}(t)\|_2} \end{bmatrix}.$$

The vector representation already leads to an important lemma for the analysis: MAX-GTM always maintains the connectivity of the chain.

Lemma 7.1 Applying the MAX-GTM protocol to a connected configuration in round t leads to a connected configuration in round $t+1$.

Proof. Above, we have seen $w_1(t+1) = \frac{1}{2}\widehat{w}_1(t) + \frac{1}{2}w_2(t)$, $w_i(t+1) = \frac{1}{2}w_{i-1}(t) + \frac{1}{2}w_{i+1}(t)$ for all $1 < i < n-1$ and $w_{n-1}(t+1) = \frac{1}{2}w_{n-2}(t) + \frac{1}{2}\widehat{w}_{n-1}(t)$. Since each vector $w_j(t)$ $(0 \leq j \leq n-1)$ has a length of at most 1 at time t, we conclude by applying the triangle inequality that each vector has a length of at most 1 at time $t+1$ as well. ■

Throughout the rest of the analysis, we take Lemma 7.1 for granted and do not always mention explicitly that the connectivity is maintained.

7.3.2 One-Dimensional Analysis

Next, we investigate the performance of MAX-GTM in a one-dimensional configuration. These configurations already reveal an interesting behavior of MAX-GTM: some configurations do not converge to a max-chain, but to a different structure, which we denote as the *marching chain*, i.e., an undesired configuration that keeps moving through the plane forever. The two classes of configurations that play a role in this analysis are marching and opposed configurations. For the analysis, we assume that the robots are distributed on the x-axis of a two-dimensional Cartesian coordinate system (not known to the robots). Hence, the vectors $w_i(t)$ are one-dimensional in the following. We divide the analysis into two parts based on the initial configuration of the robots. For opposed configurations, we assume without loss of generality that r_0 moves to the left (into negative direction; $w_1(t) > 0$) and r_{n-1} moves to the right (into positive direction; $w_{n-1}(t) > 0$). Similarly, for marching configurations, we assume that both r_0 and r_{n-1} move to the right and thus $w_1(t) < 0$ and $w_{n-1}(t) > 0$. Define d to be -1 for opposed configurations and 1 for marching configurations. Then, Equations (7.1) and (7.3) as well as Equations (7.4) and (7.5) simplify to

$$\begin{aligned} p_0(t+1) &= \frac{1}{2}p_0(t) + \frac{1}{2}p_1(t) + \frac{1}{2}d \\ p_{n-1}(t+1) &= \frac{1}{2}p_{n-2}(t) + \frac{1}{2}p_{n-1}(t) + \frac{1}{2} \\ w_1(t+1) &= \frac{1}{2}w_2(t) - \frac{1}{2}d \\ w_{n-1}(t+1) &= \frac{1}{2}w_{n-2}(t) + \frac{1}{2}. \end{aligned}$$

In some exceptional cases, a marching configuration may be transformed into an opposed configuration or vice versa. Luckily, we can prove that this happens at most once within the first $n-2$ rounds. Afterward, no further switch can happen.

Lemma 7.2 MAX-GTM switches at most once between opposed and marching configurations. The switch is executed after at most $n-2$ rounds.

Proof. Consider an outer robot, for example, r_0. Without loss of generality, we assume that initially $p_0(t) < p_1(t)$ and that r_0 is passed by r_1, i.e., $p_0(t+1) > p_1(t+1)$. Then,

$$\begin{aligned} p_0(t+1) > p_1(t+1) &\iff \frac{1}{2}p_0(t) + \frac{1}{2}p_1(t) - \frac{1}{2} > \frac{1}{2}p_0(t) + \frac{1}{2}p_2(t) \\ &\iff p_1(t) - 1 > p_2(t). \end{aligned}$$

This implies that the distance between the two robots r_1 and r_2 was greater than 1 at time t which is a contradiction since the configuration is always connected (Lemma 7.1). Thus, r_1 cannot pass r_0 in a single round. However, r_0 and r_1 could move to the same position. As a consequence, r_0 does not move in the next round and thus r_1 could pass r_0 in two rounds. We prove that this can only happen once if the initial configurations have specific properties. To do so, we start by analyzing under which conditions r_0 and r_1 move to the same position in the next round. The arguments for r_{n-2} and r_{n-1} are always analogous and for readability omitted here.

$$\begin{aligned} p_0(t+1) = p_1(t+1) &\iff \frac{1}{2}p_0(t) + \frac{1}{2}p_1(t) - \frac{1}{2} = \frac{1}{2}p_0(t) + \frac{1}{2}p_2(t) \\ &\iff p_1(t) - 1 = p_2(t) \\ &\iff p_2(t) - p_1(t) = -1 \end{aligned}$$

With analogous calculations for the mirrored case $p_0(t) > p_1(t)$, we obtain $w_2(t) = -\widehat{w}_1(t)$. Observe that especially $\|w_2(t)\|_2 = 1$ must hold. Now, consider any round t' such that $\|w_2(t')\|_2 < 1$. In the next round, $\|w_2(t'+1)\|_2 = 1$ can only hold if $w_1(t') = w_3(t')$ and $\|w_1(t')\|_2 = \|w_3(t')\|_2 = 1$ as $w_2(t'+1) = \frac{1}{2}w_1(t') + \frac{1}{2}w_3(t')$. In a similar way, one can see that $\|w_2(t')\|_2 = 1$ must hold to obtain $\|w_1(t'+1)\| = 1$. Finally, consider any round $\hat{t}$ such that $p_0(\hat{t}) = p_1(\hat{t})$ (the necessary precondition to obtaining a switch). As $p_0(\hat{t}) = p_1(\hat{t})$, we have that $\|w_1(\hat{t})\|_2 = 0$. Based on our observations above, $\|w_1(\hat{t}+1)\|_2 < 1$ and $\|w_2(\hat{t}+1)\|_2 < 1$. Moreover, in every future round $t'' > \hat{t}$ it will hold both $\|w_1(t''+1)\|_2 < 1$ and $\|w_2(t''+1)\|_2 < 1$ and hence, no second switch could occur (as $\|w_2(t'')\|_2 = 1$ must hold to obtain a switch).

It remains to argue that the precondition for the switch ($p_0(t) = p_1(t)$) can only be fulfilled in the first two rounds. Assume that the computation starts in round 0. Then, either $p_0(0) = p_1(0)$ or $w_2(0) = -\widehat{w}_1(0)$. Assume that $p_0(0) \neq p_1(0)$ and $w_2(0) \neq -\widehat{w}_1(0)$. Since, for any t, $w_2(t+1) = \frac{1}{2}w_1(t) + \frac{1}{2}w_3(t)$ there cannot be any round t' in the future such that $w_2(t') = -\widehat{w}_1(t')$. Thus, a switch can only happen if either $p_0(0) = p_1(0)$ or $w_2(0) = -\widehat{w}_1(0)$. To complete the switch, r_1 and r_0 must finally swap their positions. Either all robots are located in the same position or, if there is at least one vector of length larger than 0 in the chain, the swap will be obtained after at most $n-2$ rounds, since $w_1(t+1) = \frac{1}{2}\widehat{w}_1(t) + \frac{1}{2}w_2(t)$, $w_1(t+2) = \frac{1}{2}\widehat{w}_1(t+1) + \frac{1}{2}w_2(t+1) = \frac{1}{4}\widehat{w}_1(t+1) + \frac{1}{4}w_1(t) + \frac{1}{4}w_3(t)$ and so on (after $t+x$ rounds, $w_1(t+x)$ consists of a term $w_{1+x}(t)$ and thus, after $n-2$ rounds, $w_1(t+n-2)$ consists of a term for each initial $w_i(t)$).

■

As a summary, the switch between opposed and marching configurations can only happen if $w_2(0) = -\widehat{w}_1(0)$ or $w_{n-2}(0) = -\widehat{w}_{n-1}(0)$. In this case, r_0 and r_1 or r_{n-2} and r_{n-1} move to the same position and it can take up to $n-2$ rounds until all robots have distinct positions. As soon as all robots have distinct positions afterward, the configuration remains either a marching configuration or an opposed configuration. For ease of notation, we say in the following that a configuration is an

opposed configuration if it is an opposed configuration after applying MAX-GTM for $n-2$ rounds (similar for marching configurations).

As a consequence of Lemma 7.2, starting from a marching configuration, MAX-GTM does not converge to a max-chain. For some highly symmetric configurations, for instance, the configuration depicted in Figure 7.2 (contained in Section 7.3.2.2), this even cannot be obtained by any deterministic protocol due to a high symmetry.

Theorem 7.4 There are marching configurations that cannot be transformed into a max-chain by any deterministic protocol.

Proof. Assume that there is a deterministic protocol s that transforms an initial marching configuration into a max-chain. Then s must switch the marching configuration to an opposed configuration. Consider an initial marching configuration $w(0)$ that is symmetric in the sense that the local situation of any robot r_i is equivalent to the local situation of robot r_{n-1-i} for each $0 \leq i \leq n-1$. Assume that s transforms $w(0)$ into an opposed configuration at time step t. Thus, r_0 changes its direction at time t. Since the configuration is symmetric, s is deterministic, and the robots are anonymous, r_{n-1} changes its direction at time t as well. Then, $w(t+1)$ is still a marching configuration which poses a contradiction. ■

7.3.2.1 Analysis of Opposed Configurations

For opposed configurations, we can show that MAX-GTM converges to the max-chain.

Theorem 7.5 Started in an opposed configuration, MAX-GTM needs at most $\mathcal{O}(n^2 \cdot \log(n/\varepsilon))$ rounds to achieve an ε-approximation of the max-chain.

Proof. We use the potential $\phi_1(t) = \sum_{i=1}^{n-1}(1-\|w_i(t)\|_2)^2$ for a time step t. First, observe that for $m_i(t) = 1-\|w_i(t)\|_2$, we have that $\phi_1(t) = \sum_{i=1}^{n-1} m_i(t)^2$ and $\phi_1(t+1) = \sum_{i=1}^{n-1}\left(\frac{m_{i-1}(t)+m_{i+1}(t)}{2}\right)^2$. Also, since the virtual robots r_{-1} and r_n are always at a distance of 1 to their neighbors, $m_0(t) = m_n(t) = 0$. The potential function $\phi_1(t)$ is nearly identical to the function $\psi(t)$ of [38]. In particular, this allows us to apply the same discrete sine transformations and the proof of Theorem 2.2 of [38] works analogously up to Equation 5 resulting in $\phi_1(t+1) \leq \cos^2\left(\frac{\pi}{n+1}\right)\phi_1(t) \leq \cos^{2t}\left(\frac{\pi}{n+1}\right)\phi_1(0)$ for any time step t. Note that the difference in the denominator comes from the fact that we consider n moving robots in contrast to the $N-2$ moving robots assumed in [38]. Next, using standard methods and Taylor's theorem, we can derive that for $y \in \mathbb{R}$ and $0 \leq y \leq \frac{\pi}{2}$: $\cos(y) \leq \left(1-\frac{y^2}{4}\right)^2$. Setting $t = \left\lceil \frac{(n+1)^2}{\pi^2}\ln\left(\frac{(n-1)^2}{\varepsilon^2}\right)\right\rceil \in \mathcal{O}(n^2\log(n/\varepsilon))$ then yields $\phi_1(t) \leq \frac{\varepsilon^2}{(n-1)^2}\cdot\phi_1(0) \leq \frac{\varepsilon^2}{n-1}$. The last inequality holds because $\phi_1(0)$ is upper bounded by $n-1$. Obviously, there cannot be any vector $w_i(t)$ fulfilling $\|w_i(t)\|_2 < 1-\varepsilon$ since $\phi_1(t) \leq \frac{\varepsilon^2}{n-1}$ holds. Thus, for each individual vector w_i, $\|w_i\|_2 > 1-\varepsilon$ and thus, a ε-approximation of the max-chain is achieved. ■

The analysis of the *mixing time* of a Markov Chain allows us to prove a close lower bound of $\Omega\big(n^2 \cdot \log(1/\varepsilon)\big)$ rounds. Here, we can rewrite $w(t)$ and $S(t)$ slightly such that the resulting protocol matrix is a stochastic matrix that can be interpreted as the transition matrix of a Markov Chain with a single absorbing state. Markov Chains with a single absorbing state have a unique stationary distribution, such that some bounds for the mixing times of Markov Chains can be applied.

Theorem 7.6 There exist opposed configurations such that MAX-GTM needs at least $\Omega\big(n^2 \cdot \log(1/\varepsilon)\big)$ rounds to achieve an ε-approximation of the max-chain.

For the proof consider the protocol matrix $S(t)$ of MAX-GTM as described in Section 7.3.1. Recapitulate that the transition function for the outer robots simplifies as described at the beginning

of this section for one dimension. Therefore, the respective entries in $S(t)$ collapse for the entries $w_1(t)$ and $w_{n-1}(t)$. To achieve the addition of a constant for these vectors in each executed step, we change both $w(t)$ and $S(t)$. $w(t)$ is extended to $w'(t) = \begin{bmatrix} 1 & w_1(t) & \dots & w_{n-1}(t) \end{bmatrix}^T$. $S(t)$ changes to the following matrix, independent of the current time t:

$$A_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & \frac{1}{2} & 0 \end{bmatrix}.$$

Similar to before the behavior of MAX-GTM in one step is described by $w'(t+1) = A_1 \cdot w'(t)$. Furthermore, A_1 is stochastic and can also be interpreted as a Markov Chain with n states, where the first state is absorbing. Similar to [88], the following eigenvalues can be derived:

Lemma 7.3 A_1 has the eigenvalues $\lambda_{j+1} = \cos\left(\frac{j\pi}{n}\right)$ for $j = 0, \dots, n-1$ and the unique stationary distribution $\pi(A_1) = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}$.

Next, we introduce the *mixing time* of a Markov Chain formally. For two probability distributions α and β, $\|\alpha - \beta\|_{TV} := \frac{1}{2}\sum_{i=0}^{n-1} |\alpha_i - \beta_i|$ denotes the *total variation distance* of α and β. Each row of a stochastic matrix (such as A_1) can be interpreted as a probability distribution. We denote by $P_{i,.}$ the i-th row of a matrix P. Given that a transition matrix P of a Markov Chain that converges to a unique stationary distribution π, the *distance from stationarity* after t steps is defined as $d(t) := \max_{0 \le i \le n-1} \|P^t_{i,.} - \pi\|_{TV}$. Then, the *mixing time* is $t_{mix}(\varepsilon) := \min\{t \in \mathbb{N}_0 \,|\, d(t) \le \varepsilon\}$.

The following lemma relates the eigenvalues of the transition matrix of a reversible, irreducible, and aperiodic Markov Chain to its mixing time.

Lemma 7.4 — [81, 94]. Let P be the transition matrix of a reversible, irreducible, and aperiodic Markov Chain over a state space of size n. Furthermore, let $\pi_{min}(P)$ be the smallest entry of its stationary distribution $\pi(P)$ and λ_2 the second largest absolute eigenvalue of P. Then, we obtain $\left(\frac{1}{1-\lambda_2} - 1\right) \cdot \frac{1}{\log(2\varepsilon)} \le t_{mix}(\varepsilon) \le \left(\frac{1}{1-\lambda_2} - 1\right) \cdot \ln\left(\frac{1}{\varepsilon \cdot \pi_{min}(P)}\right)$. The lower bound even holds in case the Markov Chain is not irreducible and aperiodic; it must only converge to a unique stationary distribution.

Proof of Theorem 7.6. Observe that by Lemma 7.3, the eigenvalues of A_1 are equivalent to the ones used in [88], since we have $k = 1$ and consider configurations given by n vectors in contrast to $n+1$ vectors assumed in [88]. Due to [88, Theorem 5], for the spectral gap of A_1 we conclude $\lambda_2 \in \Theta\left(1 - \frac{1}{n^2}\right)$. Combined with Lemma 7.4, we can see that the mixing time for a factor of 2ε is at least $t_{\text{mix}}(2\varepsilon) \ge \left(\frac{1}{1-\lambda_2} - 1\right) \cdot \frac{1}{\log(4\varepsilon)} \in \Omega\left(n^2 \log\left(1/\varepsilon\right)\right)$. Assume we are at time step $t < t_{\text{mix}}(2\varepsilon)$. Consider A_1^t and note that we can express the configuration at step t by $w(t) = A_1^t \cdot w(0)$. Due to the mixing time and the stationary distribution of A_1 introduced in Lemma 7.3, we know that there is an $i \in [0, n-1]$ such that $A_1^t[i,1] < (1-2\varepsilon)$. Further, $A_1^t[i,n-1] \le 1$. Define the initial configuration by $w_1(0) = 1, w_i(0) = -0.313$ for $2 \le i \le n$, and $w_{n-1}(0) = \varepsilon$. Notice that $w(0)$ is

a valid opposed configuration. Also $w_i(t+1) = \sum_{j=0}^{n-1} A_1^t[i,j] \cdot w_j(0) < (1-2\varepsilon)+\varepsilon = (1-\varepsilon)$ and thus, no ε-approximation of the optimal configuration has been reached. ■

7.3.2.2 Analysis of Marching Configurations

Marching configurations do not converge to a max-chain but have a different convergence behavior. They converge to the *marching chain*. It is called marching chain because the robots all together move in the same direction and never stop. For even n, the configuration $w_M(t)$ defines the marching chain. $w_M(t) = (1-\frac{2}{n}, 1-\frac{4}{n}, \ldots, \frac{2}{n}, 0, -\frac{2}{n}, -\frac{4}{n}, \ldots, -(1-\frac{2}{n}))^T$. For odd n, the configuration is similar: $w_M(t) = (1-\frac{2}{n}, 1-\frac{4}{n}, \ldots, \frac{1}{n}, -\frac{1}{n}, -\frac{3}{n}, \ldots, -(1-\frac{2}{n}))^T$. Figure 7.2 visualizes the marching chain for an even number of robots. Observe that $S(t) \cdot w_M(t) = w_M(t)$ ($w_M(t)$ is an eigenvector of $S(t)$ to the eigenvalue 1). In the marching chain, each robot moves a distance of $\frac{1}{n}$ per round.

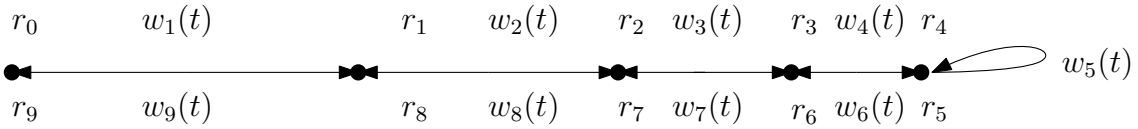

Figure 7.2: A marching chain for $n = 10$. Every position is occupied by two robots.

Starting in a marching configuration, the convergence time until all vectors only differ up to ε from their corresponding vector in the marching chain is equal to the runtime bound for opposed configurations. Here, we can again use the analysis of the mixing time of a Markov Chain for a slightly different transition matrix.

Theorem 7.7 Given a marching configuration, MAX-GTM needs at most $\mathcal{O}(n^2 \cdot \log(n/\varepsilon))$ and at least $\Omega(n^2 \cdot \log(1/\varepsilon))$ rounds to achieve an ε-approximation of the marching chain.

For the proof of Theorem 7.7, we analyze the distance a robot moves in each time step. Define $z_i(t) := p_i(t+1) - p_i(t)$ to be the vector pointing from $p_i(t)$ to $p_i(t+1)$. Without loss of generality, we assume that the outer robots both move in a positive direction. More formally:

$$\begin{aligned} z_0(t) &= \frac{1}{2}p_0(t) + \frac{1}{2}p_1(t) + \frac{1}{2} - p_0(t) \\ z_i(t) &= \frac{1}{2}p_{i-1}(t) + \frac{1}{2}p_{i+1}(t) - p_i(t) && \text{for } 1 \le i < n-1 \\ z_{n-1}(t) &= \frac{1}{2}p_{n-2}(t) + \frac{1}{2}p_{n-1}(t) + \frac{1}{2} - p_{n-1}(t) \end{aligned}$$

Note that for an arbitrary $t \in \mathbb{N}_0$, $\sum_{i=0}^{n-1} z_i(t) = 1$. The corresponding equations for the next time step can be computed as follows:

$$\begin{aligned} z_0(t+1) &= \frac{1}{2}z_0(t) + \frac{1}{2}z_1(t) \\ z_i(t+1) &= \frac{1}{2}z_{i-1}(t) + \frac{1}{2}z_{i+1}(t) && \text{for } 1 \le i < n-1 \\ z_{n-1}(t+1) &= \frac{1}{2}z_{n-2}(t) + \frac{1}{2}z_{n-1}(t) \end{aligned}$$

Now, let $z(t)$ be a column vector of length n whose i-th entry contains $z_i(t)$. For the next time step, $z(t+1)$ can be computed as the product of the transition matrix A_2 and $z(t)$:

$$z(t+1) = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & \dots & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & \dots & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & \dots & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \cdot z(t) = A_2^t \cdot z(1)$$

Note that A_2 can also be interpreted as the transition matrix of an aperiodic and irreducible Markov Chain. The mixing time of this matrix has already been analyzed in [88].

Lemma 7.5 — [88]. For A_2, $t_{mix}(\varepsilon) \in \Omega\left(n^2 \log 1/\varepsilon\right)$ and $t_{mix}(\varepsilon) \in \mathcal{O}\left(n^2 \log n/\varepsilon\right)$.

Based on the results in [88], Theorem 7.7 follows.

7.3.3 Two-Dimensional Analysis

The analysis of two-dimensional configurations exhibits a much more complex behavior than the one-dimensional case. Still, we can prove convergence: the chain either converges to a marching chain or the max-chain and thus, no additional convergence behavior exists when switching to two dimensions (Section 7.3.3.1). However, it is much harder to predict the final configuration based on the properties of the initial configuration. In Section 7.3.3.2, we prove based on results about non-linear discrete dynamical systems that marching chains are unstable fixed points. Intuitively, this means that configurations close to marching chains converge to max-chains. This gives a hint that the set of initial configurations converging to the marching chain is tiny. Using this insight about the stability of marching chains, we derive a lower bound configuration for the two-dimensional case in Section 7.3.3.3. The runtime of MAX-GTM started in this lower bound configuration does not only depend on the number of robots but also a geometric property of the initial configuration. Hence, we see an interesting gap in the runtime between one-dimensional and two-dimensional configurations. We conclude this section by considering the case that only one of the outer robots moves while the other one remains stationary in Section 7.3.3.4. We can prove the same runtime bounds as for the one-dimensional case for this case.

7.3.3.1 Convergence Result

Next, we prove a convergence result for two-dimensional configurations, stating that an initial configuration either converges to the max-chain or the marching chain. In the analysis, we again consider the vectors $z_i(t) = p_i(t+1) - p_i(t)$ and the function $\phi_2(t) = \sum_{i=0}^{n-1} \|z_i(t)\|_2^2$. We can bound $\phi_2(t) - \phi_2(t+1)$ as follows:

Lemma 7.6 $\phi_2(t) - \phi_2(t+1) \geq \frac{1}{4} \sum_{i=0}^{n-1} \|z_{i-1}(t) - z_{i+1}(t)\|_2^2$

Proof.

$$\begin{aligned} \phi_2(t+1) &= \sum_{i=0}^{n-1} \| \frac{p_{i-1}(t+1) + p_{i+1}(t+1)}{2} - p_i(t+1) \|_2^2 \\ &= \frac{1}{4} \sum_{i=1}^{n-2} \|z_{i-1}(t) + z_{i+1}(t)\|_2^2 + \|z_0(t+1)\|_2^2 + \|z_{n-1}(t+1)\|_2^2 \end{aligned}$$

With $\|z_0(t+1)\|_2^2 \leq \frac{1}{4}\|z_0(t)+z_1(t)\|_2^2$ and $\|z_{n-1}(t+1)\|_2^2 \leq \frac{1}{4}\|z_{n-2}(t)+z_{n-1}(t)\|_2^2$ it follows that $\phi_2(t+1) \leq \frac{1}{4}\sum_{i=0}^{n-1}\|z_{i-1}(t)+z_{i+1}(t)\|_2^2$. Now, define $\Delta\phi_2(t)$ to be $\phi_2(t)-\phi_2(t+1)$. With help of the parallelogram law, we derive a lower bound on $\Delta\phi_2(t)$:

$$\begin{aligned}\Delta\phi_2(t) &\geq \sum_{i=0}^{n-1}\|z_i(t)\|_2^2 - \frac{1}{4}\sum_{i=0}^{n-1}\|z_{i-1}(t)+z_{i+1}(t)\|_2^2 \\ &= \sum_{i=0}^{n-1}\|z_i(t)\|_2^2 - \sum_{i=0}^{n-1}\left(\|z_i(t)\|_2^2 - \frac{1}{4}\|z_{i-1}(t)-z_{i+1}(t)\|_2^2\right) \\ &= \frac{1}{4}\sum_{i=0}^{n-1}\|z_{i-1}(t)-z_{i+1}(t)\|_2^2\end{aligned}$$

■

Theorem 7.8 Given an arbitrarily connected chain in the Euclidean plane, MAX-GTM converges either to the marching chain or to the max-chain.

Proof. $\phi_2(t)$ is a monotonically decreasing function of t, bounded from below by 0, and the potential difference can be lower bounded by $\frac{1}{4}\sum_{i=0}^{n-1}\|z_{i-1}(t)-z_{i+1}(t)\|_2^2$ (Lemma 7.6). Hence, the potential difference can only be 0 in case all vectors $z_i(t)$ are equal. Either $z_0(t)=z_1(t)=\cdots=z_{n-1}(t)>0$ or $z_0(t)=\cdots=z_{n-1}(t)=0$. In the first case, all robots move the same distance in the same direction (a marching chain). In the second case, no robot moves at all, which can only be the case if the chain is stretched to a max-chain. ■

7.3.3.2 Stability Result

This section aims to prove that the marching chain is an unstable fixed point of MAX-GTM, interpreted as a discrete (non-linear) dynamical system. We split the vector representation of a configuration into its x- and y-components. More precisely, consider the vector $w_i(t) := (x_i(t), y_i(t))$. Now, define the state of a system to be $s(t) := (x_1(t),\ldots,x_{n-1}(t),y_1(t),\ldots,y_{n-1}(t))$. The dynamical system consists of $2(n-1)$ variables each representing an entry of the vector representation. Applying MAX-GTM to the configuration can now be interpreted as a set of $2(n-1)$ functions, one function for each variable.

$$\begin{aligned}
x_1(t+1) &= f_{x_1}(s(t)) = \frac{x_1(t)}{2\cdot\sqrt{x_1(t)^2+y_1(t)^2}} + \frac{1}{2}x_2(t) \\
x_i(t+1) &= f_{x_i}(s(t)) = \frac{1}{2}x_{i-1}(t) + \frac{1}{2}x_{i+1}(t) && \text{for } 1<i<n-1 \\
x_{n-1}(t+1) &= f_{x_{n-1}}(s(t)) = \frac{1}{2}x_{n-2}(t) + \frac{x_{n-1}(t)}{2\cdot\sqrt{x_{n-1}(t)^2+y_{n-1}(t)^2}} \\
y_1(t+1) &= f_{y_1}(s(t)) = \frac{y_1(t)}{2\cdot\sqrt{x_1(t)^2+y_1(t)^2}} + \frac{1}{2}y_2(t) \\
y_i(t+1) &= f_{y_i}(s(t)) = \frac{1}{2}y_{i-1}(t) + \frac{1}{2}y_{i+1}(t) && \text{for } 1<i<n-1 \\
y_{n-1}(t+1) &= f_{y_{n-1}}(s(t)) = \frac{1}{2}y_{n-2}(t) + \frac{y_{n-1}(t)}{2\cdot\sqrt{x_{n-1}(t)^2+y_{n-1}(t)^2}}
\end{aligned}$$

Next, we compute the *Jacobian matrix* $\mathcal{J}$ of the dynamical system. For this dynamical system, $\mathcal{J}$ is a $2\cdot(n-1)\times 2\cdot(n-1)$-matrix. Each row corresponds to one of the $2\cdot(n-1)$ transition functions. An entry $\mathcal{J}_{i,j}$ represents $\frac{\partial f_i}{\partial j}$, the derivative of f_i with respect to variable j. Each function depends on at most 3 variables, and thus each row contains at most 3 non-zero elements. For better readability, we omit the time parameter t and define $\eta_i(t) = 2\cdot(x_i^2+y_i^2)^{3/2}$.

$$\mathcal{J} = \begin{bmatrix} \frac{y_1^2}{\eta_1} & \frac{1}{2} & 0 & \dots & 0 & 0 & \frac{-x_1 \cdot y_1}{\eta_1} & 0 & \dots & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & \dots & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \frac{1}{2} & \frac{y_{n-1}^2}{\eta_{n-1}} & 0 & 0 & \dots & 0 & \frac{-x_{n-1} \cdot y_{n-1}}{\eta_1} \\ \frac{-x_1 \cdot y_1}{\eta_{n-1}} & 0 & 0 & \dots & 0 & 0 & \frac{x_1^2}{2\eta_1} & \frac{1}{2} & \dots & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & \frac{1}{2} & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & \dots & 0 & \frac{1}{2} \\ 0 & 0 & \dots & 0 & 0 & \frac{-x_{n-1} \cdot y_{n-1}}{\eta_{n-1}} & 0 & 0 & \dots & \frac{1}{2} & \frac{x_{n-1}^2}{\eta_{n-1}} \end{bmatrix} \tag{7.6}$$

To prove that the marching chain is an unstable fixed point, we have to evaluate $\mathcal{J}$ at that fixed point. Recall the marching chain is one-dimensional. Thus we assume that $x_i(t) = 0$ for $1 \leq i \leq n-1$. The variables $y_i(t)$ are defined according to the marching chain: $y_1(t) = 1 - \frac{2}{n}$, $y_2(t) = 1 - \frac{4}{n}, \dots, y_{n-1}(t) = -\left(1 - \frac{2}{n}\right)$. Plugging these values into the Jacobian matrix (Equation (7.6)) yields the following matrix:

$$\mathcal{J}_{w_M} = \begin{bmatrix} \frac{n}{2(n-2)} & \frac{1}{2} & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & \dots & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \frac{1}{2} & \frac{n}{2(n-2)} & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & \frac{1}{2} & \dots & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & \frac{1}{2} & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & \dots & 0 & \frac{1}{2} \\ 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & \dots & \frac{1}{2} & 0 \end{bmatrix} \tag{7.7}$$

The stability of the marching chain can now be analyzed with the help of the eigendecomposition of $\mathcal{J}_{w_M}$. More precisely, in the case that at least one eigenvalue of $\mathcal{J}_{w_M}$ has a magnitude larger than 1, it follows that the marching chain is an unstable fixed point (see for instance [71]). The following lemma helps us to prove a lower bound on the largest eigenvalue of $\mathcal{J}_{w_M}$.

Lemma 7.7 — [115]. Define $u^T = (1, \dots, 1)$. The largest eigenvalue $\lambda_1(A)$ of a symmetric $n \times n$ matrix A can be lower bounded by $\lambda_1(A) \geq \frac{u^T \cdot A \cdot u}{u^T \cdot u}$.

Theorem 7.9 The marching chain is an unstable fixed point.

Proof. As $\mathcal{J}_{w_M}$ is a $2 \cdot (n-1) \times 2 \cdot (n-1)$ matrix, it follows $u^T \cdot u = 2 \cdot (n-1)$. Next, observe that every column of $\mathcal{J}_{w_M}$, except for the first and the lst one, are stochastic. The first and the last column

sum up to $\frac{1}{2}+\frac{n}{2\cdot(n-2)}=1+\frac{1}{n-2}$. Thus, we can compute $u^T\cdot A=\left(1+\frac{1}{n-2},1,\ldots,1,1+\frac{1}{n-2},1,\ldots,1\right)$ and $u^T\cdot A\cdot u=2\cdot(n-1)+\frac{2}{n-2}$. Finally, we can bound $\lambda_1(\mathcal{J}_{w_M})$ via Lemma 7.7 and prove the theorem:

$$\lambda_1(\mathcal{J}_{w_M})\geq\frac{2\cdot(n-1)+\frac{2}{n-2}}{2\cdot(n-1)}=1-\frac{1}{n-1}+\frac{1}{n-2}>1\text{ (given }n>2).$$

■

7.3.3.3 Derivation of a Lower Bound

In case both outer robots move, we identify a certain class of configurations that lead to an arbitrarily high runtime based on a parameter δ, which can be seen as the width of the configuration. Before defining the configurations, we give some intuition about their construction: As the marching chain is an unstable fixed point (see Section 7.3.3.2), we know that configurations that are close to a marching chain converge to the max-chain. Additionally, the largest eigenvalue of the Jacobian matrix evaluated at that fixed point is very close to one (in the order of $1+\frac{1}{n}$), and thus, the dynamics close to the marching chain are very slow. Therefore, we investigate configurations that are close to marching chains, we denote them as *discrete δ-V-configurations*. See also Figure 7.3 for a visualization.

Definition 7.1 For odd n, a *discrete δ-V-configuration* is defined by the vectors

$$w_i(t):=\left(\frac{\delta}{n-1},1-\frac{2\cdot i}{n}\right)^T\text{ for }i=1,\ldots,n-1.$$

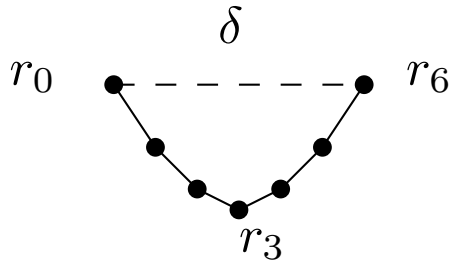

Figure 7.3: A discrete δ-V-configuration.

Observe that for $\delta=0$, discrete δ-V-configurations and marching chains coincide. Choosing *any* $\delta>0$ changes the behavior such that the configuration converges to the max-chain. The runtime, however, can be arbitrarily high depending on δ.

Theorem 7.10 Starting in a discrete δ-V-configuration, MAX-GTM needs at least $\Omega\left(n^2\cdot\log(1/\delta)\right)$ rounds to achieve an ε-approximation of the max-chain.

For the proof of Theorem 7.10, we need two auxiliary lemmata.

Lemma 7.8 During an execution of MAX-GTM starting in a discrete δ-V-configuration at time step 0, $\|w_i(t)\|_2\geq\|w_i(0)\|_2$ for all t and all $1\leq i\leq n-1$.

Proof. Observe that for all $w_i(t)$ it always holds $x_i(t)\geq 0$ and for $0\leq i\leq\frac{n}{2}-1: y_i(t)\leq 0$ and for $\frac{n}{2}-1\leq i\leq n-1: y_i(t)\geq 0$. We prove the fact for vectors $w_{n/2}$ to w_{n-1} by induction over t. The induction for the first half of vectors is analogous. The induction base is clear. Consider time step $t+1$ and an vector $w_i(t+1)=\frac{1}{2}w_{i-1}(t)+\frac{1}{2}w_{i+1}(t)$. Observe that $\|w_i(t+1)\|_2=\frac{1}{2}\|w_{i-1}(t)+w_{i+1}(t)\|_2$. Since $x_{i-1}(t),x_{i+1}(t),y_{i-1}(t)$ and $y_{i+1}(t)\geq 0$, $\|w_i(t+1)\|_2\geq\frac{1}{2}\|w_{i-1}(0)+w_{i+1}(0)\|_2=\|w_i(0)\|_2$. ■

Lemma 7.9 When applying MAX-GTM to a discrete δ-V-configuration, we have $x_1(t) \geq \frac{1}{2}$ after $\Omega\left(n^2 \cdot \log\left(1/\delta\right)\right)$ rounds.

Proof. Trivially, $x_2(t) \leq x_1(t)$ for all t. Thus, we can bound $x_1(t+1)$ as follows.

$$\begin{aligned} x_1(t+1) &\leq \frac{1}{2-\frac{4}{n}} x_1(t) + \frac{1}{2} x_2(t) \\ &= \frac{1}{2} x_1(t) + \frac{1}{n-2} x_1(t) + \frac{1}{2} x_2(t) \\ &\leq \left(1 + \frac{1}{n-2}\right) x_1(t) \end{aligned}$$

Thus, $x_1(t)$ doubles at most every $\mathcal{O}(n)$ rounds. Since $x_1(0) = \frac{\delta}{n-1}$ it requires at least $\mathcal{O}(n^2)$ rounds until $x_1(t) \geq 2\delta$. The lemma follows. ■

Lemma 7.9 together with the lower bound for one-dimensional configurations imply Theorem 7.10. Hence, the runtime of MAX-GTM can be arbitrarily high depending on δ. The dependence on δ can be removed in the continuous time model by an (at first sight) counter-intuitive approach: The outer robots move slower than the inner robots (Section 7.4). The same approach does not work in $\mathcal{F}$SYNC (see Section 7.5).

7.3.3.4 Upper bound for one Stationary Outer Robot

Interestingly, by assuming that only one of the outer robots moves while the other remains stationary, we can prove the same upper runtime bound as for one-dimensional configurations. Note that no marching chain can exist in this case as one outer robot does not move. The proof relies on the analysis of $\phi_2(t)$ for this case. Again, the analysis of a transition matrix plays a role here – since only one outer robot moves, we obtain a *substochastic* transition matrix where every row except one sums up to 1. The last row only sums up to $1/2$ such that high powers of this matrix converge to the 0-matrix. Diagonalization of the transition matrix yields the following runtime bound.

Theorem 7.11 In case one of the outer robots is stationary and all other robots move according to MAX-GTM, an ε-approximation of the max-chain is achieved after $\mathcal{O}(n^2 \cdot \log(n/\varepsilon))$ rounds.

For the proof, we assume without loss of generality that r_0 does not move and thus $p_0(0) = p_0(t)$ for all t. We again use a potential function $\phi_2(t)$, which sums up the squared distances of robots to their target points. This time, $z_0(t) = 0$ for all t, and therefore, we exclude $z_0(t)$ from the summation and obtain $\phi_2(t) = 4\sum_{i=1}^{n-1} \|z_i(t)\|_2^2$. The new equation for $z_1(t+1)$ simplifies as follows (all other equations remain unchanged):

$$\begin{aligned} \|z_1(t+1)\|_2^2 &= \|\frac{1}{2} p_0(t+1) + \frac{1}{2} p_2(t+1) - p_1(t+1)\|_2^2 \\ &= \|\frac{1}{2} p_0(t) + \frac{1}{4} p_1(t) + \frac{1}{4} p_3(t) - \frac{1}{2} p_0(t) - \frac{1}{2} p_2(t)\|_2^2 \\ &= \frac{1}{4} \|z_2(t)\|_2^2 \leq \frac{1}{2} \|z_2(t)\|_2^2. \end{aligned}$$

Define $z'(t) = \left(\|z_1(t)\|_2^2, \ldots, \|z_{n-1}(t)\|_2^2\right)^T$. Given two n-dimensional column vectors v and v' we define $v \leq v'$ if $v_i \leq v'_i$ for all $1 \leq i \leq n-1$. Then, we can upper bound $z'(t+1)$ as follows.

$$z'(t+1) \leq \begin{bmatrix} 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & \dots & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & \dots & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & \dots & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \cdot z'(t) = A_3 \cdot z'(t) = A_3^{t+1} \cdot z'(0)$$

Observe that A_3 is a substochastic matrix in which every row except for the first one is stochastic. To analyze the convergence time, we determine the largest eigenvalue and the eigenvectors of A_3.

Lemma 7.10 The eigenvalues of A_3 are $\lambda_j = \cos\left(\frac{(2j-1)\cdot\pi}{2n-1}\right)$ for $j = 1,\dots,n-1$. The corresponding eigenvectors are given by $x_j[i] = \cos\left(\frac{(2j-1)\cdot(2i-1)}{2\cdot(2n-1)}\right)$ for $i, j = 1,\dots,n-1$ where $x_j[i]$ denotes the i-th entry of eigenvector j.

Proof. The matrix A_3 is a special tridiagonal matrix whose eigenvalues and eigenvectors have been analyzed in [121]. A more general result about these matrices can be found in [99]. The matrices in [121] are defined as

$$T(a,b,c,\alpha,\beta) = \begin{bmatrix} -\alpha+b & c & 0 & 0 & 0 & 0 & \dots & 0 \\ a & b & c & 0 & 0 & 0 & \dots & 0 \\ 0 & a & b & c & 0 & 0 & \dots & 0 \\ 0 & 0 & a & b & c & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & a & -\beta+b \end{bmatrix}.$$

In our case, $a = c = \frac{1}{2}, b = \alpha = 0, \beta = -\sqrt{ac} = -\frac{1}{2}$. For these values of a, b, c, α and β, Lemma 7.10 follows from Theorem 2 in [121]. ■

Next, we introduce a technical lemma from [59] that upper bounds the entries of A_3 by its eigenvalues and eigenvectors.

Lemma 7.11 — Lemma 2 in [59]. For any irreducible, symmetric, substochastic $n \times n$ matrix P with pairwise distinct eigenvalues and any i, j we have $P^k[i,j] \leq n \cdot \alpha \cdot \beta^k$ where β is the largest absolute eigenvalue of P and $\alpha = \max_{i,j,i',j'} |x_j[i] \cdot x_{j'}[i']|$ with x_j denoting the j-th eigenvector of P.

Now, we have collected all tools needed to give the proof of Theorem 7.11.

Proof of Theorem 7.11. We apply the results of Lemma 7.11. Due to Lemma 7.10, we have $\alpha \leq 1$ and $\beta = \cos\left(\frac{\pi}{2n-1}\right)$. Using $\cos(x) \leq 1 - \frac{2x^2}{\pi^2}$ for $-\pi \leq x \leq \pi$, we can derive $\beta \leq (1 - \frac{1}{(2n-1)^2})$. For $t = (2n-1)^2$ we obtain $\beta^t \leq \frac{1}{e}$. Hence, for $t' \geq (2n-1)^2 \cdot \ln\left(\frac{4n^4}{\varepsilon^2}\right)$ we have that for all

i,j $A_3^{t'}[i,j] \leq \frac{\varepsilon^2}{4n^3}$. Since initially $\|z_i(0)\|_2^2 \leq 1$ for all $1 \leq i \leq n-1$, we obtain for all $2 \leq i \leq n$: $\|z_i(t')\|_2^2 \leq \frac{\varepsilon^2}{4n^2}$. After t' rounds, no robot moves a distance of larger than $\frac{\varepsilon^2}{4n^2}$ anymore.

We conclude by proving that after t' rounds also an ε-approximation of the max-chain is reached. First of all, we prove $\|w_i(t)\|_2 \geq (1-\varepsilon)$ for $1 \leq i \leq n-1$. Consider $z_{n-1}(t')$. Without loss of generality, we assume that $p_{n-1}(t') = (0,0)$ and $p_n(t') = (1,0)$ and $p_{n-2}(t') = (x_{n-2},0)$. Since $\|z_{n-1}(t')\|_2^2 \leq \frac{\varepsilon^2}{4n^2}$, we obtain $\|z_{n-1}(t')\|_2 \leq \frac{\varepsilon}{2n}$. Hence, $-\frac{\varepsilon}{2n} \leq \frac{1}{2}x_{n-2} + \frac{1}{2} \leq \frac{\varepsilon}{2n}$, implying $-\left(1+\frac{\varepsilon}{n}\right) \leq x_{n-2} \leq -\left(1-\frac{\varepsilon}{n}\right)$. As $0 \leq \varepsilon$, we can simplify to $-1 \leq x_{n-2} \leq -\left(1-\frac{\varepsilon}{n}\right)$. Now, observe $w_{n-1}(t') = (-x_{n-2}, -y_{n-2})$ and thus $\|w_{n-1}(t')\|_2 \geq \left(1-\frac{\varepsilon}{n}\right)$. With similar arguments, we can conclude $\|w_{n-1-j}(t')\|_2 \geq (1-(j+1)\cdot\frac{\varepsilon}{n})$ and thus $\|w_1(t')\|_2 \geq (1-(n-1)\cdot\frac{\varepsilon}{n}) \geq (1-\varepsilon)$. Hence, all vectors have a length of at least $(1-\varepsilon)$. In the same style, we can prove $\Delta_{0,n-1}(t') \geq (1-\varepsilon)\cdot(n-1)$.

■

7.4 Protocols and Analyses in the $\mathcal{OBLOT}_1^C$ Model

This section is dedicated to the MAX-MOB protocol that transforms a connected chain into a max-chain in the continuous time model. After introducing the protocol, we continue with some preliminaries in Section 7.4.1. The following Section 7.4.2 deals with a more detailed explanation of the velocity vectors of outer robots based on the Zenoness phenomenon, which is part of the continuous time model (see Section 2.1.3). Afterward, we provide an intuitive explanation of the protocol combined with a proof outline in Section 7.4.3. The complete analysis can be found in Section 7.4.4.

MAX-MOVE-ON-BISECTOR (MAX-MOB). In the following, we omit the protocol name in the velocity vectors, i.e., we only write $v_i(t)$ to describe the velocity vector of r_i at time t. Outer robots move with a maximal speed of $(1-\tau)$ for a constant $0 < \tau \leq 1/2$ as follows: In case $\|w_1(t)\|_2 < 1$: $v_0(t) = -(1-\tau)\cdot\widehat{w}_1(t)$. Similarly, in case $\|w_{n-1}(t)\|_2 < 1$, $v_{n-1}(t) = (1-\tau)\cdot\widehat{w}_{n-1}(t)$. In other words, outer robots move with a speed of $(1-\tau)$ away from their direct neighbors. Otherwise, provided $\|w_1(t)\|_2 = 1$ ($\|w_{n-1}(t)\|_2 = 1$ respectively), an outer robot adjusts its speed and tries to stay in distance 1 to its neighbor while moving with a maximal speed of $1-\tau$. An inner robot r_i with $0 < \alpha_i(t) < \pi$ moves only if at least one of the following three conditions holds: $\|w_i(t)\|_2 = 1$, $\|w_{i+1}(t)\|_2 = 1$ or $\alpha_i(t) < \psi$ for $\psi := 2\cdot\cos^{-1}(1-\tau)$. Otherwise, an inner robot does not move at all. If one of the conditions holds, an inner robot moves with speed 1 along the angle bisector formed by the vectors pointing to its neighbors. As soon as the position of the robot and the positions of its neighbors are collinear, it continues moving with a speed of 1 towards the midpoint between its neighbors while ensuring staying collinear. Once it has reached the midpoint, it adjusts its speed to stay on the midpoint. See Figure 7.4 for a visualization.

7.4.1 Preliminaries

For both outer robots, we determine the index of the first robot that is not collinear with its neighbors and the respective outer robot.

Definition 7.2 Define $\ell(t)$ to be the index, such that for all $1 < j \leq \ell(t)$ either $w_j(t) = (0,0)$ or $\widehat{w}_j(t) = \widehat{w}_1(t), w_{\ell(t)+1}(t) \neq (0,0)$ and $\widehat{w}_{\ell(t)+1}(t) \neq \widehat{w}_1(t)$. Similarly, define $r(t)$ to be the index such that for all $r(t) < j < n-1$ either $w_j(t) = (0,0)$ or $\widehat{w}_j(t) = \widehat{w}_{n-1}(t)$ and $w_{r(t)}(t) \neq (0,0)$ and $\widehat{w}_{r(t)}(t) \neq \widehat{w}_{n-1}(t)$. In case there is no such an index define $\ell(t) = r(t) = 0$. $\alpha_{\ell(t)}(t)$ and $\alpha_{r(t)}(t)$ are denoted as *outer angles*.

We omit the time parameter t in indices, when it is clear from the context, e.g., we write $\alpha_\ell(t)$ instead of $\alpha_{\ell(t)}(t)$. In addition to the indices $\ell(t)$ and $r(t)$, we define the last indices of robots (starting to count at $\ell(t)$ and $r(t)$) that are collinear with their neighbors and the corresponding robot with index $\ell(t)$ or $r(t)$.

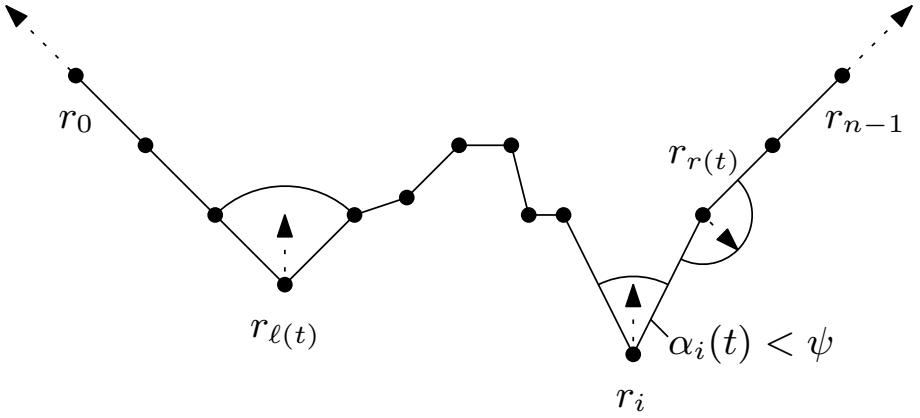

Figure 7.4: A chain visualizing the movements of MAX-MOB. The velocity vectors are depicted by dashed arrows. Both $r_\ell(t)$ and $r_{r(t)}$ move since a neighboring vector has a length of 1. Additionally, r_i moves because $\alpha_i(t) < \psi$. The robots between $r_{\ell(t)}$ and r_i do not move because their neighboring vectors have a length smaller than 1.

Definition 7.3 Let $\ell^+(t)$ be the smallest index larger than $\ell(t)$ such that $\alpha_{\ell^+}(t) < \pi$. Similarly let $r^+(t)$ be the largest index less than $r(t)$ such that $\alpha_{r^+}(t) < \pi$.

Definition 7.4 The *left outer length* is defined as $O_\ell(t) := \sum_{i=1}^{\ell(t)} \|w_i(t)\|_2$ and the *right outer length* as $O_r(t) := \sum_{i=r(t)+1}^{n-1} \|w_i(t)\|_2$. The maximal values of the left and right outer length are denoted by $\gamma_\ell(t) := \ell(t)$ and $\gamma_r(t) := n-1-r(t)$. Additionally, the *inner length* is defined as $I(t) := \sum_{i=\ell(t)+1}^{r(t)} \|w_i(t)\|_2$.

The following Figure 7.5 depicts Definitions 7.2 to 7.4.

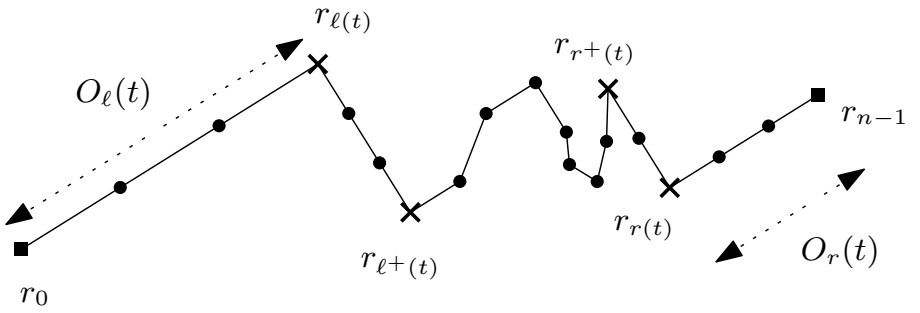

Figure 7.5: A visualization of $\ell(t), \ell^+(t), r(t), r^+(t), O_\ell(t)$ and $O_r(t)$.

We conclude this section with a helpful lemma stating how the distance between two robots changes, depending on their trajectories. We have seen this lemma with a slightly different notation in Lemma 4.2. Here, we restate it according to the notation used in this chapter. For this purpose, we define the angles

$$\beta_{i,j}(t) := \angle\big(v_i(t), p_j(t) - p_i(t)\big).$$

In other words, $\beta_{i,j}(t)$ denotes the signed angle between the velocity vector of r_i and the line segment connecting r_i and r_j at time t.

Lemma 7.12 — Lemma 3.1 in [49]. Consider two robots r_i and r_j and let $\Delta_{i,j}(t) : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ represent their distance at time t. The distance between r_i and r_j changes with speed

$$\Delta_{i,j}{}'(t) = -(\|v_i(t)\|_2 \cdot \cos(\beta_{i,j}(t)) + \|v_j(t)\|_2 \cdot \cos(\beta_{j,i}(t))).$$

7.4.2 A Note on The Velocity Vectors of Outer Robots

While the velocity vectors of inner robots might be clear from the description, the velocity vectors of outer robots are special in the sense that they might not only depend on the length of the neighboring vector. To clarify this behavior, we formally analyze and state the velocity vectors $v_0(t)$ and $v_{n-1}(t)$. As we are in a continuous time model, the velocity vectors might depend on the positions and movements of the neighbors of a robot and on robots that are at a farther distance. More precisely, if there is a subchain in which multiple robots are already located on a straight line, the velocity vectors of these robots are influenced by the first and last robot of that subchain. This phenomenon is called *Zenoness* [72], see also Section 2.1.3.

There are two possible velocity vectors for outer robots which only differ in their length. In both cases, the direction of the velocity vector is the same as the direction of the vector pointing from their neighbors to themselves. The length of the velocity vector only depends on $O_i(t)$. We now derive the velocity vector $v_0(t)$, and the vector $v_{n-1}(t)$ can be derived analogously. There are two cases to consider, $O_\ell(t) < \gamma_\ell(t)$ and $O_\ell(t) = \gamma_\ell(t)$. The case $O_\ell(t) < \gamma_\ell(t)$ consists of two sub-cases in which r_0 moves with speed $1-\tau$. If the robot detects $\|w_1(t)\|_2 < 1$, it moves with speed $1-\tau$ away from its neighbor. Otherwise, ($\|w_1(t)\|_2 = 1$), let j be the smallest index such that $\|w_j(t)\|_2 < 1$. Since $O_\ell(t) < \gamma_\ell(t)$, we have that $j < \ell(t)$. The robot r_{j-1} moves with speed 1 to the midpoint between its neighbors which lies on the straight line between r_0 and $r_{\ell(t)}$ closer to r_0. Thus, $\|w_1(t)\|_2$ immediately decreases such that r_0 moves with speed $1-\tau$.

In case $O_\ell(t) = \gamma_\ell(t)$, every vector with an index less than $\ell(t)$ has a length of 1 and points in the same direction. $r_{\ell(t)}$ moves along the bisector of vectors pointing to its neighbors and decreases the length of $O_\ell(t)$ with speed $\cos\left(\frac{\alpha_\ell(t)}{2}\right)$, since $\beta_{\ell,1}(t) = \frac{\alpha_\ell(t)}{2}$ (Lemma 7.12). Provided that $\cos\left(\frac{\alpha_\ell(t)}{2}\right) \leq 1-\tau$, r_0 can move fast enough such that $O_\ell(t)$ remains unchanged. Therefore, r_0 moves only with speed $\cos\left(\frac{\alpha_\ell(t)}{2}\right)$. Otherwise, in case $\cos\left(\frac{\alpha_\ell(t)}{2}\right) > 1-\tau$, r_0 moves at its maximum speed of $1-\tau$. The resulting velocity vectors can be summarized as follows:

$$v_0(t) = \begin{cases} -(1-\tau)\cdot \widehat{w}_1(t) & \text{if } O_\ell(t) < \gamma_\ell(t) \\ -\min\{\cos\left(\frac{\alpha_\ell(t)}{2}\right), 1-\tau\}\cdot \widehat{w}_1(t) & \text{if } O_\ell(t) = \gamma_\ell(t) \end{cases}$$

$$v_{n-1}(t) = \begin{cases} (1-\tau)\cdot \widehat{w}_{n-1}(t) & \text{if } O_r(t) < \gamma_r(t) \\ \min\{\cos\left(\frac{\alpha_r(t)}{2}\right), 1-\tau\}\cdot \widehat{w}_{n-1}(t) & \text{if } O_r(t) = \gamma_r(t) \end{cases}$$

Lastly, if every robot is located on a single point or the straight line of length $n-1$ is reached, the outer robots do not move at all such that their velocity vector is the zero vector.

7.4.3 Intuition & Proof Outline

The main idea of MAX-MOB is to flatten and stretch the chain starting at the outer robots and towards the inside of the chain. At first, $\|w_1(t)\|_2 = 1$ and $\|w_{n-1}(t)\|_2 = 1$ is ensured, afterward the angles $\alpha_1(t)$ and $\alpha_{n-2}(t)$ should reach a size of π and so on until finally all vectors have length 1 and all angles have a size of π. Figure 7.6 visualizes this core idea.

To achieve this behavior, one of the two cases in which an inner robot r_i moves demands either $\|w_i(t)\|_2 = 1$ or $\|w_{i+1}(t)\|_2 = 1$ because locally, it can assume that it is already located on the straight line to one outer robot and all vectors into the direction of the outer robot have a length of 1. In addition, an inner robot r_i moves if $\alpha_i(t) < \psi$ ($\psi := 2\cdot\cos^{-1}(1-\tau)$). In Section 7.5 we see that this property is crucial for the linear runtime of the protocol by introducing configurations that have a high runtime that not only depends on the number of robots in case the property is ignored.

To express the behavior of flattening and stretching the chain starting at the outer robots towards the inside of the chain, we have introduced the indices $\ell(t)$ and $r(t)$. For each of the two sets of robots $r_0,\ldots,r_{\ell(t)}$ and $r_{r(t)},\ldots,r_{n-1}$ it always holds that these robots continue to stay collinear

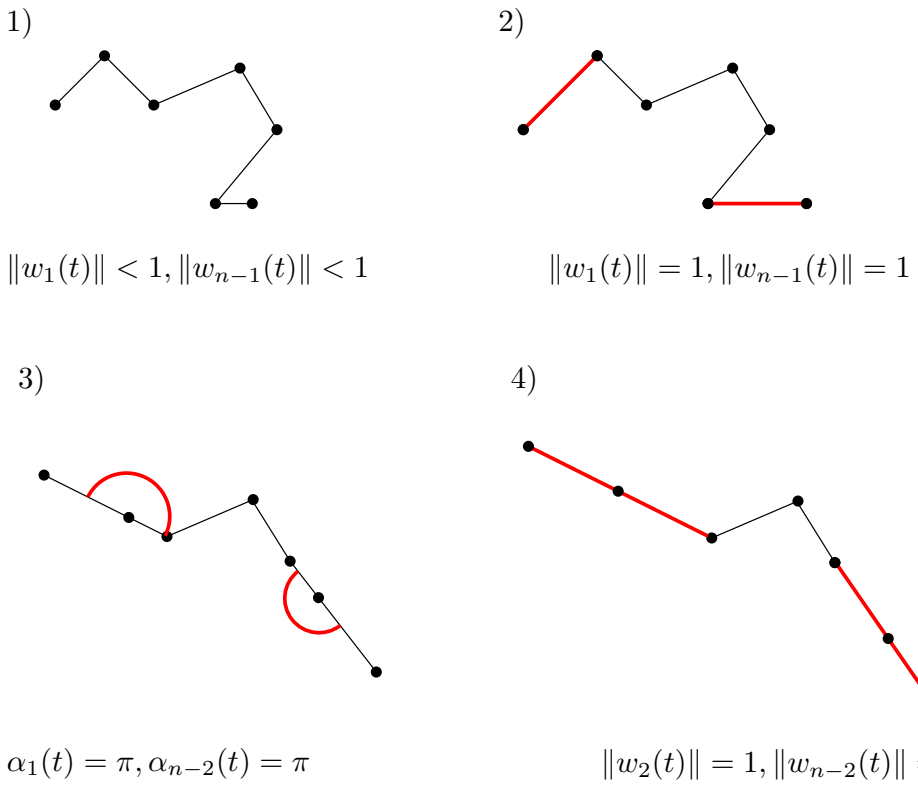

Figure 7.6: A visualization of the core idea of MAX-MOB. 1) depicts an initial configuration. 2) visualizes the configuration after stretching $w_1(t)$ and $w_{n-1}(t)$. In 3), $\alpha_1(t) = \pi$ and $\alpha_{n-2}(t) = \pi$. In 4) r_0 and r_1 as well as r_{n-2} and r_{n-1} have moved such that $\|w_1(t)\|_2 = \|w_2(t)\|_2 = \|w_{n-2}(t)\|_2 = \|w_{n-1}(t)\|_2 = 1$.

for the rest of the execution. Thus, $\ell(t)$ is monotonically increasing and $r(t)$ is monotonically decreasing such that after some time $\ell(t) = r(t)$ (Lemma 7.14). Consequently, at most one angle of size less than π remains in the chain. Assuming $\ell(t) = r(t)$ we prove that after a linear time, the chain is properly stretched to a max-chain or all robots are located on the same point (Lemma 7.23).

To bound the time until $\ell(t) = r(t)$, the outer angles $\alpha_\ell(t)$ and $\alpha_r(t)$ play an important role. We divide the possible sizes into three intervals, $[0, \psi), [\psi, \frac{3}{4}\pi]$ and $(\frac{3}{4}\pi, \pi]$. For an outer angle $\alpha_i(t) \in [0, \psi) (i \in \{\ell(t), r(t)\})$ two properties hold: $I(t)$ decreases with speed at least $1 - \tau$ (Lemma 7.15) and the corresponding outer length decreases since the outer robots move with speed at most $1 - \tau$ (Lemma 7.18). As $I(t)$ decreases with a constant speed of at least $1 - \tau$, the total time in such a case is upper bounded by $\mathcal{O}(n)$ (Corollary 7.1). Given $\alpha_i(t) \in [\psi, \frac{3}{4}\pi]$, the protocol is designed such that r_i only moves if $O_i(t) = \gamma_i(t)$. Thus, as long as $O_i(t) < \gamma_i(t)$, $O_i(t)$ increases with speed $1 - \tau$ (Lemma 7.17). As soon as $O_i(t) = \gamma_i(t)$, the robot r_i starts moving along its bisector. This movement causes a decrease of $I(t)$ with speed at least $\cos\left(\frac{3}{8}\pi\right)$ while the length of $O_i(t)$ does not change (Lemma 7.19). Since $I(t)$ decreases with constant speed, this case can hold for a time at most $\mathcal{O}(n)$ (Corollary 7.2). For the last interval, $\alpha_i(t) \in (\frac{3}{4}, \pi]$ we use a different progress measure since large angles cause a very slight decrease of $I(t)$ which cannot be bounded by a constant anymore. Therefore, we consider the height $H_i(t)$. Assume that $i = \ell(t)$, then $H_\ell(t)$ denotes the distance between $r_{\ell(t)}$ and the line segment connecting r_1 and $r_{\ell^+(t)}$. Intuitively, if we consider the line segment connecting r_1 and $r_{\ell^+(t)}$ as a line parallel to the x-axis, the robot r_ℓ moves with a velocity vector that has a small angle to the y-axis towards this line segment. Thus, $H_i(t)$ decreases with constant speed (Lemma 7.21). All in all, we can prove that the total time of outer angles in any of these intervals is upper bounded by $\mathcal{O}(n)$ such that finally $\ell(t) = r(t)$. Plugging all these results together yields the following theorem.

Theorem 7.12 Starting MAX-MOB in a two-dimensional configuration, the initial chain is either transformed into a straight line of length $n-1$ or all robots are located at the same position after time $\mathcal{O}(n)$.

7.4.4 Detailed Analysis

The basis for the entire analysis is that MAX-MOB maintains the connectivity of the chain as stated by the following lemma.

Lemma 7.13 Given a connected configuration at time t', MAX-MOB ensures that the configuration remains connected for all $t' > t$.

Proof. Consider first of all two neighboring *inner* robots r_i and r_{i+1} with $\alpha_i(t) < \pi$ and $\alpha_{i+1}(t) < \pi$. By Lemma 7.12, we obtain

$$\Delta'_{i,i+1}(t) = -\left(\|v_i(t)\|_2 \cdot \cos\left(\frac{\alpha_i(t)}{2}\right) + \|v_{i+1}(t)\|_2 \cdot \cos\left(\frac{\alpha_{i+1}(t)}{2}\right)\right).$$

Based on the protocol's description, $\|v_i(t)\|_2, \|v_{i+1}(t)\|_2 \in \{0,1\}$ depending on the sizes of $\alpha_i(t)$ and $\alpha_{i+1}(t)$. Additionally, since $\alpha_i(t), \alpha_{i+1}(t) \in [0,\pi)$, we obtain both $\cos\left(\frac{\alpha_i(t)}{2}\right) > 0$ and $\cos\left(\frac{\alpha_{i+1}(t)}{2}\right) > 0$. Consequently, $\Delta'_{i,i+1}(t) < 0$ and r_i and r_{i+1} stay in distance at most 1 of each other. Now, we assume without loss of generality that $\alpha_i(t) = \pi$ (the arguments for $\alpha_{i+1}(t)$ are analogous) and $\Delta_{i,i+1}(t) = 1$ (since we consider continuous functions this case must occur before the chain might get disconnected at this point). Next, we assume that $p_i(t) \neq \frac{1}{2}p_{i-1}(t) + \frac{1}{2}p_{i+1}(t)$ and hence, r_i moves with speed 1 to this position. The movement implies $\beta_{i,j}(t) = 0$ and thus

$$\begin{aligned}\Delta'_{i,i+1}(t) &= -\left(1 \cdot \cos(0) + \|v_{i+1}(t)\|_2 \cdot \cos(\beta_{i+1,i}(t))\right)\\ &= -(1 + \|v_{i+1}(t)\| \cdot \cos(\beta_{i+1,i}(t))).\end{aligned}$$

Since $\|v_{i+1}(t)\| \cdot \cos(\beta_{i+1,i}(t))$ is lower bounded by -1 (for $\|v_{i+1}(t)\|_2 = 1$ and $\beta_{i+1,i}(t) = \pi$), it follows $\Delta'_{i,i+1}(t) \leq 0$. In case $p_i(t) = \frac{1}{2}p_{i-1}(t) + \frac{1}{2}p_{i+1}(t)$, r_i follows the movement of $\frac{1}{2}p_{i-1}(t) + \frac{1}{2}p_{i+1}(t)$ and thus, also $\Delta'_{i,i+1}(t) \leq 0$.

Lastly, consider an outer robot and its neighbor, without loss of generality, we consider r_0 and r_1. The arguments for r_{n-2} and r_{n-1} are analogous. Based on the protocol's description, always $\beta_{0,1}(t) = \pi$, given that r_0 moves (which is always the case except for the final configuration). Given $\alpha_1(t) < \pi$ – which implies $\ell(t) = 2$ – and $\Delta_{0,1}(t) = 1$, we obtain that r_1 moves with speed 1 along its angle bisector. Based on the explanations in Section 7.4.2, we conclude

$$\begin{aligned}\Delta'_{0,1}(t) &= -\left(\min\{\cos\left(\frac{\alpha_1(t)}{2}\right), 1-\tau\} \cdot \cos(\pi) + \|v_1(t)\|_2 \cdot \cos\left(\frac{\alpha_1(t)}{2}\right)\right).\\ &= -\left(-\min\{\cos\left(\frac{\alpha_1(t)}{2}\right), 1-\tau\} + \cos\left(\frac{\alpha_1(t)}{2}\right)\right).\end{aligned}$$

Due to the minimum term in the last line, it follows $\Delta'_{0,1}(t) \leq 0$. Lastly, in case $\alpha_1(t) = \pi$ and $\Delta_{0,1}(t) = 1$ both r_0 and r_1 adjust their speed and stay in distance at most 1 based on the protocol's description. ■

Throughout the following analysis, we always take Lemma 7.13 for granted and do not always argue again that the connectivity is maintained.

Lemma 7.14 The index $\ell(t)$ is monotonically increasing and the index $r(t)$ is monotonically decreasing until $\ell(t) = r(t)$.

Proof. We prove that $\ell(t)$ is monotonically increasing. With the same arguments, $r(t)$ is monotonically decreasing. First of all, every robot located in a straight line between its neighbors or in the same position with at least one of its neighbors will never compute a target point that does not lie between its neighbors. Additionally, it will not compute a target point to the left of its left neighbor or the right of its right neighbor. Additionally, all the robots follow the movements of their neighbors and thus all robots between r_0 and $r_{\ell(t)}$ stay on the straight line connecting r_0 and $r_{\ell(t)}$. Hence, none of these robots causes a decrease of $\ell(t)$. ■

Lemma 7.15 $I(t)$ is monotonically decreasing.

Proof. An inner robot r_j can execute three different movements. Either it does not move, moves with speed 1 along the bisector, or follows the movements of the midpoint of its neighbors. Non-moving robots do not increase the length of any vector. Robots r_j that move along the bisector decrease both $\|w_j(t)\|_2$ and $\|w_{j+1}(t)\|_2$ with speed $\cos\left(\frac{\alpha_j(t)}{2}\right) > 0$. This is a conclusion from Lemma 7.12 since $\beta_{j,j-1}(t) = \beta_{j,j+1}(t) = \frac{\alpha_j(t)}{2}$ and r_j moves with speed 1. Robots that follow the movements of their neighbors also cannot increase the length of any vector because they neither follow the movement of r_0 nor the movement of r_{n-1}. Thus, they follow robots that decrease the lengths of neighboring vectors (or do not move at all). Since all possible movements cause a decrease of $I(t)$, $I(t)$ is monotonically decreasing. ■

While $I(t)$ can only decrease (Lemma 7.15), $O_\ell(t)$ and $O_r(t)$ can either increase or decrease, depending on the current sizes of outer angles.

Lemma 7.16 Consider a configuration with $\ell(t) \neq r(t)$ and an outer angle fulfilling $0 \leq \alpha_i(t) < \psi$, for $i \in \{\ell(t), r(t)\}$. Then, $-\tau < O_i'(t) \leq 0$.

Proof. Assume that $i = \ell(t)$ and let us analyze $O_\ell'(t)$. In this configuration, $\beta_{0,\ell}(t) = \pi$ and $\beta_{\ell,0}(t) = \frac{\alpha_\ell(t)}{2}$. Since $\alpha_\ell(t) < \psi$ it follows $\cos(\beta_{\ell,0}(t)) > 1-\tau$. Thus, the outer robot moves at full speed, i.e., $\|v_0(t)\|_2 = 1-\tau$. Lemma 7.12 yields $O_\ell'(t) = -\left(-(1-\tau)+\cos\left(\frac{\alpha_\ell(t)}{2}\right)\right) = 1-\tau-\cos\left(\frac{\alpha_\ell(t)}{2}\right)$. Lastly, we observe $1-\tau < \cos\left(\frac{\alpha_\ell(t)}{2}\right) \leq 1$ and conclude $-\tau < O_i'(t) \leq 0$. ■

In case the outer angle $\alpha_i(t)$ has a size of at least ψ, the robot r_i does not move at all in case $O_i(t) < \gamma_i(t)$. Consequently, the length of $O_i(t)$ increases with constant speed.

Lemma 7.17 Consider a configuration with an outer angle fulfilling $\alpha_i(t) \geq \psi$ and $O_i(t) < \gamma_i(t)$. Then, $O_i'(t) = 1-\tau$.

Proof. Without loss of generality, we assume that $i = \ell(t)$. We prove that $r_{\ell(t)}$ does not move in this case. In case $\|w_\ell(t)\|_2 < 1$, the robot $r_{\ell(t)}$ does not move at all due to the definition of the protocol. Consider the case $\|w_\ell(t)\|_2 = 1$. Since $O_i(t) < \gamma_i(t)$, there must be at least one vector with an index less than $\ell(t)$ that has a length smaller than 1. Let j be the highest index of such a vector so that $\|w_i(t)\|_2 = 1$ for all $j < i < \ell(t)$. The robot r_j moves with speed 1 into the direction of r_{j+1} as $\|w_j(t)\|_2 < \|w_{j+1}(t)\|_2$ and therefore the midpoint between r_{j-1} and r_{j+1} must lie closer to r_{j+1}. Note that $\beta_{j,\ell}(t) = 0$ and thus, by Lemma 7.12, $\Delta_{j,\ell}'(t) = -1$. Since all robots with an index between j and $\ell(t)$ follow the movement of the midpoint between their neighbors, it follows that all vectors $w_i(t)$ for an index $j < i \leq \ell(t)$ decrease in length. Hence, after an infinitesimal time interval, $\|w_\ell(t)\|_2 < 1$ and thus $r_\ell(t)$ does not move at all.

At the same time, r_0 can move with speed $1-\tau$ away from $r_{\ell(t)}$ and increases the distance between r_0 and $r_{\ell(t)}$ with speed $1-\tau$. To see this, we again have to consider two cases. In case $\|w_1(t)\|_2 < 1$, r_0 moves with speed $1-\tau$ due to the definition of the protocol. In the other case, namely $\|w_1(t)\|_2 = 1$ consider the smallest index $k < \ell(t)$ such that $\|w_i(t)\|_2 = 1$ for all $0 < i < k$ and $\|w_k(t)\|_2 < 1$. The robot r_{k-1} moves with speed 1 towards the midpoint between r_{k-2} and r_k. Due to the definition of k, this midpoint lies closer to r_{k-2} than to r_k. This implies $\beta_{k-1,0}(t) = 0$ and by Lemma 7.12 it follows that the movement of r_{k-1} decreases $\Delta_{k-1,0}(t)$ with speed 1. Since all robots with an index between 1 and k follow the midpoint movement between their neighbors, it follows that all vectors $w_i(t)$ for an index $2 \leq i < k$ decrease in length. This implies that $\|w_1(t)\|_2$ decreases after an infinitesimal time interval such that r_0 can move at its maximum speed $1-\tau$.

Combining the movements of r_0 and $r_{\ell(t)}$ it follows that $r_{\ell(t)}$ does not move at all, while r_0 moves with speed $1-\tau$ with an angle $\beta_{0,\ell}(t) = \pi$ such that $O_\ell{}'(t) = 1-\tau$ by Lemma 7.12. The arguments for $O_r(t)$ are analogous.

■

In configurations with an outer angle of size at most ψ, we have that that $I(t)$ decreases with speed $1-\tau$. Therefore, the total time such a configuration can exist is upper bounded by $\frac{n-3}{1-\tau}$ since $I(t)$ is upper bounded by $n-3$.

Lemma 7.18 In configurations having an outer angle $\alpha_i(t) < \psi$, $i \in \{\ell(t), r(t)\}$, we have that $I(t)$ decreases with speed at least $1-\tau$.

Proof. Without loss of generality, we assume that $i = \ell(t)$, the arguments for $i = r(t)$ are analogous. In such a configuration, $\ell(t)$ moves with speed 1 along the bisector of vectors pointing to its neighbors. At the same time, $r_{\ell^+}(t)$ either does not move or it moves with speed 1 along its bisector (provided $\alpha_{\ell^+}(t) < \psi$). In case it does not move, $\Delta_{\ell,\ell^+}(t)$ decreases with speed $\cos\left(\frac{\alpha_\ell(t)}{2}\right) \geq 1-\tau$, as $\beta_{\ell,\ell^+}(t) = \frac{\alpha_\ell(t)}{2}$. In case both move, $\Delta_{\ell,\ell^+}(t)$ decreases with speed $\cos\left(\frac{\alpha_\ell(t)}{2}\right) + \cos\left(\frac{\alpha_{\ell^+}(t)}{2}\right) \geq 2\cdot(1-\tau)$. This can be derived from Lemma 7.12 since $\beta_{\ell,\ell^+}(t) = \frac{\alpha_\ell(t)}{2}$ and $\beta_{\ell^+,\ell}(t) = \frac{\alpha_{\ell^+}(t)}{2}$ and $\|v_\ell(t)\|_2 = \|v_{\ell^+}(t)\|_2 = 1$. Since $\Delta_{\ell,\ell^+}(t)$ is part of $I(t)$, we can conclude that $I(t)$ decreases with speed at least $1-\tau$. ■

Corollary 7.1 The total time an outer angle of size less than ψ exists while $\ell(t) \neq r(t)$, is upper bounded by $\frac{n-3}{1-\tau}$.

Next, we analyze the behavior of outer angles that have a size of at least ψ. A robot corresponding to an outer angle $\alpha_i(t) \geq \psi$ only moves in case $O_i(t) = \gamma_i(t)$. The following lemmata assume that $O_i(t) = \gamma_i(t)$.

Lemma 7.19 Assume that $i \in \{\ell(t), r(t)\}$. In configurations with an outer angle of size $\alpha_i(t) \in [\psi, \frac{3}{4}\pi]$ while $O_i(t) = \gamma_i(t)$ and $\ell(t) \neq r(t)$ we have that $I'(t) \leq -\frac{\sqrt{2-\sqrt{2}}}{2}$.

Proof. Without loss of generality, we assume that $i = \ell(t)$. Since $O_\ell(t) = \gamma_\ell(t)$, we have that $\|w_\ell(t)\|_2 = 1$ and thus $r_{\ell(t)}$ moves with speed 1 along the bisector formed by vectors pointing to its neighbors. By noticing $\beta_{\ell,0}(t) = \frac{\alpha_\ell(t)}{2}$ and applying Lemma 7.12, we conclude that the movement of $r_\ell(t)$ decreases $\Delta_{\ell,\ell^+}(t)$ with speed at least $\cos\left(\frac{3}{8}\pi\right) = \frac{\sqrt{2-\sqrt{2}}}{2}$ and at most $\cos\left(\frac{\psi}{2}\right) = 1-\tau$. As r_0 can move with speed at most $1-\tau$, r_0 moves fast enough such that the distance between r_0 and $r_{\ell(t)}$ does not decrease and thus $O_\ell(t)$ and especially $\|w_\ell(t)\|_2$ remains constant. Hence, $r_{\ell(t)}$ continues moving along its bisector while decreasing $I(t)$ with speed at least $\frac{\sqrt{2-\sqrt{2}}}{2}$ since all other

vectors that are part of $I(t)$ are monotonically decreasing with the same arguments as used in the proof of Lemma 7.15. ■

Corollary 7.2 The total time an outer angle $\alpha_i(t)$ of size $\alpha_i(t) \in [\psi, \frac{3}{4}\pi]$ exists, while $O_i(t) = \gamma_i(t)$ and $\ell(t) \neq r(t)$, is upper bounded by $\frac{2\cdot(n-3)}{\sqrt{2-\sqrt{2}}}$.

It remains to analyze outer angles that have a size of at least $\frac{3}{4}\pi$. It turns out that an outer angle of size at least $\frac{3}{4}\pi$ only increases.

Lemma 7.20 Assume that $i \in \{\ell(t), r(t)\}$. In configurations fulfilling $\ell(t) \neq r(t)$, $\alpha_i(t) \geq \frac{3}{4}\pi$ and $O_i(t) = \gamma_i(t)$, the size of an outer angle $\alpha_i(t)$ is monotonically increasing.

Proof. We give the proof for $\alpha_\ell(t)$. For this, we rewrite $\alpha_\ell(t) = \pi - c$ for $0 \leq c < \frac{1}{4}\pi$. Let $f_\ell(t) = \cos(\alpha_\ell(t))$. We compute the derivation $f_\ell{}'(t)$ and prove $f_\ell{}'(t) < 0$. As $\cos(x)$ is monotonically decreasing in the interval $[\frac{3}{4}\pi, \pi)$, this proves that $\alpha_i(t)$ is monotonically increasing. Let $\beta^-(t)$ be the angle enclosed by the line segments connecting r_0 and $r_{\ell^+(t)}$ and r_0 and $r_{\ell(t)}$. Similarly, let $\beta^+(t)$ denote the angle enclosed by the line segments connecting r_0 and $r_{\ell^+(t)}$ and $r_{\ell^+(t)}$ and $r_{\ell(t)}$. See Figure 7.7 for a visualization.

We start by giving a formula for $f_\ell(t)$ and compute its derivation. Note that $O_\ell(t)$ stays constant as $\beta_{\ell,0}(t) = \frac{\alpha_\ell(t)}{2}$ and thus the movement of $r_{\ell(t)}$ decreases $O_\ell(t)$ with speed at most $\cos\left(\frac{3}{8}\right)$. Since $\cos\left(\frac{3}{8}\right) < 1 - \tau$, r_1 moves fast enough such that $O_\ell(t) = \gamma_\ell(t)$ remains constant.

Now, consider the triangle formed by r_0, $r_{\ell(t)}$ and $r_{\ell^+(t)}$. By our assumption, $\Delta_{0,\ell}(t) = \gamma_\ell(t)$. Via the law of cosines, we obtain

$$\Delta_{0,\ell^+}(t)^2 = \gamma_\ell(t)^2 + \Delta_{\ell,\ell^+}(t)^2 - 2 \cdot \Delta_{0,\ell^+}(t) \cdot \Delta_{\ell,\ell^+}(t) \cdot \cos(\alpha_\ell(t)).$$

By substituting $\cos(\alpha_\ell(t))$ by $f_\ell(t)$ and rearranging the terms, we get the following formula for $f_\ell(t)$.

$$f_\ell(t) = \frac{\gamma_\ell(t)^2 + \Delta_{\ell,\ell^+}(t)^2 - \Delta_{0,\ell^+}(t)^2}{2 \cdot \gamma_\ell(t) \cdot \Delta_{\ell,\ell^+}(t)}.$$

Now, we compute $f_\ell{}'(t)$. Remember that $\gamma_\ell{}'(t) = \gamma_\ell(t)$ as stated above.

$$f_\ell{}'(t) = \frac{\Delta_{\ell,\ell^+}{}'(t) \cdot \left(-\gamma_\ell(t)^2 + \Delta_{\ell,\ell^+}(t)^2 + \Delta_{0,\ell^+}(t)^2\right)}{2 \cdot \gamma_\ell(t) \cdot \Delta_{\ell,\ell^+}(t)^2} - \frac{2 \cdot \Delta_{\ell,\ell^+}(t) \cdot \Delta_{0,\ell^+}(t) \Delta_{\ell,\ell^+}{}'(t)}{2 \cdot \gamma_\ell(t) \cdot \Delta_{\ell,\ell^+}(t)^2}$$

Applying the law of cosines again gives us

$$-\gamma_\ell(t)^2 + \Delta_{0,\ell^+}(t)^2 + \Delta_{\ell,\ell^+}(t)^2 = 2 \cdot \Delta_{\ell,\ell^+}(t) \cdot \Delta_{0,\ell^+}(t) \cdot \cos\left(\beta^+(t)\right).$$

Replacing this in the original formula for $f_\ell{}'(t)$ yields

$$f_\ell{}'(t) = \frac{\Delta_{0,\ell^+}(t)}{\gamma_\ell(t) \cdot \Delta_{\ell,\ell^+}(t)} \cdot \left(\Delta_{\ell,\ell^+}{}'(t) \cdot \cos\left(\beta^+(t)\right) - \Delta_{0,\ell^+}{}'(t)\right).$$

Now, we have to consider two cases. Either $r_{\ell^+(t)}$ is moving and thus $\alpha_{\ell^+}(t) \leq \psi$ or $r_{\ell^+(t)}$ does not move. In both cases, $\Delta_{\ell,\ell^+}(t) \leq 0$ as $\ell(t) \neq r(t)$ (Lemma 7.15). We start by analyzing the case that $r_{\ell^+(t)}$ does not move. Observe $\beta_{0,\ell^+}(t) = \pi - \beta^-(t)$. By Lemma 7.12, we obtain $\Delta_{0,\ell^+}{}'(t) = -\|v_0(t)\|_2 \cdot \cos(\pi - \beta^-(t)) > 0$ as $\beta^-(t)$ can be upper bounded by $\frac{1}{4}\pi$ and thus $\pi - \beta^-(t) > \frac{3}{4}\pi$. We can conclude, $-\Delta_{0,\ell^+}{}'(t) < 0$ and $f_\ell{}'(t) < 0$ in this case. As both $\Delta_{\ell,\ell^+}(t) \leq 0$ and $-\Delta_{0,\ell^+}{}'(t) < 0$

it follows $f_\ell'(t) < 0$. It remains to analyze the case that $r_{\ell^+}(t)$ is moving. We compute ${\Delta_{0,\ell^+}}'(t)$. Depending on the orientation of $\alpha_{\ell^+}(t)$, there are two possible variants of ${\Delta_{0,\ell^+}}'(t)$. Either $\alpha_\ell(t)$ and $\alpha_{\ell^+}(t)$ have the same or different orientations. Consider the case that $\alpha_\ell(t)$ and $\alpha_{\ell^+}(t)$ have the same orientation. In this case

$$\Delta_{0,\ell^+}'(t) = \cos\left(\frac{\alpha_\ell(t)}{2}\right) \cdot \cos\left(\beta^-(t)\right) - \cos\left(\frac{\alpha_{\ell^+}(t)}{2} - \beta^+(t)\right). \tag{7.8}$$

In the other case, we have that

$$\begin{aligned}\Delta_{0,\ell^+}'(t) &= \cos\left(\frac{\alpha_\ell(t)}{2}\right) \cdot \cos\left(\beta^-(t)\right) - \cos\left(\frac{\alpha_{\ell^+}(t)}{2} + \beta^+(t)\right) \\ &\geq \cos\left(\frac{\alpha_\ell(t)}{2}\right) \cdot \cos\left(\beta^-(t)\right) - \cos\left(\frac{\alpha_{\ell^+}(t)}{2} - \beta^+(t)\right).\end{aligned}$$

Thus, we assume that Equation (7.8) holds. Note that $\cos(a-b) = \sin(a)\cdot\sin(b) + \cos(a) + \cos(b)$ and thus $\cos\left(\frac{\alpha_{\ell^+}(t)}{2} - \beta^+(t)\right) = \sin\left(\frac{\alpha_{\ell^+}(t)}{2}\right)\cdot\sin(\beta^+(t)) + \cos\left(\frac{\alpha_{\ell^+}(t)}{2}\right)\cdot\cos(\beta^+(t))$. Additionally, we obtain via Lemma 7.12, ${\Delta_{\ell,\ell^+}}'(t) = -\left(\cos\left(\frac{\alpha_\ell(t)}{2}\right) + \cos\left(\frac{\alpha_{\ell^+}(t)}{2}\right)\right)$. For improved readability, define

$$\begin{aligned}\sigma(t) &:= \frac{\Delta_{0,\ell^+}(t)}{\gamma_\ell(t)\cdot\Delta_{\ell,\ell^+}(t)} \\ \mu(t) &:= \left(\cos\left(\beta^-(t)\right) + \cos\left(\beta^+(t)\right)\right).\end{aligned}$$

Next, we plug all these insights into $f_\ell'(t)$.

$$\begin{aligned}f_\ell'(t) &= \sigma(t)\left(\sin\left(\frac{\alpha_{\ell^+}(t)}{2}\right)\cdot\sin\left(\beta^+(t)\right) - \cos\left(\frac{\alpha_\ell(t)}{2}\right)\cdot\mu(t)\right) \\ &\leq \sigma(t)\left(\sin(\psi)\cdot\sin\left(\beta^+(t)\right) - \cos\left(\frac{\alpha_\ell(t)}{2}\right)\cdot\mu(t)\right) \\ &= \sigma(t)\left(\sqrt{1-(1-\tau)^2}\cdot\sin\left(\beta^+(t)\right) - \cos\left(\frac{\alpha_\ell(t)}{2}\right)\cdot\mu(t)\right) \qquad (7.9) \\ &\leq \sigma(t)\left(\frac{\sqrt{3}}{2}\cdot\sin\left(\beta^+(t)\right) - \cos\left(\frac{\alpha_\ell(t)}{2}\right)\cdot\mu(t)\right) \qquad (7.10) \\ &\leq \sigma(t)\left(\frac{\sqrt{3}}{2}\cdot\sin(c) - \sin(c)\cdot\cos\left(\frac{c}{2} - \beta^+(t)\right)\right) \\ &\leq \sigma(t)\left(\frac{\sqrt{3}}{2}\cdot\sin(c) - \sin(c)\cdot\cos\left(\frac{c}{2}\right)\right) \\ &= \sigma(t)\sin(c)\cdot\left(\frac{\sqrt{3}}{2} - \cos\left(\frac{c}{2}\right)\right) < 0\end{aligned}$$

We use the following observations. Equation (7.9) holds as $\sin\left(\cos^{-1}(x)\right) = \sqrt{1-x^2}$. Additionally, $\cos\left(\frac{\pi}{2} - x\right) = \sin(x)$ and thus $\cos\left(\frac{\pi}{2} - \frac{c}{2}\right) = \sin\left(\frac{c}{2}\right)$. For Equation (8.1), note that $\beta^-(t) = \pi - \alpha_\ell(t) - \beta^+(t)$ as the sum of internal angles of a triangle is equal to π. Hence, we can rewrite $\cos(\beta^-(t)) = \cos(\pi - \alpha_\ell(t) - \beta^+(t)) = \cos(\pi - \pi + c - \beta^+(t)) = \cos(c - \beta^+(t))$. Observe that $\cos(\beta^+(t)) + \cos(c - \beta^+(t)) = 2\cdot\cos\left(\frac{c}{2}\right)\cdot\cos\left(\frac{c}{2} - \beta^+(t)\right)$. As a last step we use the

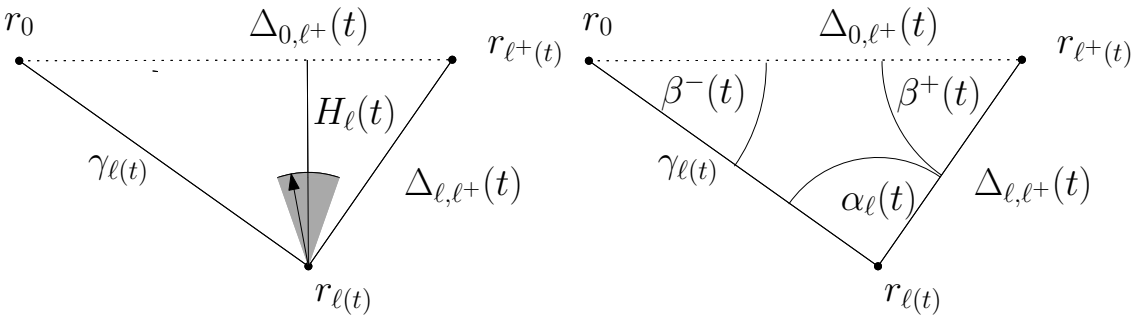

Figure 7.7: Visualization of $H_\ell(t)$. To the left, the gray part marks the area of all possible vectors $v_\ell(t)$. In the right triangle, the definitions of all angles are depicted (used in the proofs of Lemmata 7.20 and 7.21).

equality $2 \cdot \cos\left(\frac{x}{2}\right) \cdot \sin\left(\frac{x}{2}\right) = \sin(x)$. Plugging all statements together yields $\sin\left(\frac{c}{2}\right) \cdot \mu(t) = \sin(c) \cdot \cos\left(\frac{c}{2} - \beta^+(t)\right)$. Finally, we can conclude that $f_\ell(t)$ is monotonically decreasing for $\alpha_\ell(t) \in \left[\frac{3}{4}\pi, \pi\right]$ and, thus, that $\alpha_\ell(t)$ is monotonically increasing in the same interval. ■

To bound the total time such large outer angles can exist, we cannot use $I(t)$ as a progress measure because for very large angles in the order of $\pi - \frac{1}{n}$, $I(t)$ does not decrease with constant speed anymore. Therefore, we introduce another progress measure that decreases with constant speed in this case (see Figure 7.7).

Definition 7.5 Define $H_\ell(t)$ to be the distance of $r_{\ell(t)}$ to the line segment connecting r_0 and $r_{\ell^+(t)}$ and define $H_r(t)$ to be the distance of $r_{r(t)}$ to the line segment connecting $r_{r^+(t)}$ and r_{n-1}.

Lemma 7.21 In configurations fulfilling $\ell(t) \neq r(t)$, $\alpha_i(t) \geq \frac{3}{4}\pi$, $i \in \{\ell(t), r(t)\}$ and $O_i(t) = \gamma_i(t)$, the inequality $H_i'(t) \leq -\frac{1}{20}$ holds.

Proof. We assume that $i = \ell(t)$, the proof for $i = r(t)$ is analogous. We have to analyze the movements of $r_0, r_{\ell(t)}$ and $r_{\ell^+(t)}$ in this case. We rewrite $\alpha_\ell(t) = \pi - c$ for $c \leq \frac{\pi}{4}$. Without loss of generality, assume that $r_{\ell(t)}$ is located in the origin and the line segment connecting r_1 and $r_{\ell^+(t)}$ to be a parallel line to the x-axis above of $r_{\ell(t)}$. As all robots between r_1 and $r_{\ell(t)}$ as well as all robots $r_{\ell(t)}, \ldots r_{\ell^+(t)}$ are collinear, $v_\ell(t)$ must point upwards. Since $\alpha_\ell(t) \geq \frac{3}{4}\pi$, $v_\ell(t)$ must form an angle of size less than $\frac{\pi}{8}$ with the y-axis. Hence, $r_{\ell(t)}$ moves with speed at least $\cos\left(\frac{\pi}{8}\right) > 0.92$ upwards. At the same time the robots r_0 and $r_{\ell^+(t)}$ could move. Consider the movement of r_0. Similar to the proof of Lemma 7.20 (see also Figure 7.7), let $\beta^-(t)$ be the angle formed by vectors pointing from r_0 to $r_{\ell(t)}$ and from r_0 to $r_{\ell^+(t)}$ and let $\beta^+(t)$ be the angle formed by vectors pointing from $r_{\ell^+(t)}$ to r_0 and from $r_{\ell^+(t)}$ to r_ℓ. $v_0(t) = -\cos\left(\frac{\alpha_i(t)}{2}\right) \cdot \widehat{w}_1(t)$. Thus, $\beta_{0,\ell}(t) = \pi - \beta^-(t)$. Therefore, r_0 moves upwards with speed $\sin(\beta^-(t)) \cdot \cos\left(\frac{\alpha_\ell(t)}{2}\right) = \sin(\beta^-(t)) \cdot \sin\left(\frac{\pi}{2} - \frac{\alpha_\ell(t)}{2}\right) = \sin(\beta^-(t)) \cdot \sin\left(\frac{c}{2}\right)$. Lastly, observe $\beta^-(t) \leq c$ as the sum of internal angles of a triangle is π. Hence, r_0 moves upwards with speed at most $\sin(c) \cdot \sin\left(\frac{c}{2}\right) \leq \sin\left(\frac{1}{4}\pi\right) \cdot \sin\left(\frac{\pi}{8}\right) < 0.28$. It remains to analyze the speed of $r_{\ell^+(t)}$ moving upwards. As $r(t) \neq \ell(t)$, $r_{\ell^+(t)}$ either does not move at all or $\alpha_{\ell^+}(t) < \psi$. In case $r_{\ell^+(t)}$ does not move at all $H_\ell'(t) <= -0.92 + 0.28 = -0.64$. Now, consider $\alpha_{\ell^+}(t) < \psi$. $\beta^+(t)$ could be almost 0, such that in the worst case the angle formed by $v_{\ell^+}(t)$ and the line segment connecting r_0 and $r_{\ell^+(t)}$ lies completely above that line segment. In this case, $r_{\ell^+(t)}$ moves upwards with speed at most $\sin\left(\frac{\psi}{2}\right) = \sqrt{1-(1-\tau)^2} = \sqrt{1-1+2\cdot\tau-\tau^2} = \sqrt{2\cdot\tau-\tau^2} \leq \sqrt{2\cdot\frac{1}{2}-\frac{1}{4}} = \frac{\sqrt{3}}{2} < 0.87$. Therefore, $H_\ell'(t) \leq -0.92 + 0.87 = -0.05 = -\frac{1}{20}$. ■

By noticing that $H_i(t)$, for $i \in \{\ell(t), r(t)\}$, is upper bounded by $\Delta_{i,i^+}(t) \leq |i - i^+(t)|$ and using the fact that $H_i(t)$ cannot decrease anymore since $\alpha_i(t)$ is monotonically increasing, we can derive the total time an outer angle can have a size of at least $\frac{3}{4}\pi$ while $O_i(t) = \gamma_i(t)$.

Corollary 7.3 The total time an outer angle of size at least $\frac{3}{4}\pi$ exists, while $O_i(t) = \gamma_i(t)$ and $\ell(t) \neq r(t)$, is upper bounded by $20 \cdot (n-3)$.

A combination of the preceding insights leads to the following upper bound until $\ell(t) = r(t)$.

Lemma 7.22 After time at most $2 \cdot (n-3) \cdot \left(\frac{1}{1-\tau} + \frac{1}{\sqrt{2-\sqrt{2}}} + 10 \right)$, we have that $\ell(t) = r(t)$.

Proof. The total time in configurations with an outer angle of size at most ψ is upper bounded by $\frac{n-3}{1-\tau}$ (Corollary 7.1). It remains to bound the time of configurations with larger outer angles. As soon as an outer angle reaches a size of at least ψ, $O_i(t)$ increases with speed $1-\tau$ (Lemma 7.17) until it reaches maximal length. Additionally, by Lemma 7.16, $O_i(t)$ will only decrease in the future in case the corresponding outer angle has a size of less than ψ. Hence, the total time all $O_i(t)$ can decrease is upper bounded by $\frac{n-3}{1-\tau}$, the total time outer angles can have a size of less than ψ. In case an outer angle $\alpha_i(t)$ has a size of at least ψ and $O_i(t) < \gamma_i(t)$, the outer length $O_i(t)$ increases with speed $1-\tau$ (Lemma 7.17). As $O_i(t)$ is upper bounded by $n-3$ while $\ell(t) \neq r(t)$, the total time all $O_i(t)$ can increase with speed $1-\tau$ is upper bounded by $\frac{n-3}{1-\tau}$. Furthermore, the total time $O_i(t) = \gamma_i(t)$ and $\alpha_i(t) \leq \frac{3}{4}\pi$ is upper bounded by $\frac{2\cdot(n-3)}{\sqrt{2-\sqrt{2}}}$ (Corollary 7.2). The total time $O_i(t) = \gamma_i(t)$ and $\alpha_i(t) \geq \frac{3}{4}\pi$ is upper bounded by $20 \cdot (n-3)$. Hence, the total time needed until $\ell(t) = r(t)$ is upper bounded by $\frac{2\cdot(n-3)}{1-\tau} + \frac{2\cdot(n-3)}{\sqrt{2-\sqrt{2}}} + 20 \cdot (n-3) = 2 \cdot (n-3) \cdot \left(\frac{1}{1-\tau} + \frac{1}{\sqrt{2-\sqrt{2}}} + 10 \right)$. ■

Lastly, it remains to analyze the case $\ell(t) = r(t)$. The combination of Lemmata 7.22 and 7.23 yields a total runtime bound.

Lemma 7.23 Assume that $\ell(t) = r(t)$. Then, after time $3n \cdot \left(\frac{1}{\tau} + \frac{1}{1-\tau}\right)$, the configuration is transformed into a max-chain or all robots are located at the same position.

For the proof of Lemma 7.23, we need some lemmata about one-dimensional configurations.

Lemma 7.24 Starting MAX-MOB in a one-dimensional configuration, $\ell(t) \neq r(t)$ can hold for time at most $\frac{n-3}{2}$.

Proof. The robot r_i moves with speed 1 along the bisector of vectors pointing to its neighbors. This movement decreases $\Delta_{i,i^+}(t)$ with speed $\cos\left(\frac{\alpha_i(t)}{2}\right) = \cos(0) = 1$. This can be derived from Lemma 7.12 by observing $\beta_{i,i^+}(t) = \frac{\alpha_i(t)}{2} = 0$. At the same time, $r_{i^+(t)}$ is defined such that $\alpha_{i^+}(t) < \pi$. Thus, $r_{i^+(t)}$ also moves along the bisector between its neighbors. Hence, the movement of $r_{i^+(t)}$ can also not increase $\Delta_{i,i^+}(t)$. All robots in between stay on the straight line between $r_{i(t)}$ and $r_{i^+(t)}$. Lastly, observe $\Delta_{i,i^+}(t)$ is part of $I(t)$. Thus, $I(t)$ decreases with speed at least $\cos\left(\frac{\alpha_\ell(t)}{2}\right) + \cos\left(\frac{\alpha_r(t)}{2}\right) = 2 \cdot \cos(0) \;=\; 2$. ■

Lemma 7.25 A one-dimensional configuration fulfilling $\ell(t) = r(t) = 0$ is transformed into a max-chain after time at most $\frac{n-1}{2\cdot(1-\tau)}$.

Proof. In case $\ell(t) = r(t) = 0$, we have that $\alpha_\ell(t) = \pi$ and thus all vectors point into the same direction rendering the configuration an opposed configuration. In opposed configurations, the outer robots move both with speed $(1-\tau)$ away from each other such that their distance decreases with speed $2 \cdot (1-\tau)$. Since the maximal distance is $n-1$, a straight line of length $n-1$ is obtained after time at most $\frac{n-1}{2\cdot(1-\tau)}$. ■

Lemma 7.26 A one-dimensional configuration fulfilling $\ell(t) = r(t) = x$ for $0 < x < n-1$ and $p_0(t) \neq p_{n-1}(t)$ is transformed into a max-chain after time at most $\frac{n-1}{2} \cdot \left(\frac{1}{\tau} + \frac{1}{1-\tau}\right)$.

Proof. In such a configuration, $O_\ell(t) \neq O_r(t)$ and $\alpha_\ell(t) = 0$. The robot $r_{\ell(t)}$ moves with speed 1 towards the two outer robots. The outer robots move with speed $1-\tau$ away from $r_{\ell(t)}$. These movements cause a decrease of both $O_\ell(t)$ and $O_r(t)$ with speed τ. To see this, note that $\beta_{0,\ell}(t) = \beta_{n-1,\ell}(t) = \pi$ and $\beta_{\ell,0}(t) = \beta_{\ell,n-1}(t) = 0$. Lemma 7.12 yields $\Delta_{0,\ell}{}'(t) = O_\ell{}'(t) = -\|v_0(t)\|_2 \cdot \cos\beta_{0,\ell}(t) - 1 \cdot \cos\beta_{\ell,1}(t) = -(1-\tau) + 1 = -\tau$. As a consequence, after time at most $\frac{n-1}{2\cdot\tau}$ either $O_\ell(t)$ or $O_r(t)$ reaches size 0 such that in the following $\ell(t) = r(t) = 0$. Such a configuration is transformed into a straight line of length $n-1$ after time at most $\frac{n-1}{2\cdot(1-\tau)}$ (Lemma 7.24). ■

Lemma 7.27 A one-dimensional configuration fulfilling $\ell(t) = r(t) = x$ for $0 < x < n-1$ and $p_0(t) = p_{n-1}(t)$ is transformed into a configuration in which all robots are located on the same position after time at most $\frac{n-1}{2\cdot\tau}$.

Proof. Since the outer robots are located at the same position it must hold $O_\ell(t) = O_r(t)$ and $\alpha_\ell(t) = 0$. $r_{\ell(t)}$ moves with speed 1 towards the two outer robots. In this configuration, $\beta_{0,\ell}(t) = \beta_{n-1,\ell}(t) = \pi$ and $\beta_{\ell,0}(t) = \beta_{\ell,n-1}(t) = 0$. Combined with $\|v_0(t)\|_2 = \|v_{n-1}(t)\|_2 = 1-\tau$ and $\|v_\ell(t)\|_2 = 1$, Lemma 7.12 gives us $\Delta_{0,\ell}{}'(t) = O_\ell{}'(t) = -(-(1-\tau)+1) = -\tau$. As $O_\ell(t)$ and $O_r(t)$ are both bounded by $\frac{n-1}{2}$, the lemma follows. ■

Next, we consider a special two-dimensional configuration fulfilling $\ell(t) = r(t)$ and analyze the runtime started in such a configuration. See Figure 7.8 for a visualization of these configurations.

Definition 7.6 Let δ be a positive constant and define $\theta = 2 \cdot \sin^{-1}\left(\frac{\delta}{n}\right)$. For n odd, the *continuous δ-V-configuration* forms an isosceles triangle with $\|w_i(t)\|_2 = 1$ for all $1 \leq i \leq n-1$, $\alpha_{n/2}(t) = \theta$ and all other angles equal to π.

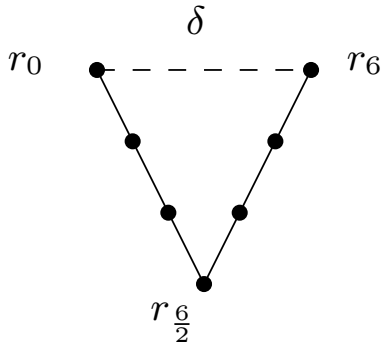

Figure 7.8: A continuous δ-V-configuration.

Lemma 7.28 Starting in a continuous δ-V-configuration, MAX-MOB needs time at most $n \cdot \left(\frac{1}{\tau} + \frac{1}{1-\tau}\right)$ to transform the configuration into a max-chain.

Proof. Fix a point in time 0 in which the configuration is a continuous δ-V-configuration. Note that a continuous δ-V-configuration forms an isosceles triangle whose legs have a length of $\frac{n}{2}$ and the base ($\Delta_{0,n-1}(0)$) has a length of δ. Without loss of generality, we assume that $p_\ell(0) = (0,0)$, $p_0(0) = (x_0, y_0)$ and $p_{n-1}(0) = (x_{n-1}, y_{n-1})$ with $x_0 = -\frac{\delta}{2}, y_0 = \frac{n}{2} \cdot \cos\left(\frac{\theta}{2}\right)$, $x_{n-1} = \frac{\delta}{2}$ and $y_{n-1} = \frac{n}{2} \cdot \cos\left(\frac{\theta}{2}\right)$. Consider the case $\alpha_\ell(t) = \theta < \psi$. $r_{\ell(t)}$ moves with speed 1 upwards. As r_0 and r_{n-1} move with speed at most $1-\tau$, $H_\ell(t)$ and $H_r(t)$ decrease with speed at least τ. As $H_\ell(0) = \frac{n}{2} \cdot \cos\left(\frac{\theta}{2}\right) < \frac{n}{2}$, $\alpha_\ell(t) \geq \psi$ must hold after time at most $\frac{n}{2 \cdot \tau}$, otherwise $H_\ell(t) = 0$ and, thus, $\alpha_\ell(t) = \pi$. Since $\alpha_\ell(t) < \psi$ initially and $\alpha_\ell(t)$ changes continuously, $\alpha_\ell(t) \geq \psi$ must hold before $H_\ell(t) = 0$. As soon as $\alpha_\ell(t)$ reaches a size of ψ, $r_{\ell(t)}$ stops moving as its movement has decreased $O_\ell(t)$ and $O_r(t)$. Then, both $O_\ell(t)$ and $O_r(t)$ increase with speed $1-\tau$ (Lemma 7.17). As $O_\ell(t)$ and $O_r(t)$ are bounded by $\frac{n}{2}$ we have that $O_\ell(t) = \gamma_\ell(t)$ and $O_r(t) = \gamma_r(t)$ after time at most $\frac{1}{1-\tau} \cdot \frac{n}{2}$. Afterward, $r_{\ell(t)}$ continues moving with speed 1 upwards, decreasing $H_\ell(t)$ with speed at least τ (we can apply the same arguments as before). Thus, finally after additional time of at most $\frac{n}{2 \cdot \tau}$, we have $H_\ell(t) = 0$ and the configuration is a one-dimensional opposed configuration that is transformed into a straight line of length $n-1$ after time at most $\frac{n-1}{2 \cdot (1-\tau)}$ (Lemma 7.25). We conclude that the total time is upper bounded by $\frac{2 \cdot n}{2 \cdot \tau} + \frac{n}{2} \cdot \frac{1}{1-\tau} + \frac{n-1}{2 \cdot (1-\tau)} < n \cdot \left(\frac{1}{\tau} + \frac{1}{1-\tau}\right)$. ■

Proof of Lemma 7.23. There are two cases in which $\ell(t) = r(t)$. Either $\ell(t) = r(t) = 0$ or $\ell(t) = r(t) = j$ with $1 \leq j \leq n-2$. In case $\ell(t) = r(t) = 0$, this means that either every vector is the 0-vector and thus all robots are located on a single point, or all robots are located on the same line in an opposed configuration. This is a one-dimensional configuration and transformed into a line of length $n-1$ after time at most $\frac{n}{2 \cdot (1-\tau)}$ (Lemma 7.25). It remains to consider $\ell(t) = r(t) = j$ with $1 \leq j \leq n-2$. In case $\alpha_\ell(t) = 0$ the configuration is one-dimensional and transformed into a straight line or a single point after time at most $\frac{n-1}{2} \cdot \left(\frac{1}{\tau} + \frac{1}{1-\tau}\right)$ (Lemmata 7.26 and 7.27). Assume that $\alpha_\ell(t) > 0$. This configuration has only a single angle of size less than π. For the special case of a continuous δ-V-configuration we have proven a runtime of at most $n \cdot \left(\frac{1}{\tau} + \frac{1}{1-\tau}\right)$ in Lemma 7.28. Now, suppose the triangle formed by $r_o, r_{\ell(t)}$ and r_{n-1} is not isosceles. Without loss of generality, we assume that $\Delta_{\ell,n-1}(t) < \Delta_{0,\ell}(t)$. In this case, enlarge the line segment connecting $r_{\ell(t)}$ and r_{n-1} such that it has length $\Delta_{0,\ell}(t)$ and place a virtual robot r_v at the end of this line segment. Now, the triangle formed by r_0, $r_\ell(t)$ and r_v is an isosceles triangle. Assume that the virtual robot r_v moves exactly as r_{n-1}. Define $H_v(t)$ to be the distance of $r_{\ell(t)}$ to the line segment connecting r_0 and r_v. $r_{\ell(t)}$ moves with speed 1 upwards while both r_0 and r_v can move with speed at most $1-\tau$ upwards. The rest of the argumentation is analogous to continuous δ-V-configurations with the only difference that $H_v(t)$ is upper bounded by $n-2$ (in case $O_\ell(t) = n-2$ and $O_r(t) = 1$ or vice versa). Thus the total time spent in such a configuration can be bounded by $2n \cdot \left(\frac{1}{\tau} + \frac{1}{1-\tau}\right) + \frac{n}{2 \cdot (1-\tau)} < 3n \cdot \left(\frac{1}{\tau} + \frac{1}{1-\tau}\right)$. ■

Finally, the combination of Lemmata 7.22 and 7.23 yields the main theorem.

Theorem 7.12 Starting MAX-MOB in a two-dimensional configuration, the initial chain is either transformed into a straight line of length $n-1$ or all robots are located at the same position after time $\mathcal{O}(n)$.

7.5 On the Speed of the Outer Robots

We conclude by discussing the role of the speeds of outer robots in MAX-MOB and MAX-GTM, revealing an interesting runtime gap between the continuous time model and the $\mathcal{F}$SYNC scheduler. There are two classes of configurations that play an important role in this discussion, *discrete* δ-V-configurations (Definition 7.1 and Figure 7.3) and *continuous* δ-V-configurations (Definition 7.6

and Figure 7.8). With the help of these configurations, we give evidence of why the speed of outer robots is reduced to $1-\tau$ in MAX-MOB.

Initially, we show the difference between the behavior of MAX-MOB and the analogous protocol in which outer robots always move with full speed started in continuous δ-V-configurations. In Section 7.4.4 we have already seen that MAX-MOB resolves continuous δ-V-configurations in time $\mathcal{O}(n)$. Consider now the naïve approach that the maximum speed of the outer robots is not reduced by a constant in MAX-MOB (thus, the maximum speed is 1). We call this protocol NAIVE-MAX-MOB. We prove that the runtime of NAIVE-MAX-MOB for continuous δ-V-configurations also depends on δ, exactly as in the lower bound for the discrete case. Thus, the inner robots must move significantly faster upwards than the outer robots in these configurations to flatten the chain.

Theorem 7.13 NAIVE-MAX-MOB transforms a continuous δ-V-configuration into a max-chain in time $\Omega(n \cdot \log(1/\delta))$.

For the proof of Theorem 7.13, we state the following lemma.

Lemma 7.29 When applying NAIVE-MAX-MOB to continuous δ-V-configurations,

$$\Delta_{0,n-1}{}'(t) = \Delta_{0,n-1}(t)\cos\left(\frac{\theta}{2}\right)\cdot\frac{2}{n}$$

.

Proof. Due to the symmetry, we obtain $\beta_{0,\ell}(t) = \beta_{n-1,\ell}(t) = \frac{\pi}{2} + \frac{\theta}{2}$. Additionally, the distances $\Delta_{\ell,\ell^+}(t)$ and $\Delta_{r,r^+}(t)$ remain constant, because the outer robots can move with speed 1. Hence, the outer robots move with speed $\cos\left(\frac{\theta(t)}{2}\right)$ as the robot $r_{\frac{n}{2}}$ reduces $\Delta_{\ell,\ell^+}(t)$ and $\Delta_{r,r^+}(t)$ with speed $\cos\left(\frac{\theta(t)}{2}\right)$. Thus, we can calculate $\Delta_{0,n-1}{}'(t)$ according to Lemma 7.12.

$$\begin{aligned}\Delta_{0,n-1}{}'(t) &= -2\cdot\left(\cos\left(\frac{\theta}{2}\right)\cdot\cos\left(\frac{\pi}{2}+\frac{\theta}{2}\right)\right)\\ &= -2\cdot\left(-\cos\left(\frac{\theta}{2}\right)\cdot\sin\left(\frac{\theta}{2}\right)\right)\\ &= \sin(\theta)\end{aligned}$$

Via the law of sines, we then obtain

$$\frac{\Delta_{0,n-1}(t)}{\sin(\theta)} = \frac{\frac{n}{2}}{\sin\left(\frac{\pi-\theta}{2}\right)} \iff \sin(\theta) = \frac{\Delta_{0,n-1}(t)\cdot\cos\left(\frac{\theta}{2}\right)}{\frac{n}{2}}.$$

■

Proof of Theorem 7.13. Fix a point 0 such that $\Delta_{0,n-1}(0) = \delta$, according to the definition of continuous δ-V-configurations. Since $\cos(\theta(t)/2) \leq 1$, we can bound $\Delta_{0,n-1}{}'(t) \leq \frac{2\cdot\Delta_{0,n-1}(t)}{n}$ (see Lemma 7.29). Thus, it requires time $\mathcal{O}(n)$ until $\Delta_{0,n-1}(t)$ doubles. To increase $\Delta_{0,n-1}(t)$ such that $\Delta_{0,n-1}(t) \geq c$ for an arbitrary constant (less than 1), it requires time $\Omega(n\cdot\log(1/\Delta_{0,n-1}(0))) = \Omega(n\cdot\log(1/\delta))$. ■

The high runtime also explains another crucial property of the MAX-MOB protocol. Remember that an inner robot moves in case either $\|w_i(t)\|_2 = 1$, $\|w_{i+1}(t)\|_2 = 1$ or $\alpha_i(t) < \psi$. Suppose we drop the last assumption and inner robots move only in case either $\|w_i(t)\|_2 = 1$, $\|w_{i+1}(t)\|_2 = 1$. Consequently, we would lose the speed gain obtained by reducing the speed of the outer robots! To see this, observe that in a continuous δ-V-configuration with very small angles θ, the robot $r_{\lceil\frac{n}{2}\rceil}$

moves fast enough such that $O_\ell(t)$ and $O_r(t)$ decrease with constant speed such that immediately $\|w_{\frac{n}{2}}(t)\|_2 < 1$ and $\|w_{\frac{n}{2}+1}(t)\|_2 < 1$ hold. Hence, $r_{\frac{n}{2}}$ stops moving and waits until $O_\ell(t)$ and $O_r(t)$ reach their maximum length again. As the process is continuous, $r_{\frac{n}{2}}$ does not wait until this happens but is slowed down to a speed of $1-\tau$ such that $r_{\frac{n}{2}}$ and the outer robots move with the same speed that results in a runtime depending on δ. To summarize, two aspects in the design of MAX-MOB are crucial for the linear runtime: Slowing down the outer robots to a speed of $1-\tau$ and allowing the inner robots to move at full speed in case they are located at a very small angle.

Since slowing down the outer robots in the continuous time model removes the dependence one δ, one could conjecture that the same approach would work in the discrete time model. Next, we prove this conjecture to be wrong. Consider the protocol $(1-\tau)$-MAX-GTM, in which the outer robots do not move the entire distance to their target point but only $1-\tau$ times the distance they would usually move. The movement of inner robots remains unchanged. The new positions of r_0 and r_{n-1} can be computed as follows:

$$\begin{aligned} p_0(t+1) &= p_0(t) + (1-\tau)\left(\frac{1}{2}\cdot p_1(t) + \frac{1}{2}p_0(t) - \frac{1}{2}\widehat{w}_1(t) - p_0(t)\right) \\ &= \frac{(1+\tau)}{2}\cdot p_0(t) + \frac{(1-\tau)}{2}\cdot p_1(t) - \frac{(1-\tau)}{2}\cdot \widehat{w}_1(t) \\ p_{n-1}(t+1) &= \frac{(1+\tau)}{2}\cdot p_{n-1}(t) + \frac{(1-\tau)}{2}\cdot p_{n-2}(t) + \frac{(1-\tau)}{2}\cdot \widehat{w}_{n-1}(t) \end{aligned}$$

For the vector representation, we obtain the following equations:

$$\begin{aligned} w_1(t+1) &= \frac{1}{2}w_2(t) + \frac{\tau}{2}w_1(t) + \frac{(1-\tau)}{2}\cdot \widehat{w}_1(t) \\ w_{n-1}(t+1) &= \frac{1}{2}\cdot w_{n-2}(t) + \frac{\tau}{2}\cdot w_{n-1}(t) + \frac{(1-\tau)}{2}\cdot \widehat{w}_{n-1}(t) \end{aligned}$$

Similar to Definition 7.1, we can define configurations that have the same behavior under $(1-\tau)$-MAX-GTM as discrete δ-V-configurations under MAX-GTM showing that a speed reduction does not work here.

Definition 7.7 For n even, a *discrete* $(\delta, 1-\tau)$*-V-configuration* is defined by the vectors $w_i(t) = \left(\frac{\delta}{n-1}, \frac{\frac{n}{2}+1-2\cdot(i-1)}{\frac{n}{2}+1}\left(\frac{1-\tau}{1-\tau+\frac{2}{n-2}}\right)\right)$ for all $1 \leq i \leq n-1$.

Note that for $\tau = 0$, discrete δ-V-configurations and discrete $(\delta, 1-\tau)$-V-configurations coincide. Also for $\delta = 0$, discrete $(\delta, 1-\tau)$-V-configurations have a marching chain behavior. However the movement distance per round scales with τ.

Theorem 7.14 Starting in a discrete $(\delta, 1-\tau)$-V-configuration, $(1-\tau)$-MAX-GTM needs at least $\Omega\left(n^2 \cdot \log(1/\delta)\right)$ rounds to achieve an ε-approximation of the max-chain.

The proof is analogous to the proof of Theorem 7.10. First, we state that all vectors $w_i(t)$ are always at least as large as their initial length.

Lemma 7.30 During an execution of $(1-\tau)$-MAX-GTM starting in a discrete $(\delta, 1-\tau)$-V-configuration at time step 0, $\|w_i(t)\|_2 \geq \|w_i(0)\|_2$ for all t and all $1 \leq i \leq n-1$.

Proof. The proof is analogous to the proof of Lemma 7.8. ■

Next, we show that it requires $\Omega\left(n^2 \cdot \log\left(1/\delta\right)\right)$ until the x-coordinate of $w_2(t)$ is larger than a constant.

Lemma 7.31 Assume that we start $(1-\tau)$-MAX-GTM in a discrete $(\delta, 1-\tau)$-V-configuration. We have that $x_1(t) \geq \frac{1}{2}$ after $\Omega\left(n^2 \cdot \log\left(1/\delta\right)\right)$ rounds.

Proof.

$$\begin{aligned} x_1(t+1) &\leq \frac{1}{2}x_2(t) + \frac{\tau}{2} \cdot x_1(t) + \frac{1-\tau}{2 \cdot \frac{1-\tau}{1-\tau+\frac{2}{n-2}}} \cdot x_1(t) \\ &= \frac{1}{2}x_2(t) + \left(\frac{1}{2} + \frac{1}{n-2}\right) \cdot x_1(t) \\ &\leq \left(1 + \frac{1}{n-2}\right) \cdot x_1(t) \end{aligned}$$

The last line is the same formula that has been obtained in the proof of Lemma 7.9 and thus, the same lower bound applies. ■

7.6 Conclusion & Outlook

We defined the MAX-CHAIN-FORMATION problem and presented two protocols, one for the $\mathcal{OBLOT}_1^F$ model and a second one for the $\mathcal{OBLOT}_1^C$ model. The MAX-CHAIN-FORMATION can be seen as an extension of the CHAIN-FORMATION problem, with different roles of the outer robots. In the same way, we also designed the protocols: we took protocols to solve the CHAIN-FORMATION problem in the respective models and added additional movement rules for the outer robots to stretch the chain. Since the outer robots can only observe the position of their direct neighbor, our rules demand that the outer robots move as far as possible away from the direct neighbor without losing connectivity. Compared to the CHAIN-FORMATION protocols, we observed a more complex convergence behavior. The MAX-GTM protocol for the $\mathcal{OBLOT}_1^F$ model converges either to the optimal configuration or to a configuration that keeps moving through the plane forever. We completely characterized this behavior for one-dimensional (all robots' positions are initially collinear) configurations and proved upper and lower runtime bounds of $\Omega(n^2 \cdot \log(1/\varepsilon))$ and $\mathcal{O}(n^2 \cdot \log(n/\varepsilon))$ rounds. For two-dimensional configurations, such a characterization is not known yet and is left for future research. Starting in two-dimensional configurations, the runtime of MAX-GTM can be arbitrarily high, depending on the initial distance δ of the outer robots. More precisely, we could prove a bound of $\Omega(n^2 \cdot \log(1/\delta))$ rounds. As δ can be chosen arbitrarily small, the high runtime follows.

Apart from a complete understanding of the MAX-GTM protocol, it is also worthwhile to consider different protocols. Can we modify the movement of the outer robots such that the undesired marching chains are removed from the configuration space? Moreover, our simulations indicate that a small random perturbation of the input configuration removes all undesired behavior and results in a configuration that converges to the max-chain with high probability. Future research could analyze the impact of random perturbations, underlying our conjecture that there are only a few configurations that converge to the marching chain.

Furthermore, the $\mathcal{LUMI}$ model allows potentially to design faster protocols. Similar ideas as for the ε-HOPPER and ε-2-HOPPER protocols (Chapter 5) could be applied here. Obviously, the merge operation needs to be handled differently as removing robots from the chain would reduce the maximum chain length. Nevertheless, the hop and the shorten operations could remain and the outer robots obtain the role to enlarge certain vectors.

For the $\mathcal{OBLOT}_1^C$ model, we have introduced the MAX-MOB protocol. Compared to MAX-GTM, MAX-MOB has an interesting twist: the outer robots are mostly slowed down by a constant. The speed reduction allows for a time-optimal protocol, requiring $\mathcal{O}(n)$ time. Nevertheless, also the MAX-MOB protocol does not always converge to the max-chain. Instead, some configurations collapse to a single point. Also here, a complete characterization of the input space is left for future

research. Additionally, a protocol that never collapses to a single point would be of interest. There are certainly highly symmetric input configurations that cannot be stretched to a maximum length. Hence, the first step would be to characterize all input configurations for which the problem is impossible to solve and afterward to design a protocol that stretches all remaining configurations properly to a max-chain.

8. The MAX-LINE-FORMATION Problem

In this last chapter, we study the MAX-LINE-FORMATION problem, where the goal is to arrange n robots located in $\mathbb{R}^2$ on a straight line of length $n-1$. While the goal is identical to Chapter 7 (the MAX-CHAIN-FORMATION problem), we consider a different robot model in this chapter: instead of considering a chain topology, we consider the standard connectivity model, where robots can observe all robots within their viewing range and do not have any predefined neighborhoods. Previously (Chapter 7), we have seen that this task is even with the potentially helpful assumption of a chain difficult to establish. In this chapter, we will see that – without a chain – the problem is for oblivious robots ($\mathcal{OBLOT}$) even impossible to solve. We discuss how to enhance the local views of the robots to allow at least solutions that converge to the optimal configuration. Moreover, we show how switching to the $\mathcal{LUMI}$ model allows the robots to solve the problem optimally. The results of this chapter are based on the following publication.

> **2021** (with T. Goette, T. Knollmann and F. Meyer auf der Heide) "The Max-Line-Formation Problem – and new Insights for Gathering and Chain-Formation" In: *Proceedings of the 23rd International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, cf. [25].

8.1 Contribution

We introduce the MAX-LINE-FORMATION problem. The goal is to arrange n robots on a straight line of length $(n-1)$. We start with an impossibility result and prove that there are initial configurations for which the problem cannot be solved deterministically by robots in the $\mathcal{OBLOT}^F$ model with constant sized *circular* viewing and connectivity ranges. In addition, also no protocol that converges to the optimal solution can exist for these configurations. The impossibility result even holds under strong assumptions: fully synchronized robots ($\mathcal{F}$SYNC) that agree on *both* axes of their local coordinate systems.

> **Theorem 8.1** In the $\mathcal{OBLOT}^F$ model, for every constant *circular* connectivity and viewing range, there exists an initial configuration with robots located at distinct positions such that MAX-LINE-FORMATION is unsolvable. Furthermore, no protocol that converges to the optimal configuration can exist for these configurations. This holds even for robots that agree on both axes of their local coordinate systems.

On the positive side, we show that the problem becomes solvable for robots with identical *square* connectivity and viewing ranges. While *square* connectivity and viewing ranges already have been proven to be useful to derive an efficient GATHERING protocol [109], MAX-LINE-FORMATION is the first known problem that can be solved under square viewing ranges but not under *circular* viewing ranges. Our protocols require the robots to agree on *one* axis of their local coordinate systems. We introduce three protocols. The first protocol considers the $\mathcal{OBLOT}$ model and converges to the optimal configuration in $\mathcal{O}(n^2 \cdot \log(n/\varepsilon))$ epochs under the $\mathcal{S}$SYNC scheduler. The analysis idea is based on the *sample variance* of time *inhomogeneous* Markov chains (a concept similar to the *mixing time* of the time-homogeneous case) inspired by [103].

Theorem 8.2 In the $\mathcal{OBLOT}^S_1$ model with *square* viewing ranges and robots that agree on one axis of their local coordinate systems, there exists a protocol such that for every $\varepsilon \in (0,1)$, after $\mathcal{O}(n^2 \cdot \log(n/\varepsilon))$ epochs, the robots have formed a line of length at least $(1-\varepsilon) \cdot (n-1)$.

Afterward, we show that enhancing the robots with the $\mathcal{LUMI}$ model allows us to derive an improved protocol, i.e., the protocol solves the problem exactly while simultaneously improving the runtime. The protocol considers the $\mathcal{F}$SYNC scheduler and solves the problem in $\Theta(n)$ epochs. The runtime is asymptotically optimal. Moreover, the protocol can be implemented with 9 colors. Compared to the $\mathcal{OBLOT}$ protocol, the $\mathcal{LUMI}$ protocol for the $\mathcal{F}$SYNC scheduler makes use of the idea of run sequences, which we have already introduced in Chapter 5. Robots that locally believe that they are endpoints of the line maximize the distance to their closest neighbor and start run sequences to swap the maximized vector along the line.

Theorem 8.3 In the $\mathcal{LUMI}^F_1$ model with *square* viewing ranges and robots that agree on one axis of their local coordinate systems, there exists a protocol such that after $\mathcal{O}(n)$ epochs, the robots have solved the MAX-LINE-FORMATION problem.

With some additional synchronization we have already used in Chapter 5, a combination of the two protocols can solve the problem exactly with the help of lights in $\mathcal{O}(n^2)$ epochs under the $\mathcal{S}$SYNC scheduler[1]. The second part of the protocol uses the same approach with run sequences as the $\mathcal{F}$SYNC protocol. However, under the $\mathcal{S}$SYNC scheduler, our current solution requires more time to arrange all robots in a line initially.

Theorem 8.4 In the $\mathcal{LUMI}^S_1$ model with *square* viewing ranges and robots that agree on one axis of their local coordinate systems, there exists a protocol such that after $\mathcal{O}(n^2)$ epochs, the robots have solved the MAX-LINE-FORMATION problem.

8.2 Model Recap and Preliminaries

Initially, we show for the $\mathcal{OBLOT}^F_c$ model, for a constant $c \geq 1$, that the MAX-LINE-FORMATION problem is unsolvable, even if the robots agree on both axes of their local coordinate systems. For the rest of the chapter, we assume that the robots agree on the x-axis of their local coordinate systems, i.e., all robots have a common understanding of left and right and the coordinate systems of all robots are commonly aligned. However, the robots do not have any agreement on the y-axis, i.e., the understanding of up and down can be different from robot to robot. Moreover, the robots have a square viewing and connectivity range of 2. This means, that robots can observe all other robots within the axis-aligned square of side length 2 centered at their own position. As we also

[1] Note that synchronization in Chapter 5 works even for the $\mathcal{A}$SYNC scheduler. However, some parts of the protocols in this section do not rely on a locally sequential movement. Since the synchronization in Chapter 5 heavily relies on the sequentiality, we cannot use it entirely in this section.

consider a square *connectivity range*, this implies that the distance between two neighboring robots can be at most $\sqrt{2}$ (and thus slightly larger than 1, compared to the previous chapters of this thesis).

Protocol	Model	Dimension	Viewing Range	Orientation	Chain
Section 8.3.3	$\mathcal{OBLOT}^S$	1, 2	2 (square)	one-axis agreement	no
Section 8.4.1	$\mathcal{LUMI}^F$	1, 2	2 (square)	one-axis agreement	no
Section 8.4.4	$\mathcal{LUMI}^S$	1, 2	2 (square)	one-axis agreement	no

Table 8.1: A summary of the most important model details for the MAX-GTM and MAX-MOB protocols.

Problem Statement. The goal of the MAX-LINE-FORMATION problem is to reach a configuration that is connected and there are two robots r_i and r_j that have a distance of $\|p_i(t) - p_j(t)\|_2 = n-1$. The positions of all robots have to be collinear and there is a renaming of the robots such that in the renaming, r_0 and r_{n-1} only observe a single neighbor (r_1 and r_{n-2}) and all other robots r_i observe exactly two neighbors r_{i-1} and r_{i+1}.

Definitions & Notation. For a robot r_i, $r_\ell^i(t)$ denotes the leftmost robot of its neighborhood in round t. The position of $r_\ell^i(t)$ in the local coordinate system of r_i in round t is denoted by $p_\ell^i(t) = (x_\ell^i(t), y_\ell^i(t))$. In case there are multiple such robots, $r_\ell^i(t)$ represents an arbitrary robot of all leftmost ones. Similarly, $r_r^i(t)$ and $p_r^i(t)$ are defined for the rightmost neighbor. Additionally, define $r_+^i(t)$ and $p_+^i(t)$ to be the *closest* neighbor above of r_i and its position in the local coordinate system of r_i. Analogously, $r_-^i(t)$ and $p_-^i(t)$ is defined as the *closest* neighbor below and its position. In case no such robot exists, $r_+^i(t) = r_i$ and $r_-^i(t) = r_i$. For a vector v, we denote by $\widehat{v}$ the normalized vector $\frac{1}{\|v\|_2} v$.

8.3 Results in the $\mathcal{OBLOT}$ Model

The following sections study the MAX-LINE-FORMATION problem in the $\mathcal{OBLOT}$ model. We start with the impossibility result in Section 8.3.1. Afterward, we emphasize why square viewing ranges circumvent the impossibility in Section 8.3.2. Our protocol for the $\mathcal{OBLOT}^S$ model is presented in Section 8.3.3 and analyzed in Section 8.3.4.

8.3.1 Impossibility Result

This section proves that MAX-LINE-FORMATION is unsolvable with constant-sized circular viewing and connectivity ranges in the $\mathcal{OBLOT}_1^F$ model (and thus also for $\mathcal{OBLOT}_1^S$ and $\mathcal{OBLOT}_1^A$).

Theorem 8.1 In the $\mathcal{OBLOT}^F$ model, for every constant *circular* connectivity and viewing range, there exists an initial configuration with robots located at distinct positions such that MAX-LINE-FORMATION is unsolvable. Furthermore, no protocol that converges to the optimal configuration can exist for these configurations. This holds even for robots that agree on both axes of their local coordinate systems.

Proof. Initially, we assume an identical viewing and connectivity range of c. The arguments for viewing ranges that are larger than the connectivity range are analogous. We prove the claim by

contradiction. We assume that there is a protocol $\mathcal{M}$ that can solve the MAX-LINE-FORMATION problem. Next, we derive a combination of 2 initial configurations C_1 and C_2 and prove that if $\mathcal{M}$ can solve the problem starting in C_1, it cannot solve it starting in C_2. The configuration C_1 consists of three robots r_1, r_2 and r_3 at arbitrary (connected) positions. Since $\mathcal{M}$ can solve the problem, there is a time step t_f such that the MAX-LINE-FORMATION problem is solved. Without loss of generality, we assume that r_1 and r_3 are located at the end of the line and $p_1(t_f), p_2(t_f)$ and $p_3(t_f)$ form a line parallel to the y-axis (otherwise we could rename the robots and rotate the following configuration C_2 accordingly). More precisely, $p_1(t_f) - p_2(t_f) = p_2(t_f) - p_3(t_f) = (0,c)$. See Figure 8.1 for a depiction of the effects of $\mathcal{M}$ started in C_1.

The configuration C_2 consists of 7 robots, $r_4, \ldots, r_{10}$ located at the following positions in a global coordinate system (not known to the robots): $p_4(t) = (-c,c), p_5(t) = (-c,0), p_6(t) = (-c,-c), p_7(t) = (0,0), p_8(t) = (c,c), p_9(t) = (c,0)$, and $p_{10}(t) = (c,-c)$. See Figure 8.2 for a visualization of the configuration. In C_2, r_4 can only see r_5 and is located at a distance of c of r_5. Moreover, it holds $p_4(t) - p_5(t) = p_1(t_f) - p_2(t_f)$. Thus, $\mathcal{M}$ is not allowed to move r_4 since $\mathcal{M}$ cannot distinguish r_1 in configuration C_1 after time t_f and r_4 in configuration C_2. By similar arguments, $\mathcal{M}$ is also not allowed to move r_6, r_8 and r_{10}. Hence, the only remaining robots that could be moved by $\mathcal{M}$ are r_5, r_7 and r_9. However, also these robots are not allowed to move. Consider the robot r_5 which is located at a maximum vertical distance of 1 to r_4, r_6 and r_7. No matter where r_5 moves, it loses the connectivity to either r_4 or r_6 as these robots remain at their position. The same arguments hold for r_7 and r_9. It follows that $\mathcal{M}$ cannot solve the problem starting in C_2, which contradicts the assumption.

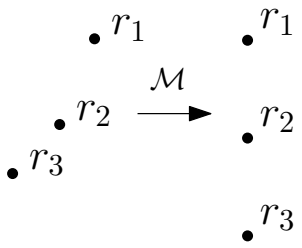

Figure 8.1: $\mathcal{M}$ applied to C_1.

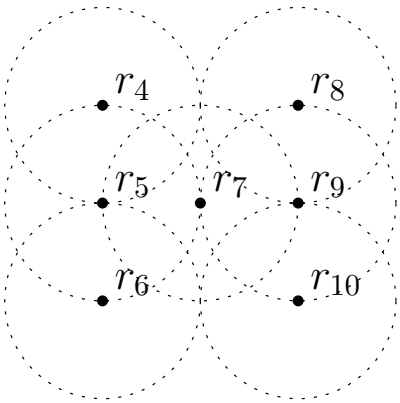

Figure 8.2: The configuration C_2.

■

8.3.2 Intuition about Square Viewing Ranges

Next, we argue why the proof of Theorem 8.1 does not hold when considering *square* viewing and connectivity ranges. We assume that the protocol $\mathcal{M}$ transforms the configuration C_1 into a line that is parallel to the y-axis. Then, also the configuration C_2 is aligned with the y-axis. Still, the robots r_4, r_6, r_8 and r_{10} are not allowed to move. The robots r_5 and r_9, however, gain the possibility to move horizontally. More precisely, r_5 is allowed to move to the right without losing the connectivity to r_4 and r_6 since the complete line segment connecting r_5 and r_7 is contained in the square viewing range of both r_4 and r_6. Similarly, r_9 can move to the left. Consequently, a protocol solving the MAX-LINE-FORMATION with the help of square ranges should arrange the robots on a line parallel to the y-axis. The square ranges are only beneficial in case the local coordinate systems have the same orientation. In case the robots are disoriented, the impossibility result of Section 8.3.1 also holds with square ranges.

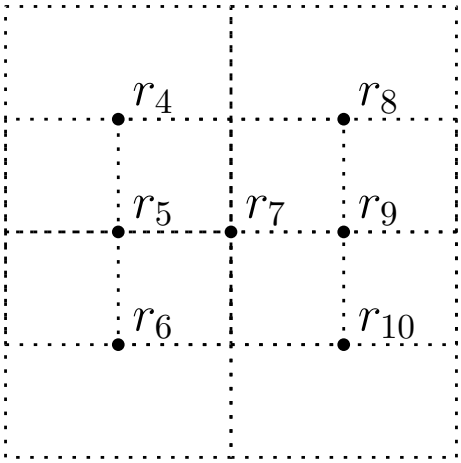

Figure 8.3: The configuration C_2 with square ranges instead of circular ones.

8.3.3 $\mathcal{OBLOT}_1^S$ Protocol with Square Viewing Ranges

The protocol works in two phases. In the first phase, the positions of all robots are arranged on a straight line parallel to the y-axis. Afterward, the line is stretched in the second phase. Since the robots are oblivious and have limited visibility, robots cannot distinguish the phases and act upon their local view. Nevertheless, we will show that there is a round t' such that all robots have joined the second phase and will remain there for the rest of the execution.

Phase 1: A robot r_i whose neighborhood has not yet formed a line parallel to the y-axis moves only if its position is rightmost in its neighborhood. Then, r_i moves horizontally to the x-coordinate of its leftmost neighbor. If another robot already occupies this position, r_i executes a slight vertical movement into the positive (from its local view) y-direction to avoid a collision. More precisely, if the robot is located topmost in its neighborhood, it moves a constant distance upwards. If the robot is not topmost, it determines the value y^i_{min}, the y-coordinate of its closest neighbor to the top. Afterward, it moves $1/3 \cdot y^i_{min}$ upwards. The factor of $1/3$ is essential since the robot with y-coordinate y^i_{min} might do the same movement while having an inverted understanding of up and down. Thereby, a collision between the two robots is avoided.

Phase 2: In the second phase, all robots are located on the same line parallel to the y-axis, which can be seen as a particular case of the MAX-CHAIN-FORMATION problem. Thus, the robots execute the MAX-GTM protocol designed for MAX-CHAIN-FORMATION (Chapter 7): each inner robot (robots that have neighbors in each direction) move to the midpoint between their closest northern and their closest southern neighbor. Outer robots (at the end of the line) have to stretch the line and move as far as possible away from their closest neighbor without losing connectivity. Concretely, outer robots move as follows. Let r_0 be an outer robot and r_1 its closest neighbor and $v(t) = p_0(t) - p_1(t)$. Then, r_0 imagines a virtual robot r_v at the position $p_v(t) = p_0(t) + \widehat{v}(t)$ and moves to $1/2 \cdot p_v(t) + 1/2 \cdot p_1(t)$.

More formally, we define the following set of possibly colliding robots. For a robot r_i, define $C_i(t) = \{r_j \in N_i(t) | x^i_j(t) = 0 \text{ or } x^i_j(t) = x^i_\ell(t)\}$. Now, $r^i_{min} \in C_i(t)$ is the robot with minimal $y^i_{min}(t)$ among all robots with $y^i_{min}(t) > 0$. Thus, r^i_{min} represents the robot lying above of r_i (from r_i's view) that has the smallest y-coordinate among all robots in $C_i(t)$. If no such robot exists, define $y^i_{min} = 1/10$. Algorithm 6 describes the movement of a robot r_i.

Algorithm 6 $\mathcal{OBLOT}$ MAX-LINE-FORMATION

if $x_r^i(t) = 0$ and $x_\ell^i(t) < 0$ **then** ▷ Check if r_i is rightmost but not leftmost
  if no robot is located on $(x_\ell^i(t), 0)$ **then**
    $p_i(t+1) \leftarrow (x_\ell^i(t), 0)$ ▷ r_i can move safely to the left
  else
    $p_i(t+1) \leftarrow (x_\ell^i(t), \frac{1}{3} \cdot y_{min}^i)$ ▷ r_i avoids a collision with a vertical movement
else
  if $x_r^i(t) = 0$ and $x_\ell^i(t) = 0$ **then** ▷ Check if neighbors are collinear
    if $y_+^i(t) = 0$ and $y_-^i(t) < 0$ **then** ▷ Check if r_i is topmost
      $v_-(t) \leftarrow p_-^i(t) - p_i(t)$
      $p_v(t) \leftarrow p_i(t) - \widehat{v}_-(t)$ ▷ Position of virtual robot
      $p_i(t+1) \leftarrow \frac{1}{2}p_-(t) + \frac{1}{2}p_v(t)$
    else if $y_+^i(t) > 0$ and $y_-^i(t) = 0$ **then** ▷ Check if r_i is bottom-most
      $v_+(t) \leftarrow p_+^i(t) - p_i(t)$
      $p_v(t) \leftarrow p_i(t) - \widehat{v}_+(t)$ ▷ Position of virtual robot
      $p_i(t+1) \leftarrow \frac{1}{2}p_+(t) + \frac{1}{2}p_v(t)$
    else
      $p_i(t+1) \leftarrow \frac{1}{2}p_-(t) + \frac{1}{2}p_+(t)$
r_i moves to $p_i(t+1)$

8.3.4 Analysis of the $\mathcal{OBLOT}^S$ Protocol

The following section is dedicated to the proof of Theorem 8.2.

Theorem 8.2 In the $\mathcal{OBLOT}_1^S$ model with *square* viewing ranges and robots that agree on one axis of their local coordinate systems, there exists a protocol such that for every $\varepsilon \in (0,1)$, after $\mathcal{O}(n^2 \cdot \log(n/\varepsilon))$ epochs, the robots have formed a line of length at least $(1-\varepsilon) \cdot (n-1)$.

All following theorems and lemmata refer to an execution of Algorithm 6.

Lemma 8.1 After $\mathcal{O}(n^2)$ epochs, all robots are located in distinct positions on the same vertical line parallel to the y-axis. Moreover, the configuration is connected.

Proof. Initially, at most n distinct x-coordinates that are occupied by robots exist. In every epoch, at least one robot that occupies the rightmost x-coordinate moves to the left as the configuration is connected. This movement does not create any new x-coordinate as the robot moves to the x-coordinate of its leftmost neighbor. Additionally, no robot moves to the right. Hence, after at most n epochs, no robot occupies the rightmost x-coordinate anymore. Thus, after $\mathcal{O}(n^2)$ epochs, all robots are located on the same vertical line by applying the same argument inductively. The connectivity and non-existence of collisions follow from the protocol's description. ■

Now, we can assume that the first phase is completed, and thus all robots are located on the same vertical line. Without loss of generality, we rename the robots such that $y_0(t) \leq y_1(t) \leq \ldots \leq y_{n-1}(t)$. Moreover, we define $w_0(t) = 1$ and $w_i(t) = y_i(t) - y_{i-1}(t)$ for $1 \leq i \leq n-1$. In addition, we define $z_i(t) = w_i(t) - w_0(t)$. The protocol is designed such that $\lim_{t\to\infty} w_i(t) = 1$ for all i. To analyze this behavior, we consider the following function.

$$\Phi(t) = \sum_{i=1}^{n-1} z_i(t)^2.$$

The function $\Phi(t)$ is also known as the *sample variance* [103]. The name comes from a relation to time inhomogeneous Markov chains. Although the protocol is deterministic, the behavior of the vectors $w_i(t)$ can be interpreted as a time inhomogeneous Markov Chain with an absorbing state (since $w_1(t) = 1$ for all t). The main course of our analysis is based on [103], where the authors analyzed a similar behavior in the context of the distributed averaging consensus problem. In this problem, there are n agents, each having a numerical opinion. Every round, an agent updates its opinion to the average opinion of its neighbors. Our application has one important difference: the values $w_i(t)$ do not average but converge to the fixed value $w_1(t)$. Hence, many parts of the proof in [103] have to be reworked and adapted to our application.

First, we derive a formula for $\Phi(t+1)$. To do so, we introduce some additional notation. We define $\tau_i(t) = 1$ if and only if r_i is active in round t. Next, we derive formulas for the vectors $w_i(t+1)$. For each vector, we have to consider 4 cases: Case 1: $\tau_{i-1}(t) = 1$ and $\tau_i(t) = 1$, Case 2: $\tau_{i-1}(t) = 1$ and $\tau_i(t) = 0$, Case 3: $\tau_{i-1}(t) = 0$ and $\tau_i(t) = 1$ and Case 4: $\tau_{i-1}(t) = 0$ and $\tau_i(t) = 0$. In the following, we introduce indicator variables that are equal to 1 if a certain case is fulfilled and otherwise equal to 0. The indicator variables are $\mu_i(t)$, $\mu_i^-(t)$ and $\mu_i^+(t)$. The variable $\mu_i(t) = 1$ if and only if Case 1 is fulfilled and $\mu_i^-(t) = 1$ and $\mu_i^+(t) = 1$ reflect Cases 2 and 3. More formally,

$$\begin{aligned}\mu_i^-(t) &:= \tau_{i-1}(t) \cdot (\tau_{i-1}(t) - \tau_i(t))\\ \mu_i(t) &:= \tau_{i-1}(t) \cdot \tau_i(t)\\ \mu_i^+(t) &:= \tau_i(t) \cdot (\tau_i(t) - \tau_{i-1}(t)).\end{aligned}$$

Next, we introduce the variables $d_i^-(t), d_i(t)$ and $d_i^+(t)$ which express how $\Phi(t+1)$ changes with respect to round t. For the ease of notation, $d_1^-(t) = d_1(t) = 0$ and $d_{n-1}^+(t) = d_{n-1}(t) = 0$ for all t. In every other case, the formulas are

$$\begin{aligned}d_i^-(t) &:= \mu_i^-(t) \cdot (w_i(t) - w_{i-1}(t))^2\\ d_i(t) &:= \mu_i(t) \cdot (w_{i-1}(t) - w_{i+1}(t))^2\\ d_i^+(t) &:= \mu_i^+(t) \cdot (w_i(t) - w_{i+1}(t))^2.\end{aligned}$$

Observe that $d_i^-(t), d_i(t)$ and $d_i^+(t)$ are defined such that at most one of the three terms can be larger than 0 (the other ones are equal to 0). Lastly, define $w_n(t) = w_0(t)$.

Lemma 8.2 For any round t, it holds

$$\Phi(t+1) = \Phi(t) - \frac{1}{4}\sum_{i=1}^{n-1} d_i^-(t) + d_i(t) + d_i^+(t).$$

Proof. Consider a vector $w_i(t)$ with $1 < i < n-1$. Next, we calculate $w_i(t+1)$ based on Cases 1-4.

- Case 1:

$$\begin{aligned}w_i(t+1) &= p_i(t+1) - p_{i-1}(t+1)\\ &= \frac{1}{2}p_{i-1}(t) + \frac{1}{2}p_{i+1}(t) - \frac{1}{2}p_{i-2}(t) - \frac{1}{2}p_i(t)\\ &= \frac{1}{2}p_{i+1}(t) - \frac{1}{2}p_i(t) + \frac{1}{2}p_{i-1}(t) - \frac{1}{2}p_{i-2}(t)\\ &= \frac{1}{2}w_{i+1}(t) + \frac{1}{2}w_{i-1}(t)\end{aligned}$$

- Case 2:

$$w_i(t+1) = p_i(t+1) - p_{i-1}(t+1)$$

$$\begin{aligned}
&= p_i(t) - \frac{1}{2}p_{i-2}(t) - \frac{1}{2}p_i(t) \\
&= \frac{1}{2}p_i(t) - \frac{1}{2}p_{i-2}(t) \\
&= \frac{1}{2}p_i(t) - \frac{1}{2}p_{i-1}(t) + \frac{1}{2}p_{i-1}(t) - \frac{1}{2}p_{i-2}(t) \\
&= \frac{1}{2}w_{i-1}(t) + \frac{1}{2}w_i(t)
\end{aligned}$$

- Case 3:

$$\begin{aligned}
w_i(t+1) &= p_i(t+1) - p_{i-1}(t+1) \\
&= \frac{1}{2}p_{i-1}(t) + \frac{1}{2}p_{i+1}(t) - p_{i-1}(t) \\
&= \frac{1}{2}p_{i+1}(t) - \frac{1}{2}p_{i-1}(t) \\
&= \frac{1}{2}p_{i+1}(t) - \frac{1}{2}p_i(t) + \frac{1}{2}p_i(t) - \frac{1}{2}p_{i-1}(t) \\
&= \frac{1}{2}w_i(t) + \frac{1}{2}w_{i+1}(t)
\end{aligned}$$

- Case 4: $w_i(t+1) = w_i(t)$

The formulas for the boundary vectors $w_1(t)$ and $w_{n-1}(t)$ are slightly different. Case 1': $\tau_0(t) = 1$, $\tau_1(t) = 1$, $\tau_{n-2}(t) = 1, \tau_{n-1}(t) = 1$, Case 2' and 3': $\tau_0(t) = 1$, $\tau_1(t) = 0, \tau_{n-2}(t) = 0$ and $\tau_{n-1}(t) = 1$ or vice versa and Case 4': $\tau_0(t) = 0, \tau_1(t) = 0, \tau_{n-2}(t) = 0$ and $\tau_{n-1}(t) = 0$. We derive the formulas explicitly for $w_1(t+1)$, and the formulas for $w_{n-1}(t+1)$ can be derived analogously.

- Case 1':

$$\begin{aligned}
w_1(t+1) &= p_1(t+1) - p_0(t+1) \\
&= \frac{1}{2}p_0(t) + \frac{1}{2}p_2(t) - \frac{1}{2}p_0(t) + \frac{1}{2} - \frac{1}{2}p_1(t) \\
&= \frac{1}{2}w_0(t) + \frac{1}{2}w_2(t) \\
w_{n-1}(t+1) &= \frac{1}{2}w_{n-2}(t) + \frac{1}{2}w_n(t)
\end{aligned}$$

- Case 2':

$$\begin{aligned}
w_1(t+1) &= p_1(t+1) - p_0(t+1) \\
&= p_1(t) - \frac{1}{2}p_0(t) + \frac{1}{2} - \frac{1}{2}p_1(t) \\
&= \frac{1}{2}w_0(t) + \frac{1}{2}w_1(t) \\
w_{n-1}(t+1) &= \frac{1}{2}w_{n-1}(t) + \frac{1}{2}w_n(t)
\end{aligned}$$

- Case 3':

$$\begin{aligned}
w_1(t+1) &= p_1(t+1) - p_0(t+1) \\
&= \frac{1}{2}p_0(t) + \frac{1}{2}p_2(t) - p_0(t) \\
&= \frac{1}{2}p_2(t) - \frac{1}{2}p_1(t) + \frac{1}{2}p_1(t) - \frac{1}{2}p_0(t) \\
&= \frac{1}{2}w_2(t) + \frac{1}{2}w_1(t) \\
w_{n-1}(t+1) &= \frac{1}{2}w_{n-2}(t) + \frac{1}{2}w_{n-1}(t)
\end{aligned}$$

- Case 4':

$$w_1(t+1) = w_1(t)$$
$$w_{n-1}(t+1) = w_{n-1}(t)$$

Next, we derive a formula for $z_i(t+1)^2$ for $1 < i < n-1$. Observe first $z_i(t)^2 = (w_i(t)-1)^2 = w_i(t)^2 - 2 \cdot w_i(t) + 1$.

- Case 1:

$$\begin{aligned}
z_i(t+1)^2 &= (\frac{1}{2}w_{i-1}(t) + \frac{1}{2}w_{i+1}(t) - 1)^2 \\
&= \frac{1}{4}w_{i-1}(t)^2 + \frac{1}{4}w_{i+1}(t)^2 + \frac{w_{i-1}(t)\cdot w_{i+1}(t)}{2} - w_{i-1}(t) - w_{i+1}(t) + 1 \\
&= \frac{1}{4}z_{i-1}(t)^2 + \frac{1}{4}z_{i+1}(t)^2 + \frac{w_{i-1}(t)\cdot w_{i+1}(t)}{2} - \frac{1}{2}w_{i-1}(t) - \frac{1}{2}w_{i+1}(t) + \frac{1}{2} \quad (8.1) \\
&= \frac{1}{2}z_{i-1}(t)^2 + \frac{1}{2}z_{i+1}(t)^2 - \frac{1}{4}w_{i-1}(t)^2 - \frac{1}{4}w_{i+1}(t)^2 + \frac{w_{i-1}(t)\cdot w_{i+1}(t)}{2} \quad (8.2) \\
&= \frac{1}{2}z_{i-1}(t)^2 + \frac{1}{2}z_{i+1}(t)^2 - \frac{1}{4}\cdot(w_{i-1}(t) - w_{i+1}(t))^2
\end{aligned}$$

Equation (8.2) can be derived by adding $\frac{1}{4}w_{i-1}(t)^2 + \frac{1}{4}w_{i+1}(t)^2$ on both sides, rearranging the right side to $\frac{1}{2}z_{i-1}(t)^2 + \frac{1}{2}z_{i+1}(t)^2 + \frac{w_{i-1}(t)\cdot w_{i+1}(t)}{2}$ and subtracting $\frac{1}{4}w_{i-1}(t)^2 + \frac{1}{4}w_{i+1}(t)^2$ on both sides.

- Case 2:

$$\begin{aligned}
z_i(t+1)^2 &= \frac{1}{4}w_{i-1}(t)^2 + \frac{1}{4}w_i(t)^2 + \frac{w_{i-1}(t)\cdot w_i(t)}{2} - w_{i-1}(t) - w_i(t) + 1 \\
&= \frac{1}{4}z_{i-1}(t)^2 + \frac{1}{4}z_i(t)^2 + \frac{w_{i-1}(t)\cdot w_i(t)}{2} - \frac{1}{2}w_{i-1}(t) - \frac{1}{2}w_i(t) + \frac{1}{2} \\
&= \frac{1}{2}z_{i-1}(t)^2 + \frac{1}{2}z_i(t)^2 - \frac{1}{4}\cdot(w_{i-1}(t) - w_i(t))^2
\end{aligned}$$

- Case 3:

$$\begin{aligned}
z_i(t+1)^2 &= \frac{1}{4}w_i(t)^2 + \frac{1}{4}w_{i+1}(t)^2 + \frac{w_i(t)\cdot w_{i+1}(t)}{2} - w_i(t) - w_{i+1}(t) + 1 \\
&= \frac{1}{4}z_i(t)^2 + \frac{1}{4}z_{i+1}(t)^2 + \frac{w_i(t)\cdot w_{i+1}(t)}{2} - \frac{1}{2}w_i(t) - \frac{1}{2}w_{i+1}(t) + \frac{1}{2} \\
&= \frac{1}{2}z_i(t)^2 + \frac{1}{2}z_{i+1}(t)^2 - \frac{1}{4}\cdot(w_i(t) - w_{i+1}(t))^2
\end{aligned}$$

- Case 4:

$$z_i(t+1)^2 = z_i(t)^2$$

Similar formulas can be derived for $z_1(t+1)^2$ and $z_{n-1}(t+1)^2$.

- Case 1':

$$\begin{aligned}
z_1(t+1) &= \left(\frac{1}{2}w_0(t) + \frac{1}{2}w_2(t) - 1\right)^2 \\
&= \left(\frac{1}{2}w_2(t) - \frac{1}{2}\right)^2 \\
&= \frac{1}{4}w_2(t)^2 - \frac{w_2(t)}{2} + \frac{1}{4} \\
&= \frac{1}{4}z_2(t) \\
z_{n-1}(t+1) &= \frac{1}{4}z_{n-2}(t)
\end{aligned}$$

- Case 2':

$$\begin{aligned} z_1(t+1) &= \left(\frac{1}{2}w_0(t) + \frac{1}{2}w_1(t) - 1\right)^2 \\ &= \left(\frac{1}{2}w_1(t) - \frac{1}{2}\right)^2 \\ &= \frac{1}{4}z_1(t)^2 \\ z_{n-1}(t+1) &= \frac{1}{4}z_{n-1}(t)^2 \end{aligned}$$

- Case 3':

$$\begin{aligned} z_1(t+1) &= \left(\frac{1}{2}w_2(t) + \frac{1}{2}w_1(t) - 1\right)^2 \\ &= \frac{1}{2}z_1(t)^2 + \frac{1}{2}z_2(t)^2 - \frac{1}{4}\cdot(w_1(t) - w_2(t))^2 \\ z_{n-1}(t+1) &= \frac{1}{2}z_{n-2}(t)^2 + \frac{1}{2}z_{n-1}(t) - \frac{1}{4}\cdot(w_{n-2}(t) - w_{n-1}(t))^2 \end{aligned}$$

- Case 4':

$$\begin{aligned} w_1(t+1) &= w_1(t) \\ w_{n-1}(t+1) &= w_{n-1}(t) \end{aligned}$$

Based on the previous formulas, we derive a form each $z_i(t+1)^2$ that reflects all cases.

$$\begin{aligned} z_1(t+1)^2 &= \frac{\tau_0(t)}{2}\cdot z_1(t)^2 + \mu_i^+(t)\cdot z_1(t)^2 + \frac{\tau_1(t)}{2}\cdot z_2(t)^2 \\ &\quad + (1-\tau_0(t))\cdot(1-\tau_1(t))\cdot z_1(t)^2 - \frac{1}{4}d_1^+(t) \\ z_i(t+1)^2 &= \frac{\tau_{i-1}(t)}{2}\cdot z_{i-1}(t)^2 + \mu^-(t)\cdot z_i(t)^2 + \mu^+(t)\cdot z_i(t)^2 + \frac{\tau_i(t)}{2}\cdot z_{i+1}(t)^2 \\ &\quad + (1-\tau_{i-1}(t))\cdot(1-\tau_i(t))\cdot z_i(t)^2 - \frac{1}{4}d_i(t) - \frac{1}{4}d_i^-(t) - \frac{1}{4}d_i^+(t) \\ z_{n-1}(t+1)^2 &= \frac{\tau_{n-1}(t)}{2}\cdot z_{n-1}(t)^2 + \mu_i^-(t)\cdot z_{n-2}(t)^2 + \frac{\tau_{n-2}(t)}{2}\cdot z_{n-2}(t)^2 \\ &\quad + (1-\tau_{n-2}(t))\cdot(1-\tau_{n-1}(t))\cdot z_{n-1}(t)^2 - \frac{1}{4}d_{n-1}^-(t) \end{aligned}$$

When focusing only on the $z_i(t)^2$ terms, we observe that $\tau_i(t) = 1$ $(1 < i < n-1)$ adds $\frac{1}{2}z_{i+1}(t)^2$ to $z_i(t+1)$ and $\frac{1}{2}z_i(t)^2$ to $z_{i+1}(t+1)$. Moreover, $\tau_i(t) = 0$ $(1 < i < n-1)$ adds $\frac{1}{2}z_i(t)^2$ to $z_i(t+1)$ and $\frac{1}{2}z_{i+1}(t)^2$ to $z_{i+1}(t+1)$. Hence, no matter if a robot is active or not, the same $z_i(t)^2$ summands are added to $\Phi(t+1)$. We can make similar observations for $\tau_1(t)$ and $\tau_{n-1}(t)$. All in all, we obtain the following formula for $\Phi(t+1)$:

$$\begin{aligned} \Phi(t+1) &= \sum_{i=1}^{n-1} z_i(t)^2 - \frac{1}{4}\cdot\sum_{i=1}^{n-1}\left(d_i^-(t) + d_i(t) + d_i^+(t)\right) \\ &= \Phi(t) - \frac{1}{4}\cdot\sum_{i=1}^{n-1}\left(d_i^-(t) + d_i(t) + d_i^+(t)\right) \end{aligned}$$

■

Based on the formula for $\Phi(t+1)$ (Lemma 8.2), we derive a bound on the change of $\Phi(t)$ between two *epochs*. Define $w_{\pi_1}(t_{e_k}), \ldots, w_{\pi_{n-1}}(t_{e_k})$ to be the values $w_i(t_{e_k})$ sorted from largest to smallest with ties broken arbitrarily.

Lemma 8.3 For any epoch k,

$$\Phi(t_{e_k}) - \Phi(t_{e_{k+1}}) \geq \frac{1}{4} \sum_{i=1}^{n-1} \left(w_{\pi_i}(t_{e_k}) - w_{\pi_{i+1}}(t_{e_k}) \right)^2 .$$

Proof. By Lemma 8.2, we obtain

$$\Phi(t_{e_k}) - \Phi(t_{e_{k+1}}) \geq \frac{1}{4} \cdot \sum_{t=t_{e_k}}^{t_{e_{k+1}}} \sum_{i=1}^{n-1} \left(d_i^-(t) + d_i(t) + d_i^+(t) \right).$$

The first part of the proof deals with finding a lower bound for any $d_i^-(t) + d_i(t) + d_i^+(t)$ given that at least one of the terms is larger than 0 (at most one of the three terms is positive). The lower bound, however, depends on the sorted sequence $w_{\pi_1}(t), \ldots, w_{\pi_{n-1}}(t)$. Since we lose much structure due to the sorting, some definitions are needed. Let $\pi : \mathbb{N} \to \mathbb{N}$ be the permutation that maps each index i of the vectors w_i to an index $\pi(i)$ such that the sequence by the indices $\pi(i)$ is sorted. In other words, $\pi : \mathbb{N} \to \mathbb{N}$ is the function that maps the indices of $w_1(t_{e_k}), \ldots, w_{n-1}(t_{e_k})$ into the sorted sequence $w_{\pi_1}(t_{e_k}), \ldots, w_{\pi_{n-1}}(t_{e_k})$ and π^{-1} its inverse. Hence, π_x ist the index of the x-th element of the sorted sequence $w_{\pi_1}(t_{e_k}), \ldots, w_{\pi_{n-1}}(t_{e_k})$. More precisely, for instance $\pi(i) = \pi_f$ if and only if $w_i(t_{e_k}) = w_{\pi_f}(t_{e_k})$. Next, we define by $d_{\pi_m}(t)$ the $d_i(t)$'s of the sorted sequence. More formally, $d_{\pi_m}(t) = d_{\pi^{-1}(\pi_m)}(t), d^-_{\pi^{-1}(\pi_m)}(t)$ and $d^+_{\pi^{-1}(\pi_m)}(t)$. Furthermore, define $\sigma_{m,i,j}(t) = 1$ if and only if one of the following three cases is fulfilled:

1. $d_{\pi_m}(t) > 0$ and $\pi(\pi^{-1}(\pi_m) - 1) = \pi_i$ and $\pi(\pi^{-1}(\pi_m) + 1) = \pi_j$ or vice versa
2. $d^-_{\pi_m}(t) > 0$ and $\pi(\pi^{-1}(\pi_m) - 1) = \pi_i$ and $\pi(\pi^{-1}(\pi_m)) = \pi_j$ or vice versa
3. $d^+_{\pi_m}(t) > 0$ and $\pi(\pi^{-1}(\pi_m)) = \pi_i$ and $\pi(\pi^{-1}(\pi_m) + 1) = \pi_j$ or vice versa

Due to the sorting, we lose the nice property that only neighboring $w_i(t)$'s are involved in $d_{\pi_m}(t)$, $d^-_{\pi_m}(t)$ and $d^+_{\pi_m}(t)$. For instance in case $d_{\pi_m}(t) > 0$ we cannot conclude that $w_{\pi_m - 1}(t)$ and $w_{\pi_m + 1}(t)$ ar involved. Thus, intuitively, $\sigma_{m,i,j}(t) = 1$ if and only if $d_{\pi_m}(t), d^-_{\pi_m}(t)$ or a $d^+_{\pi_m}(t)$ is larger than 0 and both $w_{\pi_i}(t)$ and $w_{\pi_j}(t)$ are involved.

Next, define t_ℓ ($1 \leq \ell \leq n$) to be the first round larger than or equal to t_{e_k} such that there exists an index π_ℓ as well as three indices π_i, π_j and π_m ($\pi_i \neq \pi_j$ but $\pi_i = \pi_m$, $\pi_j = \pi_m$ or $\pi_m = \pi_\ell$ might hold) with $\pi_i \leq \pi_\ell < \pi_j$ (or π_i and π_j are exchanged) and $\sigma_{m,i,j}(t) = 1$. In other words, t_ℓ denotes the first round in which the values $w_{\pi_1}(t_{e_k}), \ldots, w_{\pi_\ell}(t_{e_k})$ and $w_{\pi_{\ell+1}}(t_{e_k}), \ldots, w_{\pi_{n-1}}(t_{e_k})$ influence each other. By influencing each other, we mean that $w_{\pi_m}(t+1) = \frac{1}{2} w_{\pi_i}(t) + \frac{1}{2} w_{\pi_j}(t)$, since either $d_{\pi_m}(t) > 0, d^-_{\pi_m}(t) > 0$ or $d^+_{\pi_m}(t) > 0$.[2]

For all $t \in \{t_{e_k}, \ldots, t_{e_{k+1}}\}$ let $L(t) = \{\ell \,|\, t_\ell = t\}$, i.e., $L(t)$ represents all indices ℓ at time t such that the two sets $\{w_{\pi_1}(t_{e_k}), \ldots w_{\pi_\ell}(t_{e_k})\}$ and $\{w_{\pi_{\ell+1}}(t_{e_k}), \ldots, w_{\pi_{n-1}}(t_{e_k})\}$ influence each other for the first time.

Now, we define all pairs of indices π_i, π_j at time t such that there exists a π_ℓ and a π_m with $\pi_i \leq \pi_\ell < \pi_j$ and $\sigma_{m,i,j}(t) = 1$: $C_{\ell,m}(t) = \{\{\pi_i, \pi_j\} \,|\, \pi_i \leq \pi_\ell < \pi_j \text{ and } \sigma_{m,i,j}(t) = 1\}$. Lastly, define for fixed i, j and t: $F_{ij}(t) = \{\ell \in L(t) \,|\, \text{there is an index } \pi_m \text{ with } \{\pi_i, \pi_j\} \in C_{\ell,m}(t)\}$.

Fix some π_i and π_j with $\pi_i < \pi_j$ and a round t such that $|F_{ij}(t)| > 0$. Let $F_{ij}(t) = \{\ell_1, \ldots, \ell_k\}$ sorted in increasing order. Since $\ell_1 \in L(t)$, it holds by definition that there exists no round $t' \in [t_{e_k}, \ldots, t]$ and no index π_m with $\pi_i \leq \pi_{\ell_1} < \pi_j$ and $\sigma_{m,i,j}(t') = 1$. It follows $w_{\pi_i}(t) \geq w_{\pi_{\ell_1}}(t_{e_k})$

[2] In the context of averaging consensus each index $1, \ldots, n$ corresponds to a node in the graph. Thus, the index ℓ can be interpreted as a cut in the graph and the time t_ℓ as the first time with communication across the cut represented by ℓ.

(since $w_{\pi_i}(t)$ was so far only influenced by elements of the set $w_{\pi_1}(t_{e_k}),\dots,w_{\pi_{\ell_1}}(t_{e_k})$ which are all larger or equal to $w_{\pi_{\ell_1}}(t_{e_k})$). Similarly, one can argue $w_{\pi_j}(t) \leq w_{\pi_{\ell_k+1}}(t_{e_k})$. Hence, we can conclude

$$w_{\pi_i}(t) - w_{\pi_j}(t) \geq w_{\pi_{\ell_1}}(t_{e_k}) - w_{\pi_{\ell_k+1}}(t_{e_k}) \geq \sum_{\pi_\ell \in F_{ij}(t)} \left(w_{\pi_\ell}(t_{e_k}) - w_{\pi_{\ell+1}}(t_{e_k})\right).$$

The last line directly leads to

$$(w_{\pi_i}(t) - w_{\pi_j}(t))^2 \geq \sum_{\pi_\ell \in F_{ij}(t)} \left(w_{\pi_\ell}(t_{e_k}) - w_{\pi_{\ell+1}}(t_{e_k})\right)^2.$$

The second part of the proof now deals with fixing a round t and finding a lower bound for $\sum_{i=1}^{n-1}\left(d_i^-(t)+d_i(t)+d_i^+(t)\right)$.

$$\begin{aligned}
\sum_{i=1}^{n-1}\left(d_i^-(t)+d_i(t)+d_i^+(t)\right) &= \sum_{(\pi_m,\pi_i,\pi_j):\sigma_{m,i,j}(t)=1} \left(w_{\pi_i}(t) - w_{\pi_j}(t)\right)^2 \\
&\geq \sum_{(\pi_m,\pi_i,\pi_j):\sigma_{m,i,j}(t)=1} \sum_{\pi_\ell \in F_{ij}(t)} \left(w_{\pi_\ell}(t_{e_k}) - w_{\pi_{\ell+1}}(t_{e_k})\right)^2 \\
&\geq \sum_{\pi_\ell \in L(t)} \left(w_{\pi_\ell}(t_{e_k}) - w_{\pi_{\ell+1}}(t_{e_k})\right)^2.
\end{aligned}$$

Lastly, we plug all insights together to conclude the proof.

$$\begin{aligned}
\Phi(t_{e_k}) - \Phi(t_{e_{k+1}}) &\geq \frac{1}{4} \cdot \sum_{t=t_{e_k}}^{t_{e_{k+1}}} \sum_{i=1}^{n-1} \left(d_i^-(t)+d_i(t)+d_i^+(t)\right) \\
&\geq \frac{1}{4} \cdot \sum_{t=t_{e_k}}^{t_{e_{k+1}}} \sum_{\pi_\ell \in L(t)} \left(w_{\pi_\ell}(t_{e_k}) - w_{\pi_{\ell+1}}(t_{e_k})\right)^2. \\
&= \frac{1}{4} \sum_{\pi_\ell=1}^{n-1} \left(w_{\pi_\ell}(t_{e_k}) - w_{\pi_{\ell+1}}(t_{e_k})\right)^2.
\end{aligned}$$

The last line follows since each robot moves at least once per epoch. ■

Based on Lemma 8.3, a lower bound on the relative change is derived.

Lemma 8.4 Suppose that $\Phi(t_{e_k}) > 0$. Then, $\frac{\Phi(t_{e_k})-\Phi(t_{e_{k+1}})}{\Phi(t_{e_k})} \geq \frac{1}{8n^2}$.

Proof. Lemma 8.3 leads to

$$\begin{aligned}
\frac{\Phi(t_{e_k}) - \Phi(t_{e_{k+1}})}{\Phi(t_{e_k})} &\geq \frac{1}{4} \frac{\sum_{\pi_\ell=1}^{n-1} \left(w_{\pi_\ell}(t_{e_k}) - w_{\pi_{\ell+1}}(t_{e_k})\right)^2}{\sum_{\pi_\ell=1}^{n-1} \left(w_{\pi_\ell}(t_{e_k}) - 1\right)^2.} \\
&= \frac{1}{4} \frac{\sum_{\pi_\ell=1}^{n-1} \left(w_{\pi_\ell}(t_{e_k}) - w_{\pi_{\ell+1}}(t_{e_k})\right)^2}{\sum_{\pi_\ell=1}^{n-1} \left(w_{\pi_\ell}(t_{e_k}) - w_{\pi_1}(t_{e_k})\right)^2.}
\end{aligned}$$

The second line follows since $w_0(t) = 1$ for all t and thus $w_{\pi_1}(t_{e_k}) = 1$. Observe that the right-hand side does not change if we multiply each $w_{\pi_i}(t_{e_k})$ with the same constant. Additionally, it also does not change if we add the same constant to each $w_{\pi_i}(t_{e_k})$. Hence, we can assume without loss of generality that $\sum_{\pi_\ell=1}^{n-1} w_{\pi_\ell}(t_{e_k}) = 0$ and that $\sum_{\pi_\ell=1}^{n} \left(w_{\pi_\ell}(t_{e_k}) - w_{\pi_1}(t_{e_k})\right)^2 = 1$ and obtain

$$\frac{\Phi(t_{e_k}) - \Phi(t_{e_{k+1}})}{\Phi(t_{e_k})} \geq \frac{1}{4} \min_{\substack{w_1 \geq w_2, \ldots, \geq w_{n-1} \\ \sum_i w_i = 0 \\ \sum_i (w_i - w_1)^2 = 1}} \sum_{i=1}^{n-1} (w_i - w_{i+1})^2$$

The assumption $\sum_i (w_i - w_1)^2 = 1$ implies that the average value of all $(w_i - w_1)^2$ is $\frac{1}{n}$ and hence there is at least some j with $|w_j - w_1| \geq \frac{1}{\sqrt{n}}$. As a consequence, either $|w_1| \geq \frac{1}{2\cdot\sqrt{n}}$ or $|w_j| \geq \frac{1}{2\cdot\sqrt{n}}$. Without loss of generality, we assume that $|w_1| \geq \frac{1}{2\sqrt{n}}$ and moreover assume $w_1 > 0$. The case $w_1 < 0$ can be handled by multiplying each w_i with -1 and sorting the elements in descending order.

Now, define $u_i = w_i - w_{i+1}$ for $i < n-1$ and $u_{n-1} = 0$. It holds $u_i \geq 0$ for all i and $\sum_{i=1}^{n-1} w_i = w_0 - w_{n-1}$. Since at least one $w_1 \geq \frac{1}{2\sqrt{n}}$ and the $\sum_i w_i = 0$, we can conclude $u_{n-1} < 0$ and thus $\sum_i u_i \geq \frac{1}{2\sqrt{n}}$.

As a final step, we obtain

$$\frac{\Phi(t_{e_k}) - \Phi(t_{e_{k+1}})}{\Phi(t_{e_k})} \geq \frac{1}{4} \min_{u_i \geq 0, \sum_i u_i \geq 1/(2\sqrt{n})} \sum_{i=1}^{n-1} w_i^2.$$

The solution of the minimization problem is $u_i = \frac{1}{2\cdot n^{3/2}}$ for each i. Hence,

$$\frac{\Phi(t_{e_k}) - \Phi(t_{e_{k+1}})}{\Phi(t_{e_k})} \geq \frac{1}{4} \cdot \frac{1}{2 \cdot n^2} = \frac{1}{8n^2}.$$

■

Lemma 8.5 After $\mathcal{O}\left(n^2 \cdot \log\left(n/\varepsilon\right)\right)$ epochs, it holds $\sum_{i=1}^{n-1} w_i(t) \geq (1-\varepsilon) \cdot (n-1)$.

Proof. Fix any epoch e_k. By Lemma 8.4, we obtain $\Phi(t_{e_{k+1}}) \leq (1 - \frac{1}{8n^2}) \cdot \Phi(t_{e_k})$ and thus $\Phi(t_{e_{k+x}}) \leq \left(1 - \frac{1}{8n^2}\right)^x \cdot \Phi(t_{e_k})$. Observe that $(1-y)^x \leq e^{-y\cdot x}$ where e denotes Euler's number. Thus, choosing $x \geq 8n^2 \cdot \ln(\frac{1}{r})$ yields $\Phi(t_{e_{k+x}}) \leq r \cdot \Phi(t_{e_k})$. Since $\Phi(t_{e_k}) < n-1$, $r \leq \frac{\varepsilon}{n-1}$ leads to $\Phi(t_{e_{k+x}}) \leq \varepsilon$ and thus $\sum_{i=1}^{n-1} w_i(t) \geq (1-\varepsilon) \cdot (n-1)$. ■

8.4 Results in $\mathcal{LUMI}$ Model

In this section, we mainly derive a protocol that solves MAX-LINE-FORMATION *optimally* (no convergence) in the $\mathcal{LUMI}_1^F$ model. The protocol (Algorithm 7) achieves an optimal runtime of $\Theta(n)$ rounds and is the topic of Section 8.4.1. Afterward, we argue briefly in Section 8.4.4, how a combination of the $\mathcal{OBLOT}^S$ and the $\mathcal{LUMI}^F$ protocol solves the MAX-LINE-FORMATION also optimally in the $\mathcal{LUMI}^S$ model at the cost of a slightly larger runtime.

8.4.1 $\mathcal{LUMI}_1^F$ Protocol with Square Viewing Ranges

Similar to the $\mathcal{OBLOT}^S$ protocol, the protocol works in two phases: In the first phase, all robots are arranged on a straight line parallel to the y-axis, and in the second phase, the line is stretched until it has maximal length. Note that these are *global* phases not known to the robots. Since the robots can only act based on their local view, some robots might behave according to phase 1 while other robots already move according to phase 2. Nevertheless, we will prove that there is a point in time such that globally phase 1 is completed such that in the following, all robots move according to phase 2. Compared to the $\mathcal{OBLOT}$ protocol (Section 8.3.3), the protocol uses different core ideas in both phases. In the first phase, all robots (instead of only the rightmost ones of their

neighborhood) move to the left – this is necessary to achieve a linear speedup of the first phase. The second phase makes use of lights to implement run sequences, as we have already seen in Chapter 5. To focus on the core ideas, we initially present a variant of the protocol in which the robots still move to the left during the second phase. More precisely, after a linear number of rounds, the first phase ends, and the robots form a line parallel to the y-axis that continuously moves a distance of 1 to the left. Simultaneously, the robots stretch the line until it has maximal length. However, the line structure is always maintained such that MAX-LINE-FORMATION is solved finally and remains solved (although the line keeps moving to the left). Moving continuously to the left can be removed from the protocol with some additional effort; an intuition is given in Section 8.4.3.

Phase 1: *All* robots move as far as possible (regarding their neighborhood) to the left: each robot r_i moves to the x-coordinate $x_r^i(t)-1$. Again, collision avoidance has to be ensured. While moving to $x_r^i(t)-1$, the robot r_i could collide with every robot located on its local x-axis. The robot r_i executes a vertical movement to avoid a collision. Based on the ordering of neighbors on the local x-axis, r_i gets assigned a unique y-coordinate as follows: Define $Y_i(t) = \{r_j \in N_i(t) | y_j^i(t) = 0\}$ and let $x_{\pi_1}(t), x_{\pi_2}(t), \ldots, x_{\pi_{|Y_i(t)|}}(t)$ be the x-coordinates of robots in $Y_i(t)$ in increasing order. Additionally, let $k_i(t) \in \{1, \ldots, |Y_i(t)|\}$ denote the position of $x_i(t)$ in the sorted sequence $x_{\pi_1}(t), x_{\pi_2}(t), \ldots, x_{\pi_{|Y_i(t)|}}(t)$. Furthermore, define $y_{min}^i(t)$ to be the minimal $y_j^i(t)$ of all $y_j^i(t) > 0$ of robots $r_j \in N_i(t)$. If no such robot exists, define $y_{min}^i(t) = \frac{1}{10}$ (any constant of size at most 1 works). Then, r_i gets assigned the y-coordinate $\frac{k_i(t)-1}{|Y_i(t)|} \cdot \frac{1}{3} y_{min}^i(t)$. The factor $\frac{k_i(t)-1}{|Y_i(t)|}$ is unique for every robot on the local x-axis and the factor of $\frac{1}{3}$ is needed to prevent a collision with other robots that execute the same collision avoidance.

Phase 2: For the second phase, lights are used. Without loss of generality, we assume that the robots are ordered along the y-axis, i.e., $y_0(t) \geq \cdots \geq y_{n-1}(t)$. The core idea is to implement run sequences started at r_0 and r_{n-1} with the help of lights. Assume that a run sequence starts in round t. Then, only r_0 and r_{n-1} move. In round $t+1$, only r_1 and r_{n-2} move and so on. A new run sequence is started every three rounds. The realization with the lights works as follows. The first required light ℓ_c with color set $C_c = \{0,1,2\}$ is used as a round counter. Every round, all robots increment their light ℓ_c. Whenever $\ell_c = 2$ holds, both r_0 and r_{n-1} activate a light ℓ_{run} with $C_{run} = \{0,1\}$ (the light is either active or inactive). Thus, in the next round, it holds $\ell_c = 0$ and both r_0 and r_{n-1} detect an active light ℓ_{run}. Both r_0 and r_{n-1} now execute a movement (see below). Additionally, they deactivate the light ℓ_{run} and activate a light ℓ_{prev} with color set $C_{prev} = \{0,1\}$ to remember the movement. Simultaneously, the robots r_1 and r_{n-2} observe a neighbor on the y-axis with active light ℓ_{run} (r_0 and r_{n-1}). Additionally, neither r_1 nor r_{n-2} has activated ℓ_{prev}. Hence, the robots activate ℓ_{run} to continue the run sequence. In the next round, r_0 and r_{n-1} observe a neighbor with active light ℓ_{run} but do not activate their own light ℓ_{run} since ℓ_{prev} is active.

Robots that have a run state (the light ℓ_{run} is active) move as follows. In case r_0 has a run state and not r_1 ($n > 2$), r_0 moves at a distance of 1 vertically away from r_1. More formally, $p_0(t+1) = (x_r^0(t) - 1, -\frac{y_1^0(t)}{|y_1^0(t)|})$ (remember that in this variant the robots move also in phase 2 to the left). Similar, r_{n-1} moves away from r_{n-2} at a distance of 1. In case a robot r_i has a run state that came from r_{i-1} (r_{i-1} has activated ℓ_{prev} and r_i has activated ℓ_{run}) and r_{i+1} does not have a run state, r_i moves in vertical distance 1 away from $r+1$: $p_i(t+1) = (x_r^i(t) - 1, -\frac{y_{i+1}^i(t)}{|y_{i+1}^i(t)|})$. Lastly, in case two neighboring robots have a run, for instance, r_i and r_{i+1} have activated ℓ_{run} both move only a vertical distance of $\frac{1}{2}$ away from each other: $p_i(t+1) = (x_r^i(t) - 1, -\frac{y_{i+1}^i(t)}{2|y_{i+1}^i(t)|})$. The handling of the lights and the corresponding movement is depicted in Figure 8.4.

Algorithm 7 $\mathcal{LUMI}_1^F$ Protocol executed from the local view of r_i

if all neighbors are located on the y-axis **then**
  if $r_i = r_+^i(t)$ or $r_i = r_-^i(t)$ **then**
    if $\ell_{run} = 1$ **then** ▷ $\ell_{run} = 1$ implies $\ell_c = 0$
      $\ell_{run} \leftarrow 0; \ell_{prev} \leftarrow 1$
      $r_c \leftarrow$ closest neighbor on y-axis
      if r_c has activated ℓ_{run} **then** ▷ Special case $n = 2$
        $p_i(t+1) \leftarrow (x_r^i(t) - 1, -\frac{1}{2 \cdot |y_c(t)|} \cdot y_c(t))$ ▷ Move distance of $\frac{1}{2}$
      else
        $p_i(t+1) \leftarrow (x_r^i(t) - 1, -\frac{1}{|y_c(t)|} \cdot y_c(t))$ ▷ Move distance of 1
    else
      if $\ell_{prev} = 1$ **then** ▷ Deactivate ℓ_{prev}
        $\ell_{prev} \leftarrow 0$
      else
        if $\ell_c = 2$ **then** ▷ Start new run sequence
          $\ell_{run} \leftarrow 1$
      $p_i(t+1) \leftarrow (x_r^i(t) - 1, 0))$
  else
    if $\ell_{run} = 1$ **then**
      $\ell_{run} \leftarrow 0$, $\ell_{prev} \leftarrow 1$
      if closest neighbor above and below have set $\ell_{run} = 0$ **then**
        $r_c \leftarrow$ closest neighbor with $\ell_{prev} = 0$
        $p_i(t+1) \leftarrow (x_r^i(t) - 1, -\frac{1}{|y_c(t)|} \cdot y_c(t))$
      else
        $r_c \leftarrow$ neighbor with $\ell_{run} = 1$
        $p_i(t+1) \leftarrow (x_r^i(t) - 1, -\frac{1}{2 \cdot |y_c(t)|} \cdot y_c(t))$
    else
      if $\ell_{prev} = 1$ **then**
        $\ell_{prev} \leftarrow 0$
      else
        if closest neighbor above or below has set $\ell_{run} = 1$ **then**
          $\ell_{run} \leftarrow 1$
      $p_i(t+1) \leftarrow (x_r^i(t) - 1, 0))$
else
  $\{\ell_{run}, \ell_{prev}\} \leftarrow 0$ ▷ Deactivate lights if neighborhood is not in phase 2
  if $|Y_i(t)| > 0$ **then**
    $p_i(t+1) \leftarrow (x_r^i(t) - 1, \frac{k_i(t)-1}{|Y_i(t)|} \cdot \frac{1}{3} y_{min}^i(t))$
  else
    $p_i(t+1) \leftarrow (x_r^i(t) - 1, 0)$
$\ell_c \leftarrow \ell_c + 1$
r_i moves to $p_i(t+1)$

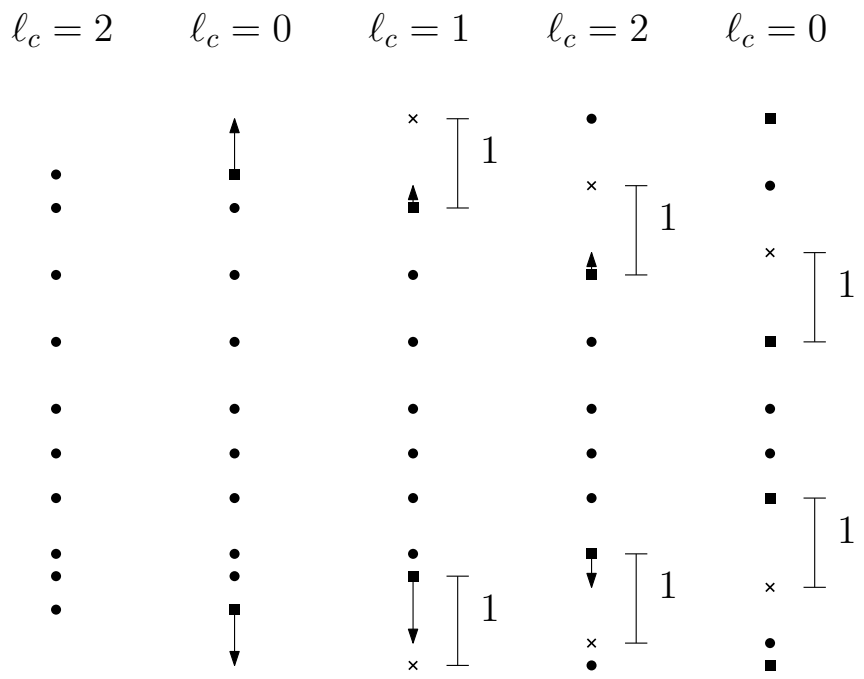

Figure 8.4: A square (cross) depicts a robot with active light ℓ_{run} (ℓ_{prev}). Time proceeds from left to right. In the first line, it holds $\ell_c = 2$ for all robots. In this round, the topmost and the bottom-most robot activate ℓ_{run}. In the next round ($\ell_c = 0$), these two robots move at a distance of 1 of their neighbor (depicted by an arrow) and additionally deactivate ℓ_{run} while activating ℓ_{prev}. Afterward, the movement continues.

8.4.2 Analysis of the $\mathcal{LUMI}_1^F$ Protocol

In the analysis, we prove that after a linear number of rounds, the first phase ends. Moreover, it is proven that as soon as phase 2 is reached, the robots remain in phase 2 (following from the protocol's description).

Lemma 8.6 After $\mathcal{O}(n)$ epochs, all robots are located in distinct positions on the same vertical line parallel to the y-axis. Moreover, the configuration is connected.

Proof. The connectivity and collision avoidance can be directly concluded from the protocol's description. To prove the linear runtime, define $x_{max}(t)$ to be the maximal x-coordinate in the global coordinate system. Furthermore, define $k_{max}(t)$ to be the number of robots with $x_i(t) \in (x_{max}(t)-1, x_{max}(t)]$. Observe that every robot r_i with $x_i(t) \in (x_{max}(t)-1, x_{max}(t)]$ moves such that $x_i(t+1) \in (x_{max}(t)-2, x_{max}(t)-1]$. Since the configuration is always connected, there must have been a robot r_j with $x_j(t) \in (x_{max}(t)-2, x_{max}(t)-1]$ that cannot leave the interval. It follows $k_{max}(t+1) \geq k_{max}(t)+1$ Thus, after $\mathcal{O}(n)$ rounds, all robots have x-coordinates in an interval of size at most 1. Fix one round t and assume that all robots have x-coordinates in an interval of size at most 1. Consider the robot r_{max} with globally maximal x-coordinate. None of its neighbors can see a robot that has a larger x-coordinate, and thus, all robots of $N_{max}(t)$ are collinear in round $t+1$. Furthermore, all of these robots still have the globally largest x-coordinate in round $t+1$. Now, consider the topmost robot $r_j \in N_{max}(t)$. It follows that all robots of $N_j(t+1)$ have a smaller or equal x-coordinate and cannot see any other robot with a larger x-coordinate. Thus, in round $t+2$ all robots in $N_{max}(t)$ and $N_j(t+1)$ are collinear. The same argument holds for the bottom-most robot. Applying the same argument inductively yields that all robots are collinear after $\mathcal{O}(n)$ rounds. ■

Afterward, the run sequences of the second phase are analyzed. The first run sequence ensures that after $\mathcal{O}(n)$ rounds, the robots $r_{\lfloor n/2 \rfloor - 1}$ and $r_{\lfloor n/2 \rfloor}$ have a vertical distance of 1. The second run

sequence ensures the same both for $r_{\lfloor n/2 \rfloor - 2}$ and $r_{\lfloor n/2 \rfloor - 1}$ as well as $r_{\lfloor n/2 \rfloor + 1}$ and $r_{\lfloor n/2 \rfloor + 2}$. Hence, after $\mathcal{O}(n)$ run sequences, the line reaches maximal length. Since each 3 rounds, a new run sequence is started, and the linear runtime follows.

Theorem 8.5 After $\mathcal{O}(n)$ epochs, the robots have solved MAX-LINE-FORMATION.

Proof. By Lemma 8.6 all robots are collinear on a line parallel to the y-axis after $\mathcal{O}(n)$ epochs. It remains to prove the linear runtime until the optimal configuration is formed. Rename the robots such that r_0 is the topmost robot and r_{n-1} is the bottom-most robot. Define $w_i(t) = y_i(t) - y_{i-1}(t)$. Both r_0 and r_{n-1} activate their light ℓ_{run} every 3 rounds and ensure $w_1(t) = w_{n-1}(t) = 1$. In round $t+1$ it holds $w_2(t+1) = w_{n-2}(t+1) = 1$ and so on. Assume n to be even (the arguments for odd n are analogous). After $\frac{n}{2} - 1$ rounds, the movement meets at the two robots $r_{n/2}$ and $r_{n/2+1}$ and they move such that $y_{n/2+1}(t + \frac{n}{2} - 1) = 1$ holds. The next two movements ensure $y_{n/2}(t + \frac{n}{2} + 2) = y_{n/2+2}(t + \frac{n}{2} + 2) = 1$ and so on. Since every third round a new movement is started, the optimal configuration is reached in $\mathcal{O}(n)$ rounds. ■

The protocol can be implemented in the classical $\mathcal{LUMI}$ model with a single light having 9 colors. Observe that no robot ever activates the lights ℓ_{prev} and ℓ_{run} at the same time. Thus, for each robot, the following always holds: either ℓ_{prev}, ℓ_{run} or none of both are activated. Additionally, each robot counts rounds with the light ℓ_c requiring 3 colors. Hence, the total number of required colors is 9: 3 colors of ℓ_c, each combined with 3 possible cases for the lights ℓ_{run} and ℓ_{prev}.

8.4.3 Adjusted $\mathcal{F}$SYNC Protocol

The $\mathcal{F}$SYNC protocol presented in Section 8.4 is – to keep the pseudocode comprehensible – designed such that the robots still move to the left after MAX-LINE-FORMATION is already solved. In this section, we explain how 3 additional lights help to remove this behavior to design a protocol that forms a stationary line.

Observe first that in Algorithm 7, two run sequences can only be located at two neighboring robots in case the protocol is already in phase 2. Otherwise, at least one robot observes that its neighborhood is not yet aligned parallel to the y-axis, and the corresponding run sequence is stopped (line 31 in Algorithm 7). We use this observation as follows: As soon as two run sequences meet at neighboring robots, the two robots activate a light ℓ_{final} to store this information. Robots with an active light ℓ_{final} do not move to the left anymore. Additionally, robots that observe a robot in their neighborhood that has activated ℓ_{final}, activate their light ℓ_{final}. Hence, after $\mathcal{O}(n)$ rounds, all robots have activated ℓ_{final}. While propagating ℓ_{final}, it might, however, happen that some robots move to the left while other robots remain stationary (due to the limited visibility). See Figure 8.5 for a depiction of such a case. To rebuild the line shape again run sequences at robots with active light ℓ_{final} behave slightly differently. The vertical movement is identical to before. The horizontal movement changes: instead of moving to the left, a robot moves a distance of 1 to the right if it is leftmost in its neighborhood, and there is at least one robot in a horizontal distance of 1 to the right. Finally, the robots align again on the initial line (before activating ℓ_{final}) and MAX-LINE-FORMATION gets solved after $\mathcal{O}(n)$ rounds.

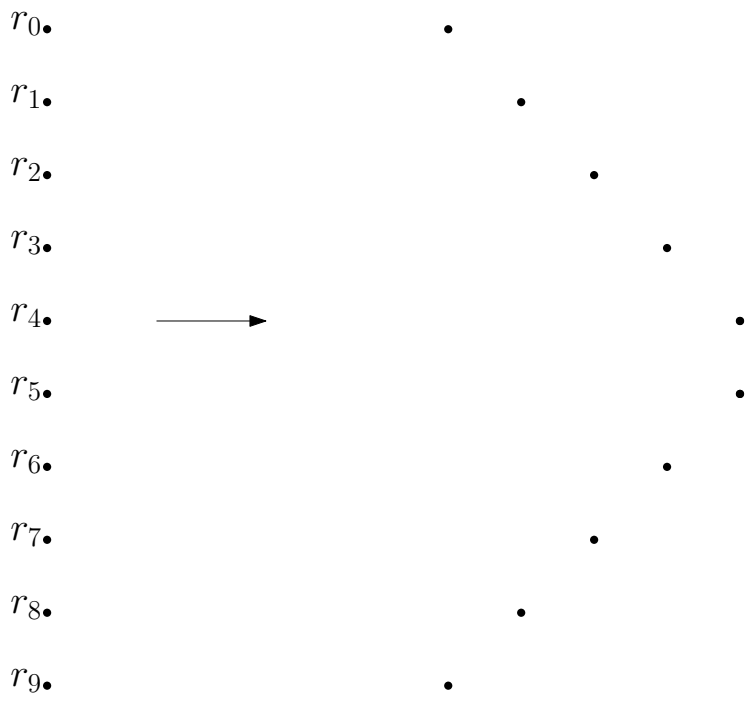

Figure 8.5: To the left, the robots are arranged on a straight line parallel to the y-axis. After some time, two run sequences meet in the middle at r_4 and r_5 that activate their light ℓ_{final}. In the next round, r_3 and r_6 activate their light ℓ_{final} and so on. However, r_0, r_1, r_2, r_3, r_6, r_7, r_8 and r_9 move a distance of 1 to the left before r_3 and r_6 become stationary. Similarly, $r_0, \ldots, r_2$ and $r_7, \ldots, r_9$ move a distance of 1 further to the left than r_2 and r_5 and so on. The final configuration might look like it is depicted to the right.

8.4.4 High-level $\mathcal{LUMI}_1^S$ Protocol

The first phase of the $\mathcal{LUMI}_1^S$ protocol is identical to the first phase of the $\mathcal{OBLOT}$ protocol (Section 8.3.3): Each robot that is rightmost in its neighborhood moves horizontally to the x-coordinate of its leftmost neighbor. In case this position is already occupied, a slight vertical movement is used to avoid collisions. The main idea of the second phase is the sequential movement (run sequence) of Section 8.4. Due to the $\mathcal{S}$SYNC scheduler, an additional synchronization procedure needs to be added. In $\mathcal{F}$SYNC, a robot with active light ℓ_{run} can always be sure that the neighbors observe and adapt the light. Since only a subset of robots is active in every round in $\mathcal{S}$SYNC, the light ℓ_{run} might not be seen, and thus, the run sequence stops. To overcome this, we add a synchronization done with the light ℓ_c. In contrast to the $\mathcal{F}$SYNC protocol, the robots do not increment the light in every round they become active. Instead, each run sequence gets associated with a color of the light ℓ_c. More precisely, the main idea is as follows. Assume that the robots have already formed a line parallel to the y-axis. Moreover, we rename the robots such that $y_1(t) \leq y_2(t) \leq \cdots \leq y_{n-1}(t)$. Additionally, assume the configuration is well-initialized, i.e., all robots have set $\ell_c = 0$. We describe the procedure from the view of r_0, it works analogously for r_{n-1}. We denote by $\ell_i(r_j)$ the color of r_j's light ℓ_i in round t (the time parameter is omitted for readability). As soon as r_1 is activated, it observes $\ell_c(r_2) = \ell_{prev}(r_2) = \ell_{run}(r_2) = 0$. Then, r_1 activates ℓ_{run}. As soon as r_1 wakes up again, it executes its movement (it moves at a distance of 1 of r_2), deactivates ℓ_{run}, activates ℓ_{prev} and increments ℓ_c such that $\ell_c = 1$. In the future, r_1 will only deactivate ℓ_{prev} in case it detects $\ell_c(r_2) = 1$ (indicating that r_2 has taken over the run sequence). Hence, as soon as r_2 is activated and detects $\ell_c(r_1) = \ell_{prev}(r_1) = 1$ and $\ell_c(r_3) = \ell_{prev}(r_3) = \ell_{run}(r_3) = 0$, it will activate ℓ_{run}. Upon its next activation, r_2 executes its movement, deactivates ℓ_{run}, activates ℓ_{prev} and increments ℓ_c. As soon as two neighboring robots have activated ℓ_{run} both move at a distance of $\frac{1}{2}$ away from each other and stop the run sequence (exactly as in Algorithm 7). This way, the run

sequences proceed along the line. To conclude, a robot r_j only takes over a run sequence from its neighbor r_{j-1} in case $\ell_c(r_{j-1}) = \ell_c(r_j) + 1$. Additionally, r_j will only deactivate ℓ_{prev} as soon as $\ell_c(r_{j-1}) \geq \ell(r_j)$ and $\ell_c(r_{j+1}) = \ell_c(r_j)$. Note that it might happen due to the limited visibility that some run sequences already start while the first phase is not completed. Hence, at the beginning of phase 2, not all robots might be initialized with the same color of the light ℓ_c. In case a robot detects such a violation (e.g., the next robot that should take over the light ℓ_{run} has a larger value of ℓ_c), the usual movement is not executed. Instead, simply the light ℓ_c is incremented. Hence, for each constant number of run sequences, the light of one more robot is well-initialized, and the protocol adjusts the colors of the lights ℓ_c in a self-stabilizing manner. All in all, the first phase has a runtime of $\mathcal{O}(n^2)$ epochs (Lemma 8.1), the second phase is after $\mathcal{O}(n)$ epochs well-initialized (arguments above) and completed after additional $\mathcal{O}(n)$ epochs (Theorem 8.5). The runtime of $\mathcal{O}(n^2)$ epochs follows. All in all, we obtain the following theorem.

Theorem 8.4 In the $\mathcal{LUMI}_1^S$ model with *square* viewing ranges and robots that agree on one axis of their local coordinate systems, there exists a protocol such that after $\mathcal{O}(n^2)$ epochs, the robots have solved the MAX-LINE-FORMATION problem.

8.5 Conclusion & Outlook

In this chapter, we considered the MAX-LINE-FORMATION problem, which is similar to the MAX-CHAIN-FORMATION problem but does not consider a chain of robots. In general, we saw that the problem is unsolvable in the $\mathcal{OBLOT}$ model, even under quite strong robot capabilities: robots that agree on both axes of their local coordinate system and operate under the $\mathcal{F}$SYNC scheduler. Afterward, we extended the local views of the robots slightly by considering square viewing ranges instead of circular ones. This slight increase in the viewing ranges allowed us to derive three protocols. The first one was for the $\mathcal{OBLOT}^S$ model and can converge to the final configuration while requiring a runtime of $\mathcal{O}\left(n^2 \cdot \log n/\varepsilon\right)$ epochs. The second and third protocols were designed for the $\mathcal{LUMI}$ model. The protocol for the $\mathcal{LUMI}_1^F$ solves the problem optimally in $\Theta(n)$ rounds. Lastly, the protocol for the $\mathcal{LUMI}_1^S$ model solves the problem optimally but requires $\mathcal{O}(n^2)$ epochs.

Based on these three protocols, we observe several future research questions. Throughout the entire thesis (for the CHAIN-FORMATION, the MAX-CHAIN-FORMATION and the MAX-LINE-FORMATION problems) we have seen $\mathcal{OBLOT}$ protocols that only converge to the optimal solution but never reach it. Also for the MAX-LINE-FORMATION problem, it would be interesting to determine whether a protocol for the $\mathcal{OBLOT}$ model exists that solves the problem exactly. As the CHAIN-FORMATION, the MAX-CHAIN-FORMATION and the MAX-LINE-FORMATION problem are quite similar in this behavior, a solution for one protocol (either an impossibility result or a positive result) would help to find a solution for the remaining problems.

Moreover, the impossibility result for the MAX-LINE-FORMATION problem only holds for the $\mathcal{OBLOT}$ model. It seems possible that there is a protocol to solve the MAX-LINE-FORMATION problem with circular viewing and connectivity ranges in the $\mathcal{LUMI}$ model. Consider again the configuration C_2 we used in the impossibility result for the $\mathcal{OBLOT}$ model (Figure 8.1). Under the $\mathcal{LUMI}$ model, the robot r_9 still cannot move immediately. However, it can communicate with help of lights that it observes the robot r_7 to the left. Doing so ensures that both r_8 and r_{10} notice that they have to move as the global configuration cannot be final. Thus, the robots get a chance to form a line. Still, lots of additional algorithmic challenges need to be solved which are left for future research. Lastly, our protocols for the $\mathcal{LUMI}_1^F$ and $\mathcal{LUMI}_1^S$ exhibit a runtime gap. The $\mathcal{LUMI}_1^F$ protocol is significantly faster ($\mathcal{O}(n)$ vs. $\mathcal{O}(n^2)$). Up to now, it is unclear whether a faster protocol for the $\mathcal{S}$SYNC scheduler can be designed. Furthermore, solving the problem under the $\mathcal{A}$SYNC scheduler remains open. The second phase of the protocols can be synchronized even under the $\mathcal{A}$SYNC schedulers with the ideas presented in Chapters 5 and 6 as also run sequences with a locally

sequential movement are used. However, in the first phase of all protocols, the movement is not sequential and hence it is unclear whether the techniques work. We conjecture that the techniques can be applied and lead to a correct protocol that solves the MAX-LINE-FORMATION problem but the runtime increases since the local synchronization might involve more than a constant number of robots in the worst case.

Bibliography

[1] *Distributed Computing by Mobile Entities, Current Research in Moving and Computing*, volume 11340 of *Lecture Notes in Computer Science*. Springer, 2019.

[2] Sebastian Abshoff, Andreas Cord-Landwehr, Matthias Fischer, Daniel Jung, and Friedhelm Meyer auf der Heide. Gathering a Closed Chain of Robots on a Grid. In *Proceedings of the 2016 IEEE International Symposium on Parallel & Distributed Processing (IPDPS)*, pages 689–699. IEEE, May 2016.

[3] Noa Agmon and David Peleg. Fault-Tolerant Gathering Algorithms for Autonomous Mobile Robots. *SIAM Journal on Computing (SICOMP)*, 36(1):56–82, 2006.

[4] Jonathan Amos. Nereus deep sea sub 'implodes' 10km-down. https://www.bbc.com/news/science-environment-27374326, May 2014. Accessed: 2023-03-08.

[5] Hideki Ando, Yoshinobu Oasa, Ichiro Suzuki, and Masafumi Yamashita. Distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Transactions on Robotics and Automation*, 15(5):818–828, 1999.

[6] Hideki Ando, Ichiro Suzuki, and Masafumi Yamashita. Formation and agreement problems for synchronous mobile robots with limited visibility. In *Proceedings of the 10th International Symposium on Intelligent Control (ISIC)*, pages 453–460. IEEE, 1995.

[7] Cédric Auger, Zohir Bouzid, Pierre Courtieu, Sébastien Tixeuil, and Xavier Urbain. Certified Impossibility Results for Byzantine-Tolerant Mobile Robots. In *Stabilization, Safety, and Security of Distributed Systems (SSS)*, Lecture Notes in Computer Science, pages 178–190. Springer, 2013.

[8] Baruch Awerbuch. Complexity of network synchronization. *Journal of the ACM (JACM)*, 32(4):804–823, 1985.

[9] Lali Barriere, Paola Flocchini, Eduardo Mesa-Barrameda, and Nicola Santoro. Uniform scattering of autonomous mobile robots in a grid. In *Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing (IPDPS)*, pages 1–8. IEEE, 2009.

[10] S. Bhagat, S. Gan Chaudhuri, and K. Mukhopadhyaya. Fault-tolerant gathering of asynchronous oblivious mobile robots under one-axis agreement. *Journal of Discrete Algorithms*, 36:50–62, 2016.

[11] Subhash Bhagat and Krishnendu Mukhopadyaya. Fault-tolerant Gathering of Semi-synchronous Robots. In *Proceedings of the 18th International Conference on Distributed Computing and Networking (ICDCN)*, pages 1–10. ACM, 2017.

[12] Mainak Biswas, Saif Rahaman, Moumita Mondal, and Sruti Gan Chaudhuri. Multiple Uniform Circle Formation by Fat Robots Under Limited Visibility. In *Proceedings of the 24th International Conference on Distributed Computing and Networking (ICDCN)*, pages 311–317. ACM, 2023.

[13] Kaustav Bose, Ranendu Adhikary, Manash Kumar Kundu, and Buddhadeb Sau. Arbitrary pattern formation on infinite grid by asynchronous oblivious robots. *Theoretical Computer Science (TCS)*, 815:213–227, 2020.

[14] Zohir Bouzid, Shantanu Das, and Sébastien Tixeuil. Gathering of Mobile Robots Tolerating Multiple Crash Faults. In *Proceedings of the 2013 IEEE 33rd International Conference on Distributed Computing Systems (ICDCS)*, pages 337–346. IEEE, 2013.

[15] Zohir Bouzid, Maria Gradinariu Potop-Butucaru, and Sébastien Tixeuil. Byzantine Convergence in Robot Networks: The Price of Asynchrony. In *Principles of Distributed Systems (OPODIS)*, Lecture Notes in Computer Science, pages 54–70. Springer, 2009.

[16] Zohir Bouzid, Maria Gradinariu Potop-Butucaru, and Sébastien Tixeuil. Optimal Byzantine-resilient Convergence in Unidimensional Robot Networks. *Theoretical Computer Science (TCS)*, 411(34-36):3154–3168, 2010.

[17] Andrew D. Bowen, Dana R. Yoerger, Chris Taylor, Robert McCabe, Jonathan Howland, Daniel Gomez-Ibanez, James C. Kinsey, Matthew Heintz, Glenn McDonald, Donald B. Peters, John Bailey, Eleanor Bors, Tim Shank, Louis L. Whitcomb, Stephen C. Martin, Sarah E. Webster, Michael V. Jakuba, Barbara Fletcher, Chris Young, James Buescher, Patricia Fryer, and Samuel Hulme. Field trials of the Nereus hybrid underwater robotic vehicle in the challenger deep of the Mariana Trench. In *OCEANS 2009*, pages 1–10. IEEE, 2009.

[18] Quentin Bramas and Sébastien Tixeuil. The Random Bit Complexity of Mobile Robots Scattering. In *Ad-Hoc, Mobile, and Wireless Networks (ADHOC-NOW)*, Lecture Notes in Computer Science, pages 210–224. Springer, 2015.

[19] Quentin Bramas and Sébastien Tixeuil. Wait-Free Gathering Without Chirality. In *Structural Information and Communication Complexity (SIROCCO)*, Lecture Notes in Computer Science, pages 313–327. Springer, 2015.

[20] Philipp Brandes, Bastian Degener, Barbara Kempkes, and Friedhelm Meyer auf der Heide. Energy-efficient strategies for building short chains of mobile robots locally. *Theoretical Computer Science (TCS)*, 509:97–112, 2013.

[21] Michael Braun. Local gathering of mobile robots in three dimensions. Master's thesis, Paderborn University, 2022.

[22] Michael Braun, Jannik Castenow, and Friedhelm Meyer auf der Heide. Local Gathering of Mobile Robots in Three Dimensions. In *Structural Information and Communication Complexity (SIROCCO)*, Lecture Notes in Computer Science, pages 63–79. Springer, 2020.

[23] Kevin Buchin, Paola Flocchini, Irina Kostitsyna, Tom Peters, Nicola Santoro, and Koichi Wada. On the Computational Power of Energy-Constrained Mobile Robots: Algorithms and Cross-Model Analysis. In *Structural Information and Communication Complexity (SIROCCO)*, Lecture Notes in Computer Science, pages 42–61. Springer, 2022.

[24] Jannik Castenow, Matthias Fischer, Jonas Harbig, Daniel Jung, and Friedhelm Meyer auf der Heide. Gathering Anonymous, Oblivious Robots on a Grid. *Theoretical Computer Science (TCS)*, 815:289–309, 2020.

[25] Jannik Castenow, Thorsten Götte, Till Knollmann, and Friedhelm Meyer auf der Heide. The Max-Line-Formation Problem. In *Stabilization, Safety, and Security of Distributed Systems (SSS)*, Lecture Notes in Computer Science, pages 289–304. Springer, 2021.

[26] Jannik Castenow, Jonas Harbig, Daniel Jung, Peter Kling, Till Knollmann, and Friedhelm Meyer auf der Heide. A Unifying Approach to Efficient (Near)-Gathering of Disoriented Robots with Limited Visibility. In *Proceedings of the 26th International Conference on*

Principles of Distributed Systems (OPODIS), volume 253 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 15:1–15:25. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023.

[27] Jannik Castenow, Jonas Harbig, Daniel Jung, Till Knollmann, and Friedhelm Meyer auf der Heide. Gathering a Euclidean Closed Chain of Robots in Linear Time. In *Algorithms for Sensor Systems (ALGOSENSORS)*, Lecture Notes in Computer Science, pages 29–44. Springer, 2021.

[28] Jannik Castenow, Jonas Harbig, Daniel Jung, Till Knollmann, and Friedhelm Meyer auf der Heide. Gathering a Euclidean closed chain of robots in linear time and improved algorithms for chain-formation. *Theoretical Computer Science (TCS)*, 939(1):261–291, 2023.

[29] Jannik Castenow, Peter Kling, Till Knollmann, and Friedhelm Meyer auf der Heide. A Discrete and Continuous Study of the Max-Chain-Formation Problem. In *Stabilization, Safety, and Security of Distributed Systems (SSS)*, Lecture Notes in Computer Science, pages 65–80. Springer, 2020.

[30] Jannik Castenow, Peter Kling, Till Knollmann, and Friedhelm Meyer auf der Heide. A discrete and continuous study of the Max-Chain-Formation problem. *Information and Computation*, 285:104877, 2022.

[31] George Chrystal. On the problem to construct the minimum circle enclosing n given points in a plane. *Proceedings of the Edinburgh Mathematical Society*, 3:30–33, 1884.

[32] Serafino Cicerone, Alessia Di Fonso, Gabriele Di Stefano, and Alfredo Navarra. MOBLOT: Molecular Oblivious Robots. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 350–358. International Foundation for Autonomous Agents and Multiagent Systems, 2021.

[33] Serafino Cicerone, Alessia Di Fonso, Gabriele Di Stefano, and Alfredo Navarra. Molecular Robots with Chirality on Grids. In *Algorithmics of Wireless Networks (ALGOSENSORS)*, Lecture Notes in Computer Science, pages 45–59. Springer, 2022.

[34] Serafino Cicerone, Alessia Di Fonso, Gabriele Di Stefano, and Alfredo Navarra. Arbitrary pattern formation on infinite regular tessellation graphs. *Theoretical Computer Science (TCS)*, 942:1–20, 2023.

[35] Mark Cieliebak, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Distributed Computing by Mobile Robots: Gathering. *SIAM Journal on Computing (SICOMP)*, 41(4):829–879, 2012.

[36] Reuven Cohen and David Peleg. Convergence Properties of the Gravitational Algorithm in Asynchronous Robot Systems. *SIAM Journal on Computing (SICOMP)*, 34(6):1516–1528, 2005.

[37] Reuven Cohen and David Peleg. Convergence of Autonomous Mobile Robots with Inaccurate Sensors and Movements. *SIAM Journal on Computing (SICOMP)*, 38(1):276–302, 2008.

[38] Reuven Cohen and David Peleg. Local spreading algorithms for autonomous robot systems. *Theoretical Computer Science (TCS)*, 399(1):71–82, 2008.

[39] Andreas Cord-Landwehr, Bastian Degener, Matthias Fischer, Martina Hüllmann, Barbara Kempkes, Alexander Klaas, Peter Kling, Sven Kurras, Marcus Märtens, Friedhelm Meyer auf der Heide, Christoph Raupach, Kamil Swierkot, Daniel Warner, Christoph Weddemann, and Daniel Wonisch. A New Approach for Analyzing Convergence Algorithms for Mobile

Robots. In *Automata, Languages and Programming (ICALP)*, Lecture Notes in Computer Science, pages 650–661. Springer, 2011.

[40] Andreas Cord-Landwehr, Matthias Fischer, Daniel Jung, and Friedhelm Meyer auf der Heide. Asymptotically Optimal Gathering on a Grid. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 301–312. ACM, 2016.

[41] Shantanu Das, Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Masafumi Yamashita. The Power of Lights: Synchronizing Asynchronous Robots Using Visible Bits. In *Proceedings of the 2012 IEEE 32nd International Conference on Distributed Computing Systems (ICDCS)*, pages 506–515. IEEE, 2012.

[42] Shantanu Das, Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Masafumi Yamashita. Autonomous mobile robots with lights. *Theoretical Computer Science (TCS)*, 609:171–184, 2016.

[43] Shantanu Das, Paola Flocchini, Nicola Santoro, and Masafumi Yamashita. On the computational power of oblivious robots: Forming a series of geometric patterns. In *Proceedings of the 29th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 267–276. ACM, 2010.

[44] Joshua J. Daymude, Kristian Hinnenthal, Andréa W. Richa, and Christian Scheideler. Computing by Programmable Particles. In *Distributed Computing by Mobile Entities: Current Research in Moving and Computing*, Lecture Notes in Computer Science, pages 615–681. Springer, 2019.

[45] Joshua J. Daymude, Andréa W. Richa, and Christian Scheideler. The Canonical Amoebot Model: Algorithms and Concurrency Control. In *Proceedings of the 35th International Symposium on Distributed Computing (DISC)*, volume 209 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

[46] Jean-Lou De Carufel and Paola Flocchini. Fault-induced dynamics of oblivious robots on a line. *Information and Computation*, 271:104478, 2020.

[47] Xavier Défago, Maria Gradinariu, Stéphane Messika, and Philippe Raipin-Parvédy. Fault-Tolerant and Self-stabilizing Mobile Robots Gathering. In *Distributed Computing (DISC)*, Lecture Notes in Computer Science, pages 46–60. Springer, 2006.

[48] Xavier Défago and Samia Souissi. Non-uniform circle formation algorithm for oblivious mobile robots with convergence toward uniformity. *Theoretical Computer Science (TCS)*, 396(1):97–112, 2008.

[49] Bastian Degener, Barbara Kempkes, Peter Kling, and Friedhelm Meyer auf der Heide. Linear and Competitive Strategies for Continuous Robot Formation Problems. *ACM Transactions on Parallel Computing (TOPC)*, 2(1):2:1–2:18, 2015.

[50] Bastian Degener, Barbara Kempkes, Tobias Langner, Friedhelm Meyer auf der Heide, Peter Pietrzyk, and Roger Wattenhofer. A tight runtime bound for synchronous gathering of autonomous robots with limited visibility. In *Proceedings of the 23rd Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 139–148. ACM, 2011.

[51] Mattia D'Emidio, Daniele Frigioni, and Alfredo Navarra. Synchronous Robots vs Asynchronous Lights-Enhanced Robots on Graphs. *Electronic Notes in Theoretical Computer Science*, 322:169–180, 2016.

[52] Zahra Derakhshandeh, Shlomi Dolev, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Amoebot - a new model for programmable matter. In *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 220–222. ACM, 2014.

[53] Giuseppe Antonio Di Luna and Giovanni Viglietta. Robots with Lights. In *Distributed Computing by Mobile Entities: Current Research in Moving and Computing*, Lecture Notes in Computer Science, pages 252–277. Springer, 2019.

[54] Yoann Dieudonné and Franck Petit. Swing Words to Make Circle Formation Quiescent. In *Structural Information and Communication Complexity (SIROCCO)*, Lecture Notes in Computer Science, pages 166–179. Springer, 2007.

[55] Yoann Dieudonné and Franck Petit. Squaring the Circle with Weak Mobile Robots. In *Algorithms and Computation (ISAAC)*, Lecture Notes in Computer Science, pages 354–365. Springer, 2008.

[56] Casey Dreier. Cost of Perseverance. https://www.planetary.org/space-policy/cost-of-perseverance. Accessed: 2023-03-08.

[57] Miguel Duarte, Vasco Costa, Jorge Gomes, Tiago Rodrigues, Fernando Silva, Sancho Moura Oliveira, and Anders Lyhne Christensen. Evolution of Collective Behaviors for a Real Swarm of Aquatic Surface Robots. *PLOS ONE*, 11(3):e0151834, 2016.

[58] Ayan Dutta, Sruti Gan Chaudhuri, Suparno Datta, and Krishnendu Mukhopadhyaya. Circle Formation by Asynchronous Fat Robots with Limited Visibility. In *Distributed Computing and Internet Technology (ICDCIT)*, Lecture Notes in Computer Science, pages 83–93. Springer, 2012.

[59] Miroslaw Dynia, Jarosław Kutyłowski, Paweł Lorek, and Friedhelm Meyer auf der Heide. Maintaining Communication Between an Explorer and a Base Station. In *Biologically Inspired Cooperative Computing*, IFIP International Federation for Information Processing, pages 137–146. Springer US, 2006.

[60] Miroslaw Dynia, Jaroslaw Kutylowski, Friedhelm Meyer auf der Heide, and Jonas Schrieb. Local strategies for maintaining a chain of relay stations between an explorer and a base station. In *Proceedings of the 19th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 260–269. ACM, 2007.

[61] D. Jack Elzinga and Donald W. Hearn. The Minimum Covering Sphere Problem. *Management Science*, 19(1):96–104, 1972.

[62] Milan Erdelj, Nathalie Mitton, and Tahiry Razafindralambo. Robust Wireless Sensor Network Deployment. *Discrete Mathematics and Theoretical Computer Science*, 3(1):105, 2016.

[63] Michael Feldmann, Andreas Padalkin, Christian Scheideler, and Shlomi Dolev. Coordinating Amoebots via Reconfigurable Circuits. *Journal of Computational Biology*, 29(4):317–343, 2022.

[64] Kaspar Fischer, Bernd Gärtner, and Martin Kutz. Fast Smallest-Enclosing-Ball Computation in High Dimensions. In *Algorithms - ESA 2003*, Lecture Notes in Computer Science, pages 630–641. Springer, 2003.

[65] P. Flocchini, N. Santoro, G. Viglietta, and M. Yamashita. Rendezvous with constant memory. *Theoretical Computer Science (TCS)*, 621:57–72, 2016.

[66] Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Moving and Computing Models: Robots. In *Distributed Computing by Mobile Entities: Current Research in Moving and Computing*, Lecture Notes in Computer Science, pages 3–14. Springer, 2019.

[67] Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Giovanni Viglietta. Distributed computing by mobile robots: Uniform circle formation. *Distributed Computing*, 30(6):413–457, 2017.

[68] Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Peter Widmayer. Gathering of asynchronous robots with limited visibility. *Theoretical Computer Science (TCS)*, 337(1-3):147–168, 2005.

[69] Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Peter Widmayer. Arbitrary pattern formation by asynchronous, anonymous, oblivious robots. *Theoretical Computer Science (TCS)*, 407(1):412–447, 2008.

[70] Paola Flocchini, Nicola Santoro, and Koichi Wada. On Memory, Communication, and Synchronous Schedulers When Moving and Computing. In *Proceedings of the 23rd International Conference on Principles of Distributed Systems (OPODIS)*, volume 153 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 25:1–25:17. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020.

[71] Oded Galor. *Discrete Dynamical Systems*. Springer, 2007.

[72] Noam Gordon, Israel A. Wagner, and Alfred M. Bruckstein. Gathering Multiple Robotic A(ge)nts with Limited Sensing Capabilities. In *Ant Colony Optimization and Swarm Intelligence*, Lecture Notes in Computer Science, pages 142–153. Springer, 2004.

[73] Tony Greicius. Mars Perseverance Mission Overview. http://www.nasa.gov/perseverance/overview, July 2016. Accessed: 2023-03-08.

[74] Branko Grünbaum. *Metamorphoses of Polygons*, volume 11, pages 35–48. American Mathematical Society, 1994.

[75] Loura Hall. Marsbee - Swarm of Flapping Wing Flyers for Enhanced Mars Exploration. http://www.nasa.gov/directorates/spacetech/niac/2018_Phase_I_Phase_II/Marsbee_Swarm_of_Flapping_Wing_Flyers_for_Enhanced_Mars_Exploration, March 2018. Accessed: 2023-03-08.

[76] Rory Hector, Gokarna Sharma, Ramachandran Vaidyanathan, and Jerry L. Trahan. Optimal Arbitrary Pattern Formation on a Grid by Asynchronous Autonomous Robots. In *Proceedings of the 2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1151–1161. IEEE, 2022.

[77] Nojeong Heo and Pramod K. Varshney. A distributed self spreading algorithm for mobile wireless sensor networks. In *Proccedings of the 2003 IEEE Conference on Wireless Communications and Networking (WCNC)*, volume 3, pages 1597–1602 vol.3. IEEE, 2003.

[78] Taisuke Izumi, Zohir Bouzid, Sébastien Tixeuil, and Koichi Wada. Brief Announcement: The BG-Simulation for Byzantine Mobile Robots. In *Distributed Computing (DISC)*, Lecture Notes in Computer Science, pages 330–331. Springer, 2011.

[79] Taisuke Izumi, Maria Gradinariu Potop-Butucaru, and Sébastien Tixeuil. Connectivity-Preserving Scattering of Mobile Robots with Limited Visibility. In *Stabilization, Safety, and Security of Distributed Systems (SSS)*, Lecture Notes in Computer Science, pages 319–331. Springer, 2010.

[80] Taisuke Izumi, Samia Souissi, Yoshiaki Katayama, Nobuhiro Inuzuka, Xavier Défago, Koichi Wada, and Masafumi Yamashita. The Gathering Problem for Two Oblivious Robots with Unreliable Compasses. *SIAM Journal on Computing (SICOMP)*, 41(1):26–46, 2012.

[81] Daniel Jerison. General mixing time bounds for finite markov chains via the absolute spectral gap. https://arxiv.org/abs/1310.8021, 2013.

[82] Heinrich Jung. Über die kleinste Kugel, die eine räumliche Figur einschliesst. *Journal für die reine und angewandte Mathematik*, 123:241–257, 1901.

[83] Heinrich Jung. Über den kleinsten Kreis, der eine ebene Figur einschließt. *Journal für die reine und angewandte Mathematik*, 137:310–313, 1910.

[84] Branislav Katreniak. Biangular Circle Formation by Asynchronous Mobile Robots. In *Structural Information and Communication Complexity (SIROCCO)*, Lecture Notes in Computer Science, pages 185–199. Springer, 2005.

[85] Branislav Katreniak. Convergence with Limited Visibility by Asynchronous Mobile Robots. In *Structural Information and Communication Complexity (SIROCCO)*, Lecture Notes in Computer Science, pages 125–137. Springer, 2011.

[86] Latif Ullah Khan. Visible light communication: Applications, architecture, standardization and research challenges. *Digital Communications and Networks*, 3(2):78–88, 2017.

[87] David Kirkpatrick, Irina Kostitsyna, Alfredo Navarra, Giuseppe Prencipe, and Nicola Santoro. Separating Bounded and Unbounded Asynchrony for Autonomous Robots: Point Convergence with Limited Visibility. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 9–19. ACM, 2021.

[88] Peter Kling and Friedhelm Meyer auf der Heide. Convergence of local communication chain strategies via linear transformations: Or how to trade locality for speed. In *Proceedings of the 23rd Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 159–166. ACM, 2011.

[89] Peter Kling and Friedhelm Meyer auf der Heide. Continuous Protocols for Swarm Robotics. In *Distributed Computing by Mobile Entities: Current Research in Moving and Computing*, Lecture Notes in Computer Science, pages 317–334. Springer, 2019.

[90] Irina Kostitsyna, Cai Wood, and Damien Woods. Turning Machines. In *Proceedings of the 26th International Conference on DNA Computing and Molecular Programming (DNA)*, volume 174 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 11:1–11:21. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020.

[91] Jarosław Kutyłowski and Friedhelm Meyer auf der Heide. Optimal strategies for maintaining a chain of relays between an explorer and a base camp. *Theoretical Computer Science (TCS)*, 410(36):3391–3405, 2009.

[92] Geunho Lee and Nak Young Chong. A geometric approach to deploying robot swarms. *Annals of Mathematics and Artificial Intelligence*, 52(2):257–280, 2008.

[93] Geunho Lee, Yasuhiro Nishimura, Kazutaka Tatara, and Nak Young Chong. Three dimensional deployment of robot swarms. In *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5073–5078. IEEE, 2010.

[94] David A. Levin and Yuval Peres. *Markov Chains and Mixing Times*. American Mathematical Society, 2017.

[95] S. Li. Concise Formulas for the Area and Volume of a Hyperspherical Cap. *Asian Journal of Mathematics & Statistics*, 4(1):66–70, 2010.

[96] Shouwei Li, Christine Markarian, Friedhelm Meyer auf der Heide, and Pavel Podlipyan. A continuous strategy for collisionless gathering. *Theoretical Computer Science (TCS)*, 852:41–60, 2021.

[97] Shouwei Li, Friedhelm Meyer auf der Heide, and Pavel Podlipyan. The impact of the Gabriel subgraph of the visibility graph on the gathering of mobile autonomous robots. *Theoretical Computer Science (TCS)*, 852:29–40, 2021.

[98] J. Lin, A. S. Morse, and B. D. O. Anderson. The Multi-Agent Rendezvous Problem. Part 1: The Synchronous Case. *SIAM Journal on Control and Optimization (SICON)*, 46(6):2096–2119, 2007.

[99] L. Losonczi. Eigenvalues and eigenvectors of some tridiagonal matrices. *Acta Mathematica Hungarica*, 60(3):309–322, 1992.

[100] Marcello Mamino and Giovanni Viglietta. Square Formation by Asynchronous Oblivious Robots. In *Proceedings of the 28th Canadian Conference on Computational Geometry (CCCG)*, pages 1–6. Simon Fraser University, Vancouver, British Columbia, Canada, 2016.

[101] Mark W. Moffett, Simon Garnier, Kathleen M. Eisenhardt, Nathan R. Furr, Massimo Warglien, Costanza Sartoris, William Ocasio, Thorbjørn Knudsen, Lars A. Bach, and Joachim Offenberg. Ant colonies: Building complex organizations with minuscule brains and no leaders. *Journal of Organization Design*, 10(1):55–74, 2021.

[102] Moumita Mondal and Sruti Gan Chaudhuri. Uniform Circle Formation by Swarm Robots Under Limited Visibility. In *Distributed Computing and Internet Technology (ICDCIT)*, Lecture Notes in Computer Science, pages 420–428. Springer, 2020.

[103] Angelia Nedic, Alex Olshevsky, Asuman Ozdaglar, and John N. Tsitsiklis. On Distributed Averaging Algorithms and Quantization Effects. *IEEE Transactions on Automatic Control*, 54(11):2506–2517, 2009.

[104] Linda Pagli, Giuseppe Prencipe, and Giovanni Viglietta. Getting Close without Touching. In *Structural Information and Communication Complexity (SIROCCO)*, Lecture Notes in Computer Science, pages 315–326. Springer, 2012.

[105] Linda Pagli, Giuseppe Prencipe, and Giovanni Viglietta. Getting close without touching: Near-gathering for autonomous mobile robots. *Distributed Computing*, 28(5):333–349, 2015.

[106] Debasish Pattanayak, Kaushik Mondal, H. Ramesh, and Partha Sarathi Mandal. Fault-Tolerant Gathering of Mobile Robots with Weak Multiplicity Detection. In *Proceedings of the 18th International Conference on Distributed Computing and Networking (ICDCN)*, pages 1–4. ACM, 2017.

[107] Pavan Poudel and Gokarna Sharma. Time-optimal uniform scattering in a grid. In *Proceedings of the 20th International Conference on Distributed Computing and Networking (ICDCN)*, pages 228–237. ACM, 2019.

[108] Pavan Poudel and Gokarna Sharma. Fast Uniform Scattering on a Grid for Asynchronous Oblivious Robots. In *Stabilization, Safety, and Security of Distributed Systems (SSS)*, Lecture Notes in Computer Science, pages 211–228. Springer, 2020.

[109] Pavan Poudel and Gokarna Sharma. Time-Optimal Gathering under Limited Visibility with One-Axis Agreement. *Information*, 12(11):448, 2021.

[110] Giuseppe Prencipe. Impossibility of gathering by a set of autonomous mobile robots. *Theoretical Computer Science (TCS)*, 384(2-3):222–231, 2007.

[111] Giuseppe Prencipe. Pattern Formation. In *Distributed Computing by Mobile Entities: Current Research in Moving and Computing*, Lecture Notes in Computer Science, pages 37–62. Springer, 2019.

[112] Melanie Schranz, Martina Umlauft, Micha Sende, and Wilfried Elmenreich. Swarm Robotic Behaviors and Current Applications. *Frontiers in Robotics and AI*, 7, 2020.

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

[114] Ichiro Suzuki and Masafumi Yamashita. Distributed Anonymous Mobile Robots: Formation of Geometric Patterns. *SIAM Journal on Computing (SICOMP)*, 28(4):1347–1363, 1999.

[115] S. G. Walker and P. Van Mieghem. On lower bounds for the largest eigenvalue of a symmetric matrix. *Linear Algebra and its Applications*, 429(2):519–526, 2008.

[116] Damien Woods, Ho-Lin Chen, Scott Goodfriend, Nadine Dabby, Erik Winfree, and Peng Yin. Active self-assembly of algorithmic shapes and patterns in polylogarithmic time. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 353–354. ACM, 2013.

[117] Masafumi Yamashita and Ichiro Suzuki. Characterizing geometric patterns formable by oblivious anonymous mobile robots. *Theoretical Computer Science (TCS)*, 411(26):2433–2453, 2010.

[118] Yukiko Yamauchi, Taichi Uehara, Shuji Kijima, and Masafumi Yamashita. Plane Formation by Synchronous Mobile Robots in the Three Dimensional Euclidean Space. In *Distributed Computing (DISC)*, Lecture Notes in Computer Science, pages 92–106. Springer, 2015.

[119] Yukiko Yamauchi, Taichi Uehara, and Masafumi Yamashita. Pattern formation problem for synchronous mobile robots in the three dimensional euclidean space. https://arxiv.org/abs/1509.09207, 2015.

[120] Yukiko Yamauchi and Masafumi Yamashita. Pattern Formation by Mobile Robots with Limited Visibility. In *Structural Information and Communication Complexity (SIROCCO)*, Lecture Notes in Computer Science, pages 201–212. Springer, 2013.

[121] Wen-Chyuan Yueh. Eigenvalues of several tridiagonal matrices. *Applied Mathematics E-Notes*, 5:66–74, 2005.