Geometric Pattern Formation on Triangular and Rectangular Grids

Doctoral dissertation submitted by

Pritam Goswami

for the award of the PhD degree of

JADAVPUR UNIVERSITY

Kolkata, India

Supervisor:

Prof. Buddhadeb Sau

Department of Mathematics

Jadavpur University

Kolkata, India

FEBRUARY, 2025

Certificate from the Supervisor

This is to certify that this thesis entitled "*Geometric Pattern Formation on Triangular and Rectangular Grids*" submitted by **Mr. Pritam Goswami** who got his name registered on **10th February, 2022** for the award of Ph.D (Science) degree of Jadavpur University, is absolutely based upon his own work under the supervision of **Dr. Buddhadeb Sau, Professor, Department of Mathematics, Jadavpur University**, and that neither this thesis nor any part of it has been submitted for either any degree/ diploma or any other academic award anywhere before under my knowledge.

Buddhadeb Sau. 10/02/2025

(Signature of the Supervisor with date and official seal)

DR. BUDDHADEB SAU
Professor
Dept. of Maths., Jadavpur University
Kolkata - 700 032, INDIA

Dedicated To

Chandrima, My friend
Philosopher, Guide and
obviously the"better half".....

Preface

This thesis is submitted at Jadavpur University, Kolkata 700032, India for the degree "*Doctor of Philosophy*" in science. The research described herein is conducted under the supervision of Prof. Buddhadeb Sau at the Department of Mathematics, Jadavpur University, in the time period between July, 2019 and January, 2025.

This research work is original to the best of my knowledge except where the references and acknowledgments are made to the previous works. Neither this nor any substantially similar research work has been or is being submitted for any other degree, diploma or other qualification at any other university.

Some parts of this work have been presented in the following publications:

- Pritam Goswami, Avisek Sharma, Satakshi Ghosh, Buddhadeb Sau: **Time Optimal Gathering of Myopic Robots on an Infinite Triangular Grid**. Theoretical Computer Science, 1023:114930, 2025 `https://www.sciencedirect.com/science/article/pii/S0304397524005474`

- Pritam Goswami, Avisek Sharma, Satakshi Ghosh, Buddhadeb Sau: **Brief Announcement: Rendezvous on a Known Dynamic Point in a Finite Unoriented Grid.** SSS 2023: 374-379.

  `https://doi.org/10.1007/978-3-031-44274-2_27`

- Pritam Goswami, Manash Kumar Kundu, Satakshi Ghosh, Buddhadeb Sau: **Circle Formation by Asynchronous Opaque Fat Robots on an Infinite Grid.** Int. J. Parallel Emergent Distributed Syst., 39(2):214–247, 2024

  `https://doi.org/10.1080/17445760.2024.2316017`

Pritam Goswami
10/02/2025

.....

Date: 10/02/2025 Pritam Goswami

Acknowledgement

Phew!!! I've finally reached the end of a truly vibrant chapter of my life—my PhD journey. When I first embarked on this path, I often heard from my seniors that"PhD is a lifetime compressed into just 5 years." I'm not sure who coined this phrase, but now, I wholeheartedly agree. Throughout this"lifetime," I've encountered many people, gained immense knowledge from them, faced moments of stress, but overall, I've enjoyed every bit of it

As I reflect on this journey, I realize it wouldn't have been as rewarding without the presence of some extraordinary individuals by my side. Their constant guidance, constructive criticism, and unwavering support played a crucial role in helping me overcome various challenges. It would be unfair of me not to acknowledge them.

First of all, I would like to express my whole hearted gratitude to my supervisor, Professor Buddhadeb Sau. Without his extraordinary guidance and support and constant enthusiasm, it would have been really difficult to complete my thesis. His wisdom, academic and even sometimes beyond the academic realm, that he has bestowed upon me is a treasure for my whole life.

I would also like to express my sincere gratitude to all the faculty members of the Department of Mathematics at Jadavpur University, Ramakrishna Mission Residential College Narendrapur, and Ballygunge Science College. Their teaching and research have been a great source of motivation in my academic journey.

I am deeply grateful to Professor Kajal De, whose involvement in my research assessment committee greatly enriched my pursuit of knowledge.

I would like to give special recognition to all the members and office staff of Jadavpur University. Their unwavering efforts, from administrative support to maintaining the research infrastructure, have greatly facilitated my journey.

I would like to acknowledge University Grants Commission, Government of India for their financial support, which played a crucial role in enabling me to continue my research.

Continuing my research would have been impossible without the early nurturing of my love for logic and mathematics. For this, I am deeply grateful to my Mejojethu, Brajagopal Goswami, my Chotomama, Subrata Chatterjee, and my teacher, Amal Sir.

A special mention must go to my research collaborators—Dr. Manash Kumar Kundu, Dr. Satakshi Ghosh, Dr. Archak Das, Avisek Sharma, Raja Das, Brati Mondal, and Adri Bhattacharya. Their collective efforts and endless brainstorming made what could have been a challenging journey so much more manageable. I am also grateful to Dr. Kaustav Bose for his guidance and continuous motivation as a senior. Within the academic community, the companionship of my labmates—Dr. Ranendu Adhikary, Deboyoti Biswas, Dr. Sangita Patra, Dr. Arpita Dey, Dr. Suvankar Barai, Somnath, Nupur, Ayantika, and Payel—has been an invaluable gift.

My first teacher was my father, who introduced me to countless scientific concepts through simple stories and encouraged me to ask questions—an approach that still serves as the foundation of my research journey.

I am deeply grateful to my parents, Pran Gopal Goswami and Sujata Goswami, a gratitude that words can hardly express. Their constant love and belief in me, the sacrifices they've made for my education, and the values they've imparted have enriched my life in ways I continue to carry with me.

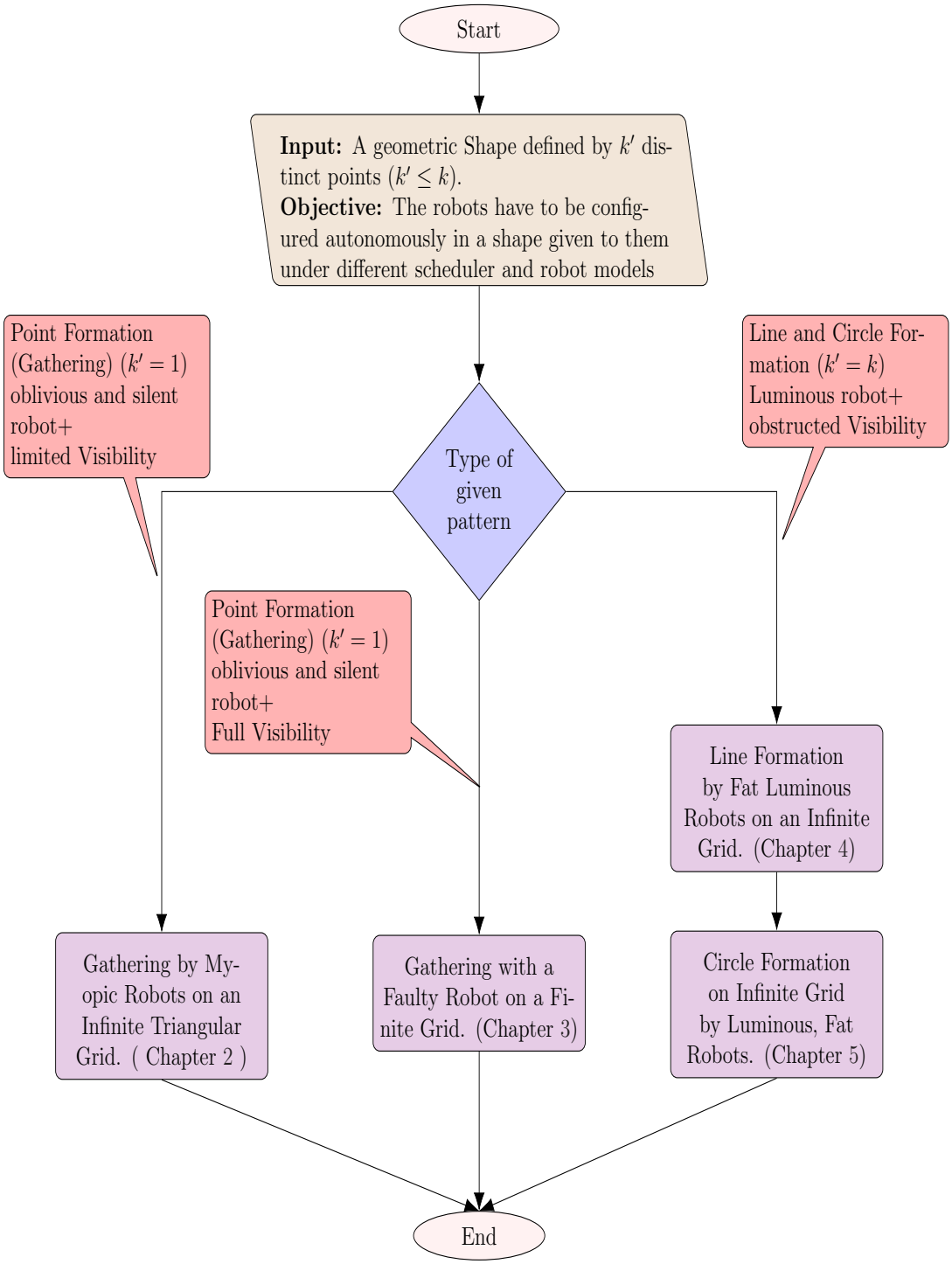
It would be truly unfair not to mention Soumya, my brother from another mother and Tintin, my cousin, in this acknowledgment. Their companionship is one of the most cherished and valuable things in my life. To all my family members, Chotka, Mintuma, Jhilli, Mejojethima, Pipi, Pipu, Nontu dada and Boudibhai, Bu, Dadubhai, Dida, Tipu dada, Moumita Boudibhai, Putu, Jolly didi, Avik da, Atri, Babu dada and Boudibhai, Punkai, Munkai thanks for being the source of my strength and joy throughout this journey.

I would also like to express my deepest gratitude to my father-in-law, Gopal Chandra Kayal, and my mother-in-law, Rina Kayal. Their unwavering support, selfless love, and affection have helped me navigate through many challenging

situations, making this journey much easier for me.

Finally, To whom I dedicated my thesis, my best friend, my philosopher and my guide and the love of my life, Chandrima Kayal. She is the sole reason I pursued academia and enjoyed it. Her faith in my abilities and constant positive criticism helped me shape myself to the person I am now. She is the main pillar supporting all my achievements so far in the last ten years. Her endless love for me, her support during demanding situations always been a great motivation throughout. So, as a tribute, I dedicate this thesis to my partner in crime, my source of strength, and the secret admirer of my bad puns—my "better" half, Hutom.

I feel incredibly blessed to have such a strong support system, and I acknowledge that this achievement would not have been possible without each and every one of you. My heartfelt thanks to all of you.


Start
Input: A geometric Shape defined by k' distinct points ($k' \leq k$).
Objective: The robots have to be configured autonomously in a shape given to them under different scheduler and robot models
Point Formation (Gathering) ($k' = 1$) oblivious and silent robot+ limited Visibility
Type of given pattern
Line and Circle Formation ($k' = k$) Luminous robot+ obstructed Visibility
Point Formation (Gathering) ($k' = 1$) oblivious and silent robot+ Full Visibility
Line Formation by Fat Luminous Robots on an Infinite Grid. (Chapter 4)
Gathering by Myopic Robots on an Infinite Triangular Grid. (Chapter 2)
Gathering with a Faulty Robot on a Finite Grid. (Chapter 3)
Circle Formation on Infinite Grid by Luminous, Fat Robots. (Chapter 5)
End

Contents

1 Introduction **1**

1.1 Fundamentals of Robot Swarm 5

1.2 Related Works 16

1.3 Overview of the Thesis 23

2 Gathering by Myopic Robots on an Infinite Triangular Grid **25**

2.1 Robot and Scheduler Abilities for Gathering 27

2.2 Impossibility Result 30

2.3 Description of Gathering 31

2.4 Conclusion 47

3 Gathering with a Faulty Robot on a Finite Grid **49**

3.1 Gathering Model 51

3.2 Lower Bound of Time and Impossibility 54

3.3 A Solution for Dynamic Gathering 58

3.4 Conclusion 81

4 Line Formation by Fat Luminous Robots on an Infinite Grid **83**

4.1 Robot Capabilities, Scheduler and Grid Environment 84

4.2 P1FC Formation 87
4.3 Modification to Algorithm 6 for Line Formation 112
4.4 Concluding Remarks 112

5 Circle Formation on Infinite Grid by Luminous, Fat Robots 115
5.1 Robot Abilities for Circle Formation and Related Definitions . . . 115
5.2 Circle Formation 116
5.3 Concluding Remarks 134

6 Conclusion 137

Chapter 1

Introduction

A robot swarm consists of a collection of simple, inexpensive, autonomous robots that work collaboratively to perform complex tasks. The motivation for using a collection of generic robots with limited capabilities to perform complex tasks comes from nature. The main idea is to simulate biological systems like bee swarms, fish schools, ant colonies, bird flocks, etc. In these systems, the group exhibits complex behaviors to execute complex tasks collectively, even though each individual entity within the group has limited capabilities. Likewise, the goal of swarm robotics is to enable a large number of simple, generic robots to execute complex tasks using only local rules and without centralized control. In a swarm, each robot independently follows an algorithm based on its own observations, making the system a distributed one.

A swarm of such weak, simple and low-cost robots are often more efficient in executing some tasks rather than using a single powerful and complex robot. This is because swarming offers many advantages that are useful and can not be achieved in a single robot system. Some of the issues that a swarm of robot can easily overcome are given in the following subsection.

1.0.1 Issues faced in single robot system and how swarm of robots overcome them.

- **Cost-Effectiveness :** manufacturing a single powerful robot with advanced and complex structures and control systems can be costly. On the contrary, the individual units of a robot swarm are simple and generic, making them affordable and suitable for mass production. Even with a large number of robots, the cost remains lower than that of a single powerful robot. There are certain tasks where it is not possible to recover and reuse a robot afterwards. Using a costly robot to execute these tasks is economically not suitable. Deploying a swarm of robots to do these kind of tasks can be a viable alternative solution to such scenarios.

- **Robustness and Stability :** Another advantage that a swarm of robot offers is stability and robustness of the system. In a single robot system malfunction of some module can jeopardize the whole task that has to be executed by the robot whereas, in a swarm robot system, the robots collectively progress towards achieving the solution even when some robots stops working. This ability to withstand malfunctions or faults makes robot swarms especially attractive for tackling large-scale tasks in hostile or hazardous environments.

- **Scalability :** Scalability is a big problem in a single robot system. It takes a huge cost to scale a system in this model. On the contrary, a swarm of robot can be easily scaled up or down in the size of the swarm according to the requirement of the task. This flexibility in scaling comes useful in handling problems of different complexity and size.

- **Efficient Execution Time :** Robots in a swarm can work on different parts of the task simultaneously unlike the single robot that can execute only one part of the task at a time. This inherent parallel nature of the swarm ensures faster execution of the whole task for certain problems for the swarm, especially those that can be divided into smaller, parallelizable sub-problems.

- **Energy Efficiency:** Swarm robots can be designed to be more energy-efficient

for specific tasks, reducing the overall energy consumption compared to a single, more powerful robot.

1.0.2 Real world applications of robot swarm

Thanks to these advantages, robot swarms can be utilized in a diverse array of scenarios across various applications. Here are some examples of how robot swarms can be applied:

1. **Environmental Monitoring**

   - *Pollution Tracking :* Swarm robots can monitor air and water quality over large areas, detecting pollution sources and spreading patterns.
   - *Wildlife Monitoring :* A swarm of robots can observe wildlife, collect data on animal behaviour and population dynamics without causing any hindrance for them at their natural habitat.

2. **Agriculture**

   - *Crop monitoring :* Real time data on soil assessment, crop health can be provided to farmers by robot swarms.
   - *Precision Farming :* Targeted actions such as applying fertilizers, planting seeds, watering crops can be performed by a robot swarm while optimizing the resource use and increasing yield.

3. **Disaster Response**

   - *Search and Rescue :* A swarm of robot can quickly cover a large area in the aftermath of a disaster. Thus it can be used in search and rescue missions in case of such natural calamities [90]. Hazards such as gas leak, fire etc. can be searched by the robots. Furthermore, they can also be used for locating and rescuing survivors.

4. **Military and Defence**

- *Surveillance :* Swarm robots can conduct surveillance mission providing real-time data for hostile and inaccessible areas [80].

5. **Exploration**

- *Space Exploration :* A swarm of robots can explore planetary surfaces to gather data and samples.
- *Underwater Exploration :* A robot swarm can map and monitor underwater environments, studying marine life and geological formations. Also they can monitor leakage in underwater oil storage tanks and pipelines [65]

There is an extensive range of potential application scenarios for robot swarms, even beyond those mentioned above. Swarming, due to its advantages over the single robot system and the huge application possibilities distributed coordination of robot swarms has attracted a considerable amount of research interest.

1.0.3 Dawn of research in swarm robotics

Initial research in this area was predominantly carried out by the artificial intelligence community. One of the pioneering studies by Maja Matarić in 1994 demonstrated how simple local interactions within a swarm could result in complex global behaviors [73]. Another significant study by Beckers et al. in 2000 explored how a system of simple autonomous robots could collect and aggregate pucks distributed over a rectangular area, showcasing the effectiveness of stigmergic communication [8]. Stigmergy, a method where individuals in decentralized systems communicate indirectly by altering their environment rather than through direct interaction, was a focal point of this research.

In 1992 Beni introduced the concept of Swarm Intelligence [9] providing another alternative approach to study multi robot systems. These methodologies provide different perspectives and frameworks for understanding and advancing the field of swarm robotics.

All these above mentioned approaches are of experimental nature, and focuses on practical implementations rather than formal correctness proofs of the algorithms. The topic of robot swarms was introduced to the distributed computing community in 1996 through two different studies of the pattern formation problem by Suzuki, Sugihara and Yamashita [91, 93]. These works took a different approach and investigated the computational perspective, aiming to understand how various features and capabilities of robots relate to the solvability of tasks. Since then, numerous theoretical investigations have examined the computational aspects of robot swarms from a distributed computing perspective. These studies aim to identify the minimum capabilities required for robots to solve specific problems, emphasizing rigorous mathematical proofs over heuristic approaches. The long-term goal is to develop a clear understanding of the relations between different robot features (such as memory, communication, sensing, synchronization, and agreement on local coordinate systems, visibility, dimension,) and the overall computability of the swarm. The recent book [53] offers a comprehensive survey of the extensive research conducted in this field. In this thesis, we continue this line of research by studying the Geometric Pattern Formation problem, a fundamental issue in robot coordination.

In Section 1.1, we give an overview of standard models of robot swarms considered in theoretical studies. In Section 1.2, we discuss the earlier works in this direction. Section 1.3 gives a brief overview of the thesis.

1.1 Fundamentals of Robot Swarm

This section presents the theoretical framework for examining computational and complexity issues in distributed swarm robot systems. The framework encompasses various aspects, including robot capabilities, scheduler functionalities, and different environments where the problems are analyzed. We begin by outlining the different robot capabilities below.

1.1.1 Robots in a Swarm

A *robot swarm*, denoted by $\mathcal{R} = \{r_1, r_2, \ldots r_k\}$ is a finite set of small, inexpensive and simple computing entities called "robots". Each robots can do local computation and move independently. Each of these robots are considered to be:

- *autonomous:* The robots are not controlled in a centralized way
- *anonymous:* The robots do not have any unique identification such as IDs
- *homogeneous:* Robots execute the same algorithm
- *identical:* Robots are physically indistinguishable

1.1.1.1 Physical Feature of a Robot

In most studies on swarm robot algorithms, robots are considered as points, referred to as "*point robots*". While point robots have significant theoretical im-

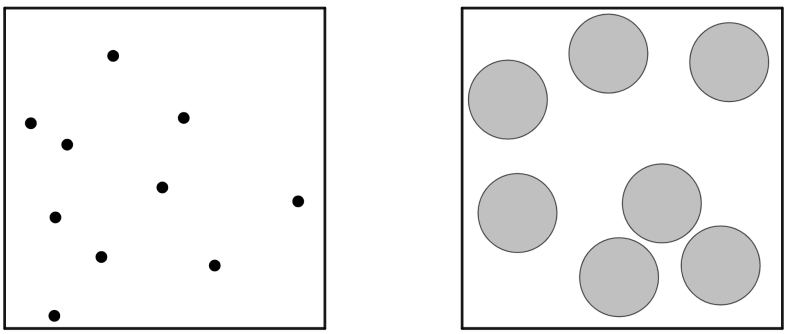


Figure 1.1: (a) Point robots on a plane. (b) Fat robots on a plane.

plications, in practice, a robot inevitably possesses some dimensions, no matter how small. To address this, the "*fat robot*" model was introduced. In this model, robots are represented as unit disks that occupy space within the environment, rather than merely points. This assumption provides a more realistic representation of robots, accounting for physical interactions and constraints.

1.1.1.2 Local Coordinate System

Each robot have a coordinate system that is only available to it. The coordinate system of a robot is called its *local coordinate system* . For any given robot r_i, the origin of its local coordinate system at any moment is its current location. It

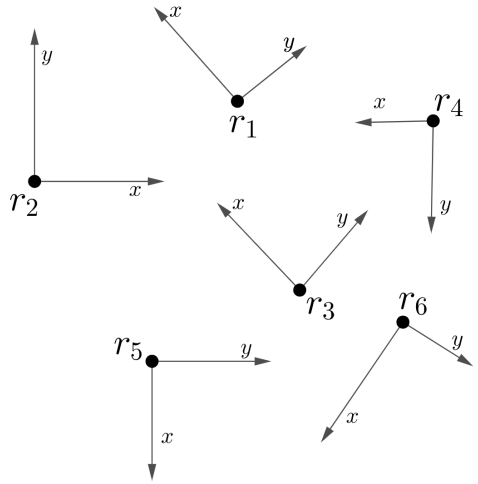


Figure 1.2: Local coordinates of robots in a swarm on a plane

is assumed that robots do not have access to any global coordinates. However, there may be some consistencies in the local coordinates within a swarm of robots, leading to a classification based on these consistencies:

1. **Two Axis Agreement:** Robots agree on the direction and orientation of both axes in a two-dimensional environment (See Figure 1.3(a))

2. **One Axis Agreement:** Robots agree on the direction and orientation of only one specific axis in their local coordinate system (See Figure 1.3(b))

3. **Chirality:** Robots agree on the cyclic orientation, that is, the clockwise and counter-clockwise direction (See Figure 1.4(a))

4. **No Agreement:** Robots have no consistency among their local coordinate systems. They do not agree on the direction or orientation of any axis, nor do they agree on cyclic orientation (Figure 1.4(b)).

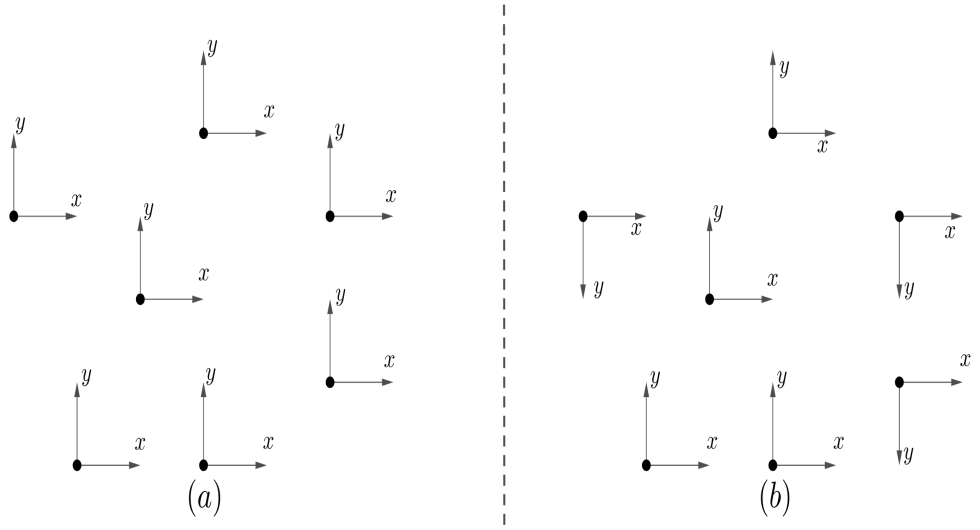


Figure 1.3: (*a*) *Two axis agreement*: robots agree on direction and orientation of both $x-$axis and $y-$axis. (*b*) *One axis agreement*: robots agree on the direction and orientation of only $x-$axis

1.1.1.3 Look-Compute-Move Cycle

Each robot has two states *idle* and *active*. A robot can move into active state from idle state or into idle state from active state. The moment when a robot moves into active state from idle state is known as *activation* of the robot. After activation, the active state of a robot can be further subdivided into three more phases: Look, Compute, and Move. Therefore, instead of saying that the robots continuously transition between the idle and active states, we describe it as follows: after activation, a robot sequentially executes the Look, Compute, and Move phases, then returns to the idle state until its next activation, at which point it repeats the process. So, each robot operates in a cycle where it executes the Look, Compute, and Move phases in sequence before becoming idle again. This cycle is called a *Look-Compute-Move* cycle, or *LCM* cycle (See Figure 1.5). Upon activation, a robot executes the Look Compute and Move phases in the following way.

- **Look :** A robot takes a snapshot of the positions of the other robots according to its local coordinate system.

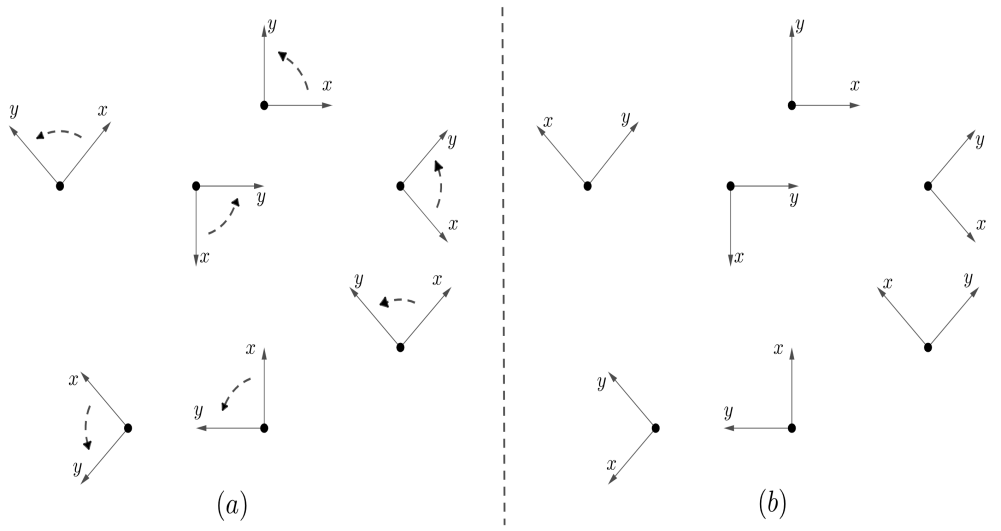


Figure 1.4: (*a*)*Chirality*: robots agree on clockwise and counter-clockwise direction. (*b*) *No agreement*: robots neither agree on the direction and orientation of any axes nor agree on any clockwise or counter-clockwise direction

- **Compute :** A robot runs an algorithm that takes as input the perceived locations of other robots relative to its local coordinate system (i.e., information from the Look phase). The output of this computation is a point with respect to its local coordinate system. This point is also known as *destination point*. Note that, because the robots are homogeneous, all robots in the swarm execute the same algorithm. However, since each robot has its local coordinate system, the input might differ from one robot to another. Consequently, even if they execute the algorithm simultaneously, the resulting output point may vary for each robot.

- **Move :** After executing the Compute phase, the robot determines a destination point. If this destination point differs from its current location, the robot moves to the destination point. Conversely, if the destination point is the same as its current location, the robot remains in place.

After completing the Move phase of the current LCM cycle, a robot becomes idle until its next activation. Upon activation, it executes another LCM cycle, continuing this process repeatedly.

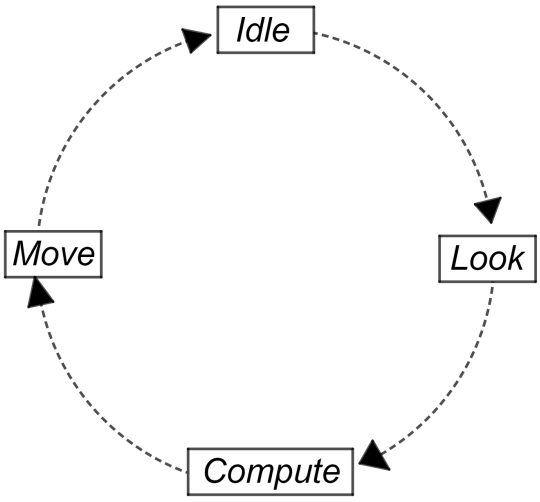


Figure 1.5: Look-Compute-Move cycle

1.1.1.4 Multiplicity Detection

Multiplicity refers to a point or vertex in the domain where multiple robots are located at the same time. Some robot models are capable of detecting multiplicities, and there are four main types of multiplicity detection abilities a robot can possess:

- *Global-strong multiplicity detection:* Robots with this ability can determine the exact number of robots at any given point or vertex.
- *Global-weak multiplicity detection:* Robots with this ability can detect if a point or vertex has multiple robots, but cannot determine the exact number of robots at that location.
- *Local-strong multiplicity detection:* Robots with this ability can count the exact number of robots at a point or vertex, but only if the detecting robot is also part of that multiplicity.
- *Local-weak multiplicity detection:* Robots with this ability can detect the presence of a multiplicity at a point or vertex, but only if the detecting robot is also part of that multiplicity.

1.1.1.5 Visibility

In the Look phase, a robot captures a snapshot of its surroundings to determine the locations of other robots based on its local coordinate system. The visibility model of a robot defines what is captured in this snapshot, essentially determining the extent of the robot's vision.In the literature, two primary visibility models are discussed: *non-restricted visibility* and *restricted visibility*.

In the non-restricted visibility model, a robot r_i captures a snapshot of the entire environment, including all robots present, regardless of their distance from r_i. Additionally, one robot cannot obstruct the view of another in this model. This model is also known as *full visibility* model.

The restricted visibility model can be further categorized into two types: *limited vision* and *obstructed visibility* .

- **Limited Vision :** This model deals with the range of vision of a robot. In this model it is assumed that a robot can only see upto a certain distance called *visibility range*, denoted as ϕ. The part of the environment visible to a robot r_i is called the *visibility region* of r_i. Robots having a finite visibility range is called a *myopic robot*. Note that in full visibility model it is assumed that $\phi \to \infty$.

  In a continuous environment $\mathcal{E}_{cont}$, the visibility region of a robot r_i is $B(R_i, \phi) \cap \mathcal{E}_{cont}$, where R_i is the location of r_i and $B(R_i, \phi)$ is an open disk of radius ϕ centered at R_i. In a discrete domain $\mathcal{E}_{disc}$ where $\mathcal{E}_{disc} = (V, E)$ is a graph, the visibility region of a robot r_i is the induced sub-graph $\mathcal{E}'_{disc} = (V', E')$ of $\mathcal{E}_{disc}$ where V' is the set of vertices in V that are at most ϕ hop away from the vertex on which r_i is located (See Figure 1.6).

- **Obstructed Visibility :** In obstructed visibility model it is assumed that the robots have unlimited visibility range but visibility of a robot can be obstructed by presence of some other robot.

  Formally, in case of point robots having obstructed visibility model a robot r_i can't see another robot r_j iff there is no other robot on the line segment

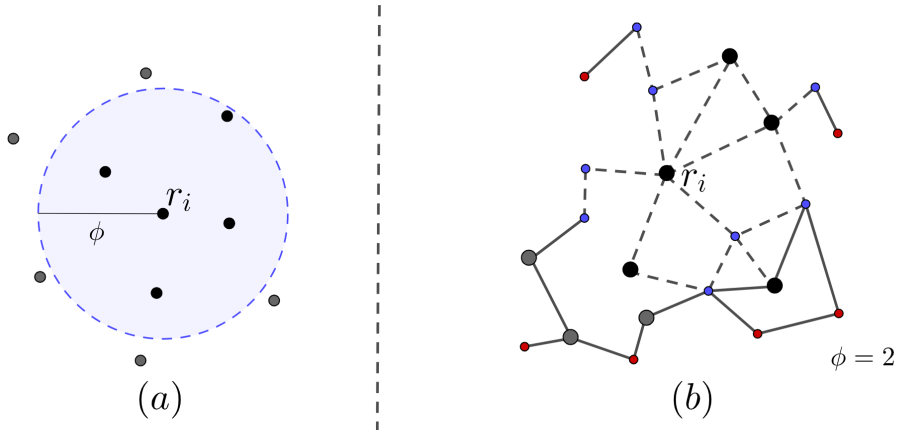


Figure 1.6: (a)*Limited vision on plane:* visibility region of r_i is the blue open ball. (b) *Limited vision on graph:* Visibility region of r_i is the sub-graph whose edges are marked with dotted lines for $\phi = 2$. r_i can not see the red vertices or the robots marked with color grey.

joining the locations of r_i and r_j (See Figure 1.7(a)).

For fat robots, r_i can see r_j iff there exists a point p_i on perimeter of r_i and a point p_j on the perimeter of r_j such that the line segment $\overline{p_i p_j}$ does not intersects with any other point p on a robot $r(\neq r_i, r_j)$ (See Figure 1.7(b)).

Note that, one can think of robots having this model of visibility as non-transparent or opaque. Hence, these robots are called *opaque robots*.

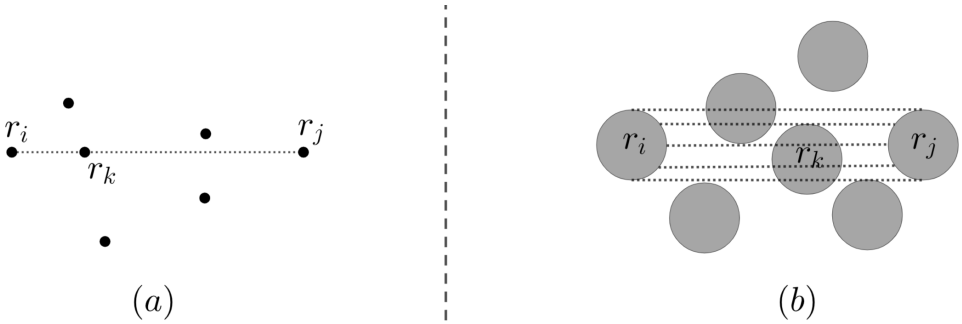


Figure 1.7: (a) *Opaque Point Robots:* r_i can not see r_j due to r_k. (b) *Opaque Fat Robots:* No three robots are co-linear still r_i can not see r_j.

1.1.1.6 Memory and Communication

Robots in a swarm can be categorized based on the availability of persistent memory and communication abilities. These categories are as follows.

- $\mathcal{OBLOT}$ **robots:** In this robot model, the robots are *oblivious* and *silent*. A robot is considered silent if it lacks the ability to communicate explicitly with other robots. The term oblivious means that the robots have no persistent memory to retain previous observations, computations, or actions. At the end of each LCM cycle, all gathered information (observations, computations, and actions) is erased. Therefore, the computations in each cycle are based solely on the observations made in the current cycle.

- $\mathcal{LUMI}$ **robots:** In this robot model, each robot is equipped with a persistent light that can display a finite number of colors. A robot can update its light's color during the Compute phase of a cycle. Once set, the color remains unchanged into the next cycle or until the robot updates it again. During the Look phase of a cycle, a robot can see its own color as well as the colors of other visible robots. This ability allows a robot to use its color as a state marker to remember information in the next cycle, acting as a form of persistent memory. Since the number of available colors is finite, a robot can remember only a limited number of states from previous cycles. Additionally, in this model, robots use colors to communicate with each other. Because robots in the $\mathcal{LUMI}$ model can see the colors of other robots, colors serve as signals. However, the number of possible messages is finite due to the limited number of colors.

- $\mathcal{FSTA}$ **robots:** In this robot model the robots are equipped with persistent light having finite colors as well. However, a robot can only see its own color and not the colors of other robots. Thus, in this robot model, robots can use colors as finite persistent memory, making them non-oblivious. However, they are unable to communicate explicitly, as they cannot see the colors of other robots, making them silent.

- $\mathcal{FCOM}$ **robots:** Similar to the $\mathcal{LUMI}$ model, these robots are equipped with a persistent light that can display a finite number of colors. However, in this model, robots can see the colors of other robots but not their own. As a result, the robots do not have finite persistent memory to remember previous states (i.e., oblivious), but they do have the ability to communicate with other robots using signals of finite size (i.e., non-silent).

1.1.1.7 Activation and Synchronization

It is assumed that an entity called the *scheduler* controls the activation and timing of the robots' actions. The literature primarily considers three types of schedulers: (1) Fully Synchronous ($\mathcal{FSYNC}$), (2) Semi-Synchronous ($\mathcal{SSYNC}$), and (3) Asynchronous ($\mathcal{ASYNC}$) (See Figure 1.8). The descriptions of each scheduler are as follows.

1. **Fully Synchronous ($\mathcal{FSYNC}$):** In the Fully Synchronous model, time is divided into global rounds. In each round, all robots perform one LCM cycle simultaneously. This means that all robots take their snapshots at the same time and execute their actions simultaneously. Consequently, the Look, Compute, and Move phases of all robots are synchronized.

2. **Semi-Synchronous ($\mathcal{SSYNC}$):** In the Semi-Synchronous scheduler model, time is still divided into global rounds similar to the Fully Synchronous model. The key difference here is that not all robots are activated in every round. However, to ensure fairness in the system, it is assumed that each robot is activated infinitely often.

3. **Asynchronous ($\mathcal{ASYNC}$):** Among all scheduler models, the most general one is the asynchronous scheduler model.In this kind of scheduler, the robots are activated independently, and each robot executes its LCM cycles independently. The time spent in Look, Compute, Move, and idle states is finite but unbounded, unpredictable, and varies among different robots. As a consequent, the robots do not share a common notion of time. Additionally in this scheduler model, a robot can be observed while moving,

leading computations to rely on outdated positional information. Moreover, the configuration perceived by a robot during the Look phase may change significantly before it decides to move.

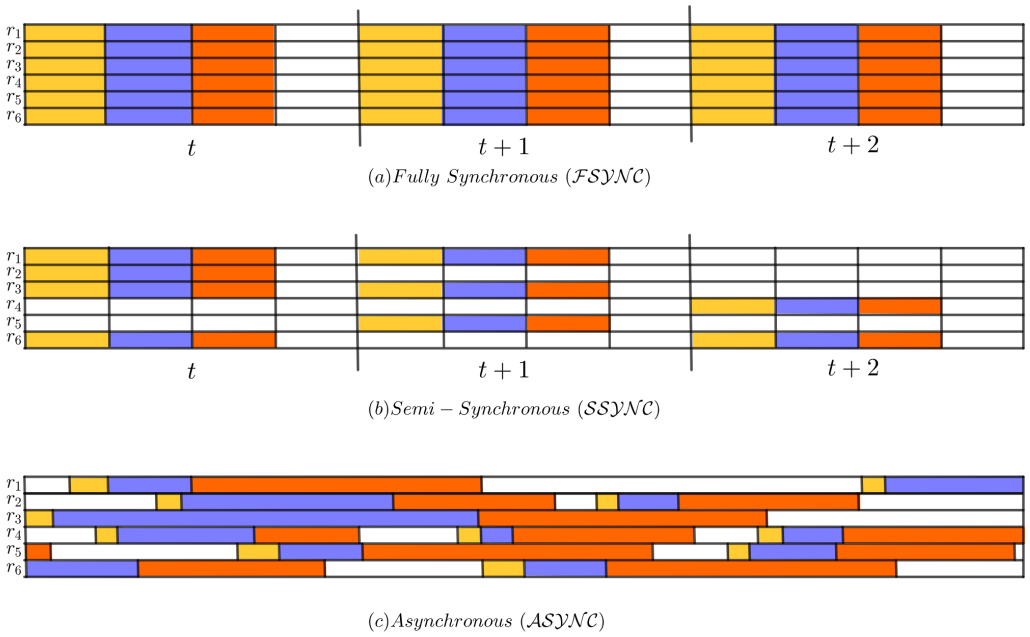


Figure 1.8: Three main scheduler models. The yellow, blue, orange and white colors are used for Look phase, Compute phase, Move phase and idle state of a robot.

1.1.1.8 Movement by robots

Typically, robots are assumed to move in a straight line, though some studies consider robots moving along specific trajectories. The speed of movement is another important factor. In the $\mathcal{FSYNC}$ and $\mathcal{SSYNC}$ scheduler models, robot speed is irrelevant because all movements complete before the next global round begins. However, in the $\mathcal{ASYNC}$ scheduler model, a robot's speed determines the duration of its Move phase in each LCM cycle. To ensure uniformity in the duration of the Move phase, a robot may sometimes stop before reaching its intended destination within the current LCM cycle. In this context, there are two main models describing how a robot moves, as discussed below.

- **Rigid:** In the rigid movement model, each robot is assumed to reach its destination point within its current LCM cycle without any interruptions.
- **Non-rigid:** In this model of robot movement, a robot may stop before reaching its destination point within a LCM cycle. More formally, there exists a $\delta > 0$ such that: 1)If the distance to the destination point exceeds δ, the robot moves at least δ towards the destination point in the current LCM cycle. 2) If the distance to the destination point is no more than δ, the robot is guaranteed to reach the destination point in the current cycle.

  These assumptions are necessary because, without them, the model could interrupt a robot after moving $\frac{1}{2^i}$ distance during its i-th activation, effectively restricting the robot to within 1 unit of distance from its starting position.

1.2 Related Works

This section provides a brief survey of theoretical studies on the computational and complexity issues in distributed computing by a swarm of robots.

1.2.1 A brief history on formation problems

One of the most fundamental and important coordination problem that has been extensively studied in the literature of this domain is Formation problem. There are mainly two categories in Formation problem: 1) Arbitrary Pattern Formation ($\mathcal{APF}$) and 2) Geometric Shape Formation.

In Arbitrary Pattern Formation or, $\mathcal{APF}$ problem, the robots are initially provided with the target pattern as a set of points with respect to some fixed coordinate system. However, in Geometric Shape Formation problem the robots are not provided with any set of target coordinates with respect to some fixed coordinate system. Instead in this problem the robots are instructed to form a particular geometric shape (e.g., point, line, circle etc.). Note that in Arbitrary Pattern Formation, a swarm can form any geometric shape,

but the robots must be given a consistent input of target points. In contrast, in the Geometric Shape Formation problem, the robots know only the shape they need to form without being given any specific coordinates.

The Formation problem was first studied in [94, 95]. These papers characterized the class of patterns formable in $\mathcal{FSYNC}$ and $\mathcal{SSYNC}$ schedulers, assuming robots with unbounded memory. The classification of formable patterns for $\mathcal{OBLOT}$ robots in $\mathcal{FSYNC}$ and $\mathcal{SSYNC}$ schedulers was later studied in [100]. Subsequently, [56] examined the $\mathcal{APF}$ problem for $\mathcal{OBLOT}$ robots under the $\mathcal{ASYNC}$ scheduler. This study showed that, with one axis agreement, a swarm with an even number of robots might fail to form certain patterns in the worst case, whereas a swarm with an odd number of robots can form any arbitrary pattern. They further demonstrated that with agreement on both axes, robots can form any pattern. The same paper provided a groundbreaking result connecting $\mathcal{APF}$ to the Leader Election problem. They proved that for a swarm of size $n \geq 3$, it is possible to solve Leader Election without any agreement on the coordinate system if $\mathcal{APF}$ can be solved. The Leader Election problem requires the robots to agree on a single robot as the leader. The relationship between Leader Election and $\mathcal{APF}$ for robots with chirality and no agreement on the coordinate system was further studied in [47]. In this paper, the authors provided algorithms to solve $\mathcal{APF}$ for a $\mathcal{OBLOT}+\mathcal{ASYNC}$ swarm of size $n \geq 4$ with chirality, and for a $\mathcal{OBLOT}+\mathcal{ASYNC}$ swarm of size $n \geq 5$ without chirality. Together with the results from [56], it followed that the $\mathcal{APF}$ and Leader Election problems are equivalent. More precisely, it is possible to form any arbitrary pattern without multiplicity points by either a swarm of $\mathcal{OBLOT}+\mathcal{ASYNC}$ robots of size $n \geq 4$ with chirality, or a swarm of $\mathcal{OBLOT}+\mathcal{ASYNC}$ robots of size $n \geq 5$ without chirality, if and only if Leader Election is solvable by the swarm. In both [56] and [47], the target pattern does not admit multiplicity points. The $\mathcal{APF}$ problem considering multiplicity points in the target pattern under the $\mathcal{OBLOT}+\mathcal{ASYNC}$ model was studied in [25, 58].

1.2.1.1 Issues in full vision

All the aforementioned works considered the FORMATION problem under a non-restricted visibility model. The problem was first studied under a limited visibility model in [102]. Under an obstructed visibility model, $\mathcal{APF}$ was examined in [14] with the assumption of opaque point robots and in [12] with opaque fat robots, both under the $\mathcal{LUMI}+\mathcal{ASYNC}$ model. Additionally, $\mathcal{APF}$ was studied from a randomized algorithm perspective in [17,103]. The problem of forming a sequence of patterns was investigated in [42].

1.2.1.2 Formation problem on discrete domains

Note that all the aforementioned works focused on the continuous domain. In the discrete domain, $\mathcal{APF}$ was first introduced in [13]. In this work, the authors defined the problem on an infinite rectangular grid and provided an algorithm that solves $\mathcal{APF}$ starting from any asymmetric configuration under the $\mathcal{OBLOT}+\mathcal{ASYNC}$ model. Multiplicity points were not allowed in the pattern to be formed. Later, in [24], the authors demonstrated that starting from any asymmetric configuration, any pattern can be formed if the domain is a regular tessellation graph. There are only three types of regular tessellation graphs: the infinite rectangular grid, the infinite triangular grid, and the infinite hexagonal grid. In this work, the pattern to be formed can include multiplicity points, thus generalizing the work in [13]. Under obstructed visibility model $\mathcal{APF}$ was studied in [69] for opaque fat robots under $\mathcal{LUMI}+\mathcal{ASYNC}$ model.

1.2.2 Development of works on gathering

In the GEOMETRIC SHAPE FORMATION problem, one of the most extensively studied issues is point formation, also known as GATHERING. In the GATHERING problem, n robots need to converge at a single point that is not known to them beforehand. The special case where $n = 2$ is known as the RENDEZVOUS problem. The significance of the RENDEZVOUS problem lies in demonstrating the distinction between the $\mathcal{OBLOT}+\mathcal{FSYNC}$ model and the $\mathcal{OBLOT}+\mathcal{SSYNC}$

model. RENDEZVOUS can be easily solved in the $\mathcal{OBLOT} + \mathcal{FSYNC}$ model by instructing the robots to move to the midpoint of the line segment joining them. However, in [95] it is proved that, it is impossible to solve RENDEZVOUS under the $\mathcal{OBLOT} + \mathcal{SSYNC}$ model without any additional assumptions, even with multiplicity detection. From a computational perspective, GATHERING of $n \geq 3$ robots is very different from RENDEZVOUS. Unlike RENDEZVOUS, GATHERING of $n \geq 3$ robots is solvable under the $\mathcal{OBLOT} + \mathcal{ASYNC}$ model if the robots have multiplicity detection capability [27]. RENDEZVOUS has been studied in models beyond the $\mathcal{OBLOT}$ model in [41,57,99]. Additionally, [26] provides an algorithm for solving GATHERING with $n \geq 3$ robots, where the robots can gather at a point even without multiplicity, but they have unlimited persistent memory to store all computations and information from the beginning of the execution. Without any persistent memory and multiplicity detection, GATHERING was solved under the asynchronous scheduler model ($\mathcal{OBLOT} + \mathcal{ASYNC}$) by assuming the robots agree on one axis (the one axis agreement model) for a swarm size of $n \geq 2$ [10].

POINT-CONVERGENCE is another relaxed version of the GATHERING problem. In this problem the robots need to move arbitrarily close to each other. More formally, for a set of robots $\mathcal{R} = \{r_1, r_2, \ldots r_k\}$, define $D^t(\mathcal{R}) = \max_{1 \leq i < j \leq k} \{r_i, r_j\}$. Then, POINT-CONVERGENCE is said to be solved if $\lim_{t \to \infty} D^t(\mathcal{R}) = 0$. The "move to the middle point" strategy for solving RENDEZVOUS in $\mathcal{OBLOT} + \mathcal{FSYNC}$ model can be generalized to move to the center of gravity of the configuration to solve POINT-CONVERGENCE under the $\mathcal{OBLOT} + \mathcal{ASYNC}$ model [28]. Note that, though it solves POINT-CONVERGENCE the move to COG strategy fails to solve GATHERING in absence of synchronicity.

1.2.2.1 Gathering with fat robots

The requirements to solve GATHERING is defined only for point robots. For fat robots the requirement to solve GATHERING is to form a *connected configuration*. In a connected configuration between any two points of any two robots there must exist a polygonal line each of whose points belong to some robot. GATHERING with fat robots have been extensively studied in [3, 22, 33].

1.2.2.2 Gathering in discrete domain

GATHERING has also been studied in discrete domains. In the context of trees, it has been shown that GATHERING can be solved from any initial configuration if and only if the initial configuration is balanced (cf. [39]). Later, in [68], it was demonstrated that GATHERING is unsolvable on a ring if multiplicity detection is completely absent in the $\mathcal{OBLOT} + \mathcal{ASYNC}$ model. The authors further showed that even with global-strong multiplicity detection, there are still certain initial configurations from which GATHERING is impossible.

In addition to the initial configurations mentioned in [68] where GATHERING is impossible, there are many other initial configurations for which the problem remains unsolvable, as provided in [40, 46, 88]. Subsequently in [64], an algorithm was introduced to solve GATHERING on a ring with fewer than $\lfloor \frac{n}{2} \rfloor$ robots starting from an asymmetric configuration under the $\mathcal{OBLOT} + \mathcal{ASYNC}$ model, assuming local-weak multiplicity detection. Additionally, another algorithm was proposed in [66] to solve GATHERING on a ring of n vertices with k robots, where $k < n - 3$ and k is odd, under the $\mathcal{OBLOT} + \mathcal{ASYNC}$ model starting from any initial configuration, assuming local-weak multiplicity detection. The most comprehensive result on the GATHERING problem in a ring, assuming local-weak multiplicity detection, was provided in [38] under the $\mathcal{OBLOT} + \mathcal{ASYNC}$ model. This study demonstrated that GATHERING is achievable for any initial configuration on a ring of n vertices with k robots, provided that $3 \leq k < n-4$ and $k \neq 4$. In [40], the authors, using global-weak multiplicity detection, offered a complete characterization of all initial configurations where GATHERING is solvable under the $\mathcal{OBLOT} + \mathcal{ASYNC}$ model.

The GATHERING problem has also been explored in the context of a rectangular grid. In [89], this problem was studied under full visibility, while in [21] and [78], it was examined under limited visibility models. Moreover, [89] approached the GATHERING problem from an optimization perspective, presenting an algorithm that not only solves the problem on a grid but also minimizes the total traveled distance. Apart from these, GATHERING and POINT-CONVERGENCE, RENDEZVOUS have also been studied under different faulty and error-prone robot

models [5, 10, 15, 16, 29, 44, 63, 75].

1.2.3 Development to the works on circle formation

Another extensively studied problem under Geometric Shape Formation is Circle Formation problem. The origin of this problem dates back to the work in [92]. The heuristic algorithm presented in this work was able to form an approximate circle on plane. Later, another approximation algorithm was presented in [96]. In [95], another variant of Circle Formation problem was explored. In this variant the robots not only have to form a circle but also the distance between any two consecutive robots on the circle has to be same. This variant is known as Uniform Circle Formation problem. In [43], authors presented one algorithm that solves Circle Formation problem without the assumption of orientation. The authors in another of their later work in [45] presented another deterministic algorithm for Circle Formation where the robots form a non-uniform circle and converges towards uniformity. Flocchini et al. presented another alternative algorithm for Uniform Circle Formation problem removing certain restrictions in [54]. Though their work is restricted only for $n \neq 4$ robots.

Note that all these previous works considering Circle Formation problem was considered under non-restricted visibility model on plane. Feletti et. al in [50] first considered the Uniform Circle Formation problem with opaque point robots under $\mathcal{LUMI} + \mathcal{FSYNC}$ model. In this work, the robots are initially placed on plane and are equipped with a light comprises of 5 colors. Building on their prior research, their current work in [51], extends the solutions to incorporate $\mathcal{ASYNC}$ scheduler. To solve this under asynchronous scheduler, it was shown that 19 colors are sufficient.

The pioneering work on using opaque fat robots for circle formation was presented in [48]. This algorithm addressed circle formation under a limited visibility model, taking into account global coordinate agreement.

In discrete domain, the Circle Formation problem was first defined by Ad-

hikary et al. in [2]. In this work, an algorithm was presented for solving the Circle Formation on an infinite rectangular grid with luminous opaque point robots equipped with a light possessing 7 colors under $\mathcal{LUMI}+\mathcal{ASYNC}$ model. In [62], authors extended upon [2] by solving Uniform Circle Formation under the same model, on infinite rectangular grid using only 5 colors. The algorithm provided by them was able to form an uniform circle of diameter length asymptotically equals to $O(n)$ where n being the swarm size. In the same work, another alternative algorithm is provided that is able to form an uniform circle of diameter asymptotically equals to $O(n^2)$ using only 4 colors.

1.2.4 Previous works on some other variants of formation problem

Another type of Geometric Shape Formation problem is Plane Formation. In this problem, robots are assumed to be in a domain with a dimension greater than two, and they need to form a configuration where all robots are located on the same plane. In [101], this problem was first studied under the $\mathcal{FSYNC}$ scheduler with chirality. In [97], the same problem was examined without the chirality assumption. Plane Formation was later studied in [98] under the $\mathcal{SSYNC}$ scheduler. Note that, all these works considers the domain to be three dimensional continuous space.

Another type of Formation problem is the Mutual Visibility problem, which is particularly relevant for opaque robots. The goal of this problem is to form a configuration where any pair of robots can see each other. The first study on the Mutual Visibility problem considered the $\mathcal{OBLOT}+\mathcal{SSYNC}$ model (cf. [72]), assuming that the robots were aware of the swarm size. The round complexity of this problem was later analyzed and improved in [82] under the $\mathcal{OBLOT}+\mathcal{FSYNC}$ model. Without knowledge of the swarm size, the Mutual Visibility problem was explored in [71], where the authors presented three algorithms: first, under the $\mathcal{LUMI}+\mathcal{SSYNC}$ model with non-rigid movement; second, under the $\mathcal{LUMI}+\mathcal{ASYNC}$ model with rigid movement; and finally, under the $\mathcal{LUMI}+\mathcal{ASYNC}$ model with non-rigid movement and one-axis agreement. This work

was later improved in [83] regarding the number of colors used. Algorithms that work faster were provided in [84, 85]. In [11], the problem was solved under the $\mathcal{LUMI} + \mathcal{ASYNC}$ model with non-rigid movement without any axis agreement. The Mutual Visibility problem was also considered for opaque fat robots in [79, 81], considering faults in [6], and on infinite grids in [1, 61].

Apart from the Formation problem, many other challenges exist within this research domain. One such problem is flocking, where a swarm of robots collaborates to follow a leader, known as the flockhead [19, 20, 59, 87, 104]. Another extensively studied problem involves searching collaboratively for target(s) [7, 18, 31, 34–37, 70, 76]. Additionally, there is the area patrolling or perimeter patrolling problem [4, 23, 30, 32, 67], among others.

1.3 Overview of the Thesis

This thesis explores the computational and complexity issues related to the Geometric Shape Formation problem, with a particular emphasis on the Gathering, Line Formation, and Circle Formation problems. These problems are among the most extensively studied within the field of distributed swarm robot algorithms. However, the majority of previous research has been conducted under highly idealized and impractical conditions, such as assuming non-restricted visibility models and considering robots as point entities. Additionally, the few studies that have addressed these limitations have predominantly examined them in the continuous domain. In the continuous domain, robot movements are prone to errors since robots move based on distances that can be approximated real numbers, deviating from actual values. Addressing these problems within an embedded geometric graph provides a solution to this issue. In a graph, a robot can only be seen at a vertex, thereby eliminating the dependency on distances for movement in this domain. In particular, we examine the problem under models with limited and obstructed visibility, considering robots with physical dimensions (fat robots), and assuming faults within discrete domains. Our primary objective is to determine the optimal robot capabilities in terms of memory, com-

munication, and axis agreement necessary to solve these problems.

In Chapters 2 and 3 of this thesis, we explore the effects of restricted vision and faults on the GATHERING problem within a discrete domain. Chapter 2 delves into the necessary and sufficient conditions for resolving the GATHERING problem on an infinite triangular grid, where the robots possess a one-hop visibility range under the $\mathcal{OBLOT}+\mathcal{SSYNC}$ model. Notably, a problem solvable by robots with a limited one-hop vision range can also be tackled by the same set of robots even in an obstructed visibility model. In Chapter 3, we examine the GATHERING problem on a finite grid with n robots, one of which is faulty and can move arbitrarily in any of the four directions, provided it does not coincide with another robot.

In the subsequent chapters, we scrutinise the impact of physical dimensions and obstructed visibility on the LINE FORMATION and CIRCLE FORMATION problems within a discrete domain. Chapter 4 investigates the LINE FORMATION problem involving opaque, fat robots on an infinite rectangular grid. Following this, Chapter 5 delves into the CIRCLE FORMATION problem on the same infinite rectangular grid. Both of these problems are analysed under the $\mathcal{LUMI}+\mathcal{ASYNC}$ model.

Finally, in Chapter 6, we conclude the thesis by suggesting several potential directions for future research.

Chapter 2

Gathering by Myopic Robots on an Infinite Triangular Grid

GATHERING is a widely studied problem in the field of distributed swarm robot algorithms, with research dating back to the early days of distributed computing. Historically, much of the work on GATHERING has focused on scenarios where robots operate on a plane or have non-restricted visibility. In these planar or continuous domains, certain models assume that robots can move along curved trajectories with high precision. However, the accuracy of these algorithms depends on the robots' ability to execute these precise movements, which is often difficult due to mechanical limitations. Consequently, movements in a planar domain are more prone to errors. Additionally, the assumption of non-restricted visibility over arbitrarily large domains is impractical due to hardware limitations and high cost of implementing vision. This chapter aims to address these issues by proposing more realistic models that consider restricted visibility and discrete domains, thereby enhancing the feasibility and reliability of GATHERING in real-world applications.

The GATHERING problem with limited vision range was previously addressed in the context of the plane by Flocchini et al. (2005) under the $\mathcal{OBLOT}+\mathcal{ASYNC}$ model, assuming 2 axes agreement among robots [55] . Poudel et al. extended this work to infinite rectangular grids in [78], presenting two algorithms: the first algorithm assumes both axis agreement and a visibility range of two hops, while

the second algorithm assumes only one axis agreement but a visibility range of three hops .

In this chapter, we examine the GATHERING problem on an *infinite triangular grid* (See Definition 2.1) with $\mathcal{OBLOT}$ robots possessing one-hop vision under the $\mathcal{SSYNC}$ scheduler model. Infinite triangular grids have significant applications in programmable matter. Additionally, as demonstrated in [105], the coverage of robots equipped with sensors is maximized when they form a triangular grid with edge lengths of $\sqrt{3}s$, where s is the sensing radius of the sensors on the robots . Therefore, in terms of coverage efficiency, the triangular grid surpasses other regular tessellation grids. For these specific reasons, it was chosen to address the GATHERING problem within this particular discrete domain.

In this chapter, we first establish that one-axis agreement is both necessary and sufficient to solve the GATHERING problem on an infinite triangular grid with $\mathcal{OBLOT}$, myopic robots possessing a one-hop visibility range under the $\mathcal{SSYNC}$ scheduler. For the sufficiency part, we present an algorithm, 1-HOP 1-AXIS GATHER, which solves the problem in $O(n)$ *epochs*, where n is the size of the swarm. An epoch is defined as a time interval in which each robot has been activated at least once. We also demonstrate that any GATHERING algorithm requires at least $\Omega(n)$ epochs to solve the problem, establishing that 1-HOP 1-AXIS GATHER is time-optimal. In the following Table 2.1 a brief comparison has been presented between the mentioned previous works and this chapter.

The remainder of the chapter is structured as follows. Section 2.1 introduces the notations, definitions, and formal description of the model and problem under consideration. In Section 2.2, we prove the necessity of one-axis agreement. Then, in Section 2.3, we describe the 1-HOP 1-AXIS GATHER algorithm and provide a correctness analysis. This section also includes the proof of the lower bound on time complexity. Finally in Section 2.4 we conclude this chapter.

SL. No.	Algorithm	Axis Agreement	Visibility	Type
1	Algorithm in [24]	No axis agreement	Full visibility	Ideal
2	Algorithm in [55]	Two axis	$V \in \mathbb{R}(> 0)$	Ideal
3	1^{st} Algorithm in [78]	Two axis	$2\times$ edge length	Ideal
4	2^{nd} Algorithm in [78]	One axis	$3\times$ edge length	Relaxed
5	Algorithm in this Chapter	One axis	$1\times$ edge length	Ideal

Table 2.1: Comparison table

2.1 Robot and Scheduler Abilities for Gathering

Initially, this section includes the formal description of the model considered along with some definitions. Then after that, the problem is defined formally. Before that let us define the triangular grid formally.

Definition 2.1. *[Infinite Triangular Grid] An infinite triangular grid $\mathcal{G}$ is an infinite geometric graph $G = (V, E)$, where the vertices are placed on $\mathbb{R}^2$ having coordinates $\{(k, \frac{\sqrt{3}}{2}i) : k \in \mathbb{Z}, i \in 2\mathbb{Z}\} \cup \{(k + \frac{1}{2}, \frac{\sqrt{3}}{2}i) : k \in \mathbb{Z}, i \in 2\mathbb{Z} + 1\}$ and two vertices are adjacent if the Euclidean distance between them is 1 unit.*

2.1.1 Robot and Scheduler Model:

Let $\mathcal{R} = \{r_1, r_2, r_3, \ldots r_n\}$ be a set of n robots placed on the vertices of an infinite triangular grid $\mathcal{G}$. The robots are considered to be autonomous, anonymous, homogeneous and identical. Furthermore, they are oblivious and silent ($\mathcal{OBLOT}$). The robots do not have any multiplicity detection ability i.e., a robot can not decide if a vertex contains more than one robot or not. The robots do not agree on any global coordinate system. However, they agree on the direction and orientation of the y-axes. Note that any vertex v of the infinite triangular grid $\mathcal{G}$ is

at the intersection of three types of lines. In this chapter, the robots will agree on the orientation and direction of any one of these three types of lines and consider it as its y-axis. Note that in this model the robots have a common notion of up and down but not about left or right.

As an input, a robot r_i takes a snapshot after waking. This snapshot contains the position of other robots r_i can see on $\mathcal{G}$, according to the local coordinates of the robot. In a realistic setting due to limitations of hardware, a robot might not see all of the grid points in a snapshot. So to limit the visibility of the robots we have considered the limited visibility model. In k-hop visibility model, each robot r can see all the grid points which are at most at a k-hop distance from r. In this chapter, the robots are considered to have 1-hop visibility (i.e., $k = 1$). Note that when $k = 1$, a robot placed on a vertex v of the infinite triangular grid $\mathcal{G}$ can only see the adjacent six vertices of v.

The robots operate in *Look-Compute-Move (LCM)* cycle. In each cycle a previously inactive or idle robot wakes up and does the following phases:

After being activated in Look phase, a robot placed on $u \in V$ takes a snapshot of the current configuration visible to it as an input. In this step, a robot gets the positions of other robots expressed under its local coordinate system. Then in Compute phase, a robot computes a destination point x adjacent to its current position, where $x \in V$, according to some deterministic algorithm with the previously obtained snapshot from Look phase as input. After determining a destination point $x \in V$ in the Compute phase, in Move phase, the robot moves to x through the edge $ux \in E$. Note that if $x = u$ then the robot does not move.

After completing one *LCM* cycle a robot becomes inactive and again wakes up after a finite but unpredictable number of rounds and executes the *LCM* cycle again.

In this chapter, we consider a semi-synchronous scheduler ($\mathcal{SSYNC}$). Under a semi-synchronous scheduler model, time is divided in global rounds and at the beginning of a round, a subset of all robots are activated. Note that, in a specific round, all activated robots executes the LCM cycle synchronously. Observe that,

a semi-synchronous scheduler can activate each robot in every round to mimic a fully synchronous scheduler. However, a fully synchronous scheduler cannot replicate a semi-synchronous scheduler, as it cannot select a proper subset of robots to activate exclusively in a particular round. This distinction renders a semi-synchronous scheduler more general than a fully synchronous scheduler.

Furthermore, the local coordinate system of a robot is initially determined by an adversary, following the agreed-upon rules of axes and orientation. Once established, this coordinate system cannot be altered thereafter.

2.1.2 Notations and definitions

It is to be noted that robots do not have access to this coordinates. This coordinates are used simply for describing the infinite triangular grid $\mathcal{G}$. We now, provide some notations and definitions required as preliminaries for this chapter.

Definition 2.2 (Configuration)**.** *A configuration formed by a set of robots R, denoted as $\mathcal{C}_R$ (or, simply $\mathcal{C}$) is the pair $(\mathcal{G}, f)$ where, f is a map from V to $\{0, 1\}$. For $v \in V$, $f(v) = 1$ if and only if there is at least one robot on the vertex v.*

For some round $t \geq 0$, the configuration at the beginning of round t is denoted as $\mathcal{C}(t)$.

Definition 2.3 (Visibility Graph)**.** *A visibility graph $G_{\mathcal{C}}$ for a configuration $\mathcal{C} = (\mathcal{G}, f)$ is the sub graph of $\mathcal{G}$ induced by set of vertices $\{v \in V : f(v) = 1\}$.*

It is not hard to produce a configuration $\mathcal{C}$ with disconnected $G_{\mathcal{C}}$, such that there exists no deterministic algorithm which can gather a set of robots starting from $\mathcal{C}$. So in this chapter, it is assumed that initially the visibility graph is connected and any algorithm that solves the Gathering problem should maintain this connectivity during its complete execution.

Definition 2.4 (Extreme)**.** *A robot r is said to be an extreme robot if the following conditions hold with respect to its visibility:*

1. *Robot r has a non empty view.*
2. *There is no other robot on the positive y-axis of r.*
3. *Either left or right or both the open half planes of r with respect to its y-axis is empty.*

2.1.3 Gathering on infinite triangular grid

Suppose, a swarm of n robots are placed on the grid points of an infinite triangular grid $\mathcal{G}$. The GATHERING problem requires the robots to perform moves in such a way that after some finite time all robots assemble at exactly one grid point and stay forever gathered at that grid point.

2.2 Impossibility Result

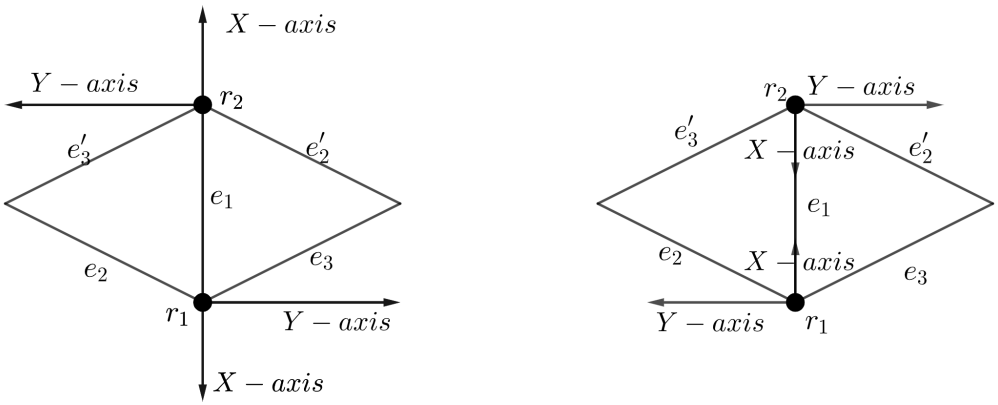


Figure 2.1: In the diagram the robots agree on direction of both the axes but do not agree on the orientation of the axes.

Theorem 2.1. *Gathering in a triangular grid is impossible without agreement on the orientation of any axis even when agreement on direction is present and under a fully synchronous scheduler and 1-hop visibility.*

Proof. Let Configuration A in Fig 2.1 is the initial configuration. Let there be an algorithm $\mathcal{AL}$ on finite execution of which two robots r_1 and r_2 from initial configuration A gathers on a single vertex of the triangular grid $\mathcal{G}$. Let there is an adversary who has decided the direction and orientation of both y and x axes for both the robots as it is in Fig 2.1. Note that view of r_1 and r_2 is same and they agree on the direction of both axes in the diagram. So if r_1 moves through e_1 according to $\mathcal{AL}$ on activation, r_2 also moves through e_1. Observe that, if r_1 moves through any other edge other than e_1 then r_2 moves in such a way that the visibility graph becomes disconnected. So, both r_1 and r_2 moves through edge e_1 and the configuration transforms into configuration B. Using a similar argument it can be shown that Configuration B can only transform into Configuration A as the vision of the robots are 1-hop. So, a deadlock situation occurs. So, our assumption must be wrong. Thus we can conclude that there does not exist any algorithm $\mathcal{AL}$ on the execution of which robots on a triangular grid with a vision of 1-hop gather even under a fully synchronous scheduler without any axis agreement. To be more precise, even if the robots agree on the direction of the axes they will still not gather when they do not have any agreement on the orientation of axes. □

Due to Theorem 2.1 we have considered one axis agreement model and devised an algorithm considering 1-hop vision under semi-synchronous scheduler.

2.3 Description of Gathering

In this section, an algorithm 1-HOP 1-AXIS GATHER (Algorithm 1) is provided that will work for a swarm of n myopic robots with one axis agreement and 1-hop visibility under a semi-synchronous scheduler. Note that under one axis agreement a robot can divide the grids into two half planes based on the agreed

line as the y-axis. An extreme robot r will always have either left or right or both the open half planes empty. Thus it is easy to see that when any one of the open halves is non-empty and r is on a grid point v (v is at coordinate $(0, 0)$ according to r's local coordinate system), two adjacent grid points of v on the empty open half plane and another adjacent grid point of v on y-axis and above r will always be empty. In this situation, r can uniquely identify the remaining three adjacent grid points of v (one on the y-axis and below r and the remaining two are on the non-empty half) based on the different values of their $y - coordinates$. So an extreme robot r can uniquely name them as $v_1(r), v_2(r)$ and $v_3(r)$ such that $y - coordinate$ of $v_i(r)$ is less than $y - coordinate$ of $v_{i+1}(r)$ and $i \in \{1, 2\}$ (Figure2.2). We simply write v_j instead of $v_j(r)$, $j \in \{1, 2, 3\}$ when r is clear from the context. Note that for a non-extreme robot r, there are two $v_2(r)$ and two $v_3(r)$ positions as r have either both open halves empty or both open halves non empty.

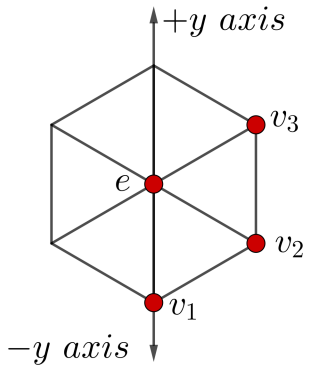


Figure 2.2: e is an extreme robot it can uniquely identify the positions of v_1, v_2 and v_3 if it sees right or left open half plane non empty.

In "1-HOP 1-AXIS GATHER" (See Algorithm 1), an extreme robot r moves to $v_1(r)$ if there is a robot on $v_1(r)$ and there is no robot on $v_3(r)$. r does not move when there is only a robot on $v_3(r)$ or there are robots only on $v_3(r)$ and $v_1(r)$. In the other remaining cases, if r sees at least one robot on the adjacent vertices it moves to $v_2(r)$. An extreme robot do not move knowing gathering has been achieved when it does not see any other robot on the adjacent vertices.

If r is not an extreme robot, then it only moves if there is no robot with $y-coordinate$ greater than zero within its vision and there are two robots on both of its $v_2(r)$ positions. In this scenario the robot r moves to $v_1(r)$.

In Figure 2.3 we have shown all possible views when a robot r moves and in which direction it moves. In Figure 2.3 suppose a robot r is placed on the node denoted by a black solid circle. The grid points that are encircled are occupied by other robots. For all the views of r in $View-I$, r moves to $v_1(r)$ and for all the views of r in $View-II$, r moves to $v_2(r)$.

Algorithm 1: 1-HOP 1-AXIS GATHER (for a robot r)

Data: Position of the robots on the adjacent grid points of r on triangular grid $\mathcal{G}$.
Result: A null movement or a vertex on $\mathcal{G}$ adjacent to r, as destination of r.

```
1  if r has an empty view then
2      do not move // This condition can be used by the robots to
           be aware of termination
3  else
4      if r is extreme then
5          if There is a robot only on v3(r) or there are robots only on both
             v1(r) and v3(r) then
6              do not move
7          else if There is a robot on v1(r) and no robot on v3(r) then
8              move to v1(r) // Figure 2.3(a) and 2.3(b)
9          else
10             move to v2(r) // Figure 2.3(e), 2.3(f) and 2.3(g)
11     else
12         if There is a robot on both v2(r) and no robot on the vertices with
             y − coordinate > 0 then
13             move to v1(r) // Figure 2.3(c) and 2.3(d)
```

2.3.1 Correctness results:

The intuition of Algorithm 1 is that the width of the configuration decreases while the visibility graph stays connected by the movement of the robots. The

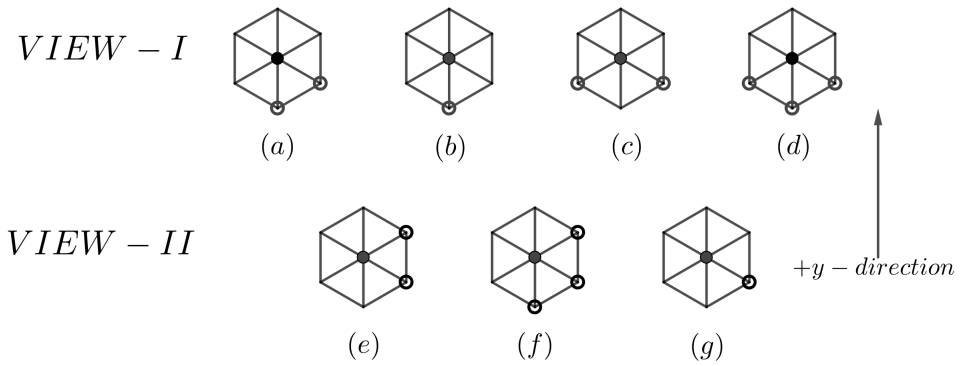


Figure 2.3: All possible views of a robot r placed on a node indicated by a black solid circle when r decides to move. Encircled point represents a robot occupied node. For all views in $View-I$, r moves to $v_1(r)$ position and for all views in $View-II$, r moves to $v_2(r)$ position. The cases where the robot positions are swapped with respect to $y-$ axis (for $(a), (e), (f)$ and (g)) are also included in the possible views where r decides to move.

following results will make this intuition more concrete. Before that let us have some definitions which will be needed in the proof of the results.

Definition 2.5 (Layer)**.** *Let H be a straight line perpendicular to the agreed direction of $y-$axis such that there is at least one robot on some grid points on H, then H is called a layer.*

Note that all points on a layer have same $y-$ coordinate value according to a coordinate system $\mathcal{O}$ in which the direction and orientation of $y-$ axis is same as the agreed direction and orientation of $y-$ axis by the robots. By $y-$ coordinate of a layer we mean the $y-$ coordinate of all points on that layer with respect to $\mathcal{O}$. The top most layer denoted as H_t is a layer such that there is no other layer with greater $y-$coordinate value with respect to $\mathcal{O}$. Observe that this coordinate system $\mathcal{O}$ is considered only for the sake of the definition and the robots are not aware of it.

Definition 2.6 (Vertical line, L_v)**.** *Let L_v be a line that is parallel to the agreed direction of the y-axis such that there is at least one robot on some grid point on L_v, then L_v is called a vertical line.*

We now define left and right edges of a configuration. The robots we have considered here do not share left and right orientations. The left and right directions that are mentioned in the definitions are used only for the sake of the correctness.

Definition 2.7 (Left and right edge, e_l and e_r)**.** *Left (right) edge of a configuration $\mathcal{C}$ or, e_l (resp. e_r) is the vertical line such that there is no other vertical line on the left (right) of e_l (e_r).*

Definition 2.8 (Width of a configuration $\mathcal{C}$)**.** *Width of a configuration $w(\mathcal{C})$ is defined as the distance between e_l and e_r.*

Definition 2.9 (Depth of a vertical line L_v)**.** *Depth of a vertical line L_v is defined as the distance between the layers H_t and the layer on which the lowest robot on L_v is located. We denote the depth of line L_v as $d(L_v)$.*

Fig 2.4 shows all the entities of the above definitions. A brief overview of the correctness proof is given below along with the statements of the results.

Overview of the correctness proof: In the subsequent Lemma 2.1, we prove that the visibility graph will remain connected throughout the execution of the algorithm. It is necessary to prove this as otherwise, the robots may gather in several clusters on the infinite triangular grid. Then we have shown in the subsequent Lemma 2.6 that the width of the configuration will decrease in finite time. Now when the width of the configuration becomes one then there are only two vertical lines that contain robots. These lines are left edge e_l and right edge e_r. Now in this scenario from the subsequent Lemma 2.2 the robots on the topmost layer will always move below and the depth of both e_l and e_r never increases (by the subsequent Lemma 2.5). So the depth of both the right and left edge now decreases in each epoch. Hence within finite time, the depth will also become one for either e_l or e_r. And in this scenario when the topmost layer shifts down again, all the robots gather at one grid vertex (by the subsequent Theorem 2.2).

Lemma 2.1. *For any $t \geq 0$, $G_{\mathcal{C}(t)}$ is connected.*

Proof. For some $t \geq 0$, let us consider the visibility graph $G_{\mathcal{C}(t)}$. We now prove the Lemma by showing the following.

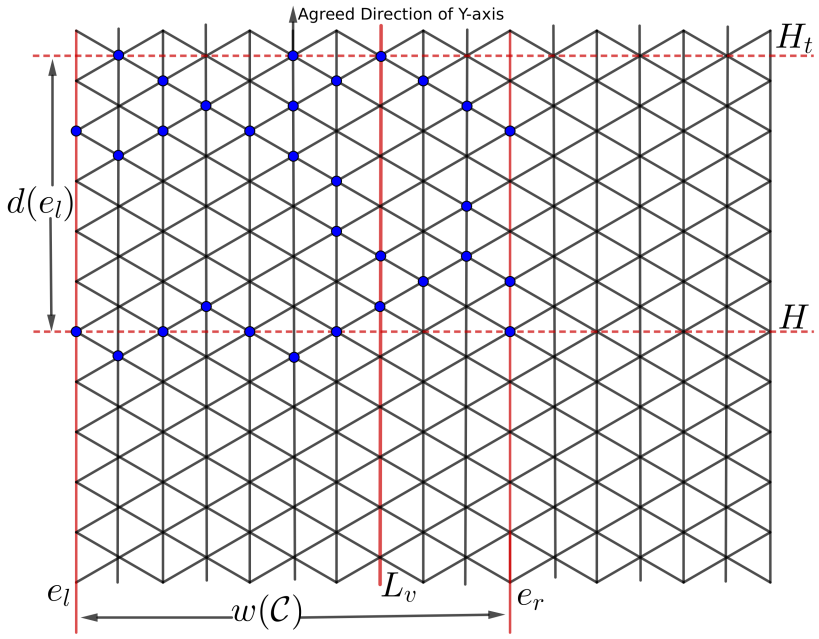


Figure 2.4: diagram of a configuration $\mathcal{C}$ mentioning layer (H), top most layer (H_t), vertical line (L_v), left edge (e_l), right edge (e_r), width of the $\mathcal{C}$ ($w(\mathcal{C})$) and depth of the vertical line e_l ($d(e_l)$).

"For a robot r , if $r' \in R$ is adjacent to r in $G_{\mathcal{C}(t)}$ then either r and r' are adjacent in $G_{\mathcal{C}(t+1)}$ or, r and r' are on the same vertex in $\mathcal{C}(t+1)$."
This will imply for any two robots $r, r' \in R$, if there is a path P between r and r' in $G_{\mathcal{C}(t)}$, then there exists a path P' between r and r' in $G_{\mathcal{C}(t+1)}$. So, if $G_{\mathcal{C}(t)}$ is connected then so is $G_{\mathcal{C}(t+1)}$. It is also assumed that for the initial configuration $\mathcal{C}(0)$, $G_{\mathcal{C}(0)}$ is connected. So, this argument is sufficient to prove that $G_{\mathcal{C}(t)}$ will stay connected, for any $t \geq 0$.

If the vertex on which r is located has more than one robot and at least one robot is not activated in $\mathcal{C}(t)$, it stays connected to r even if r moves during the round t, as even after the move of r, the distance from the previous vertex to the vertex r reaches after the move is 1-hop. So without loss of generality let r be singleton on its location. We now have two cases:

Case-I: Let us consider r is not an extreme robot. And in $\mathcal{C}(t)$, r decides to move. This implies in $\mathcal{C}(t)$, there are at least two robots, say r_1 and r_2, on both

of its $v_2(r)$ positions and no robot with $y-coordinate$ greater than zero. Now there are two sub-cases.

Case-I(a): Let us first assume that there is no robot on $v_1(r)$ position in $\mathcal{C}(t)$ (Figure 2.3(c)). So, only the robots on both the $v_2(r)$ positions are adjacent to r in $\mathcal{C}(t)$. Note that in $G_{\mathcal{C}(t)}$, r is adjacent to both r_1 and r_2 where, r_1 and r_2 be any robots that occupy two $v_2(r)$ positions in $\mathcal{C}(t)$. Also note that even if r_1 and r_2 are extreme and gets activated in $\mathcal{C}(t)$, they do not move during this round as r_1 and r_2 see r at $v_3(r_1)$ and $v_3(r_2)$ positions respectively and does not see any robot on $v_2(r_1) = v_2(r_2)$. Also, they do not move if they are not extreme as both of them see r at a position with $y-coordinate$ greater than zero. So in this case r moves to $v_1(r)$ during the move phase of round t. Note that $v_1(r)$ is 1-hop away from both the $v_2(r)$ position of r in $\mathcal{C}(t)$. So after r moves, r_1 and r_2 are both still adjacent to r in $G_{\mathcal{C}(t+1)}$.

Case-I(b): For the second case, assume there is at least a robot at $v_1(r)$ in $\mathcal{C}(t)$ along with r_2 and r_3 on both $v_2(r)$ positions (Figure 2.3(d)). So, only the robots at both the $v_2(r)$ positions and on the $v_1(r)$positions are adjacent to r in $\mathcal{C}(t)$. Let r_3 be any robot on $v_1(r)$. Then in $G_{\mathcal{C}(t)}$, r is adjacent to r_1, r_2 and r_3. Note that r_3 can not be extreme in $\mathcal{C}(t)$ as sees r on its positive $y-$axis. Hence, r_3 does not move even if it is activated in round t. Also, if r_1 and r_2 are not extreme they will not move during round t even if they are activated. Note that, r moves to $v_1(r)$ i.e., to the location of r_3 in $\mathcal{C}(t)$. Now with a similar argument for the above case, we can say r_1, r_2 are adjacent to r in $G_{\mathcal{C}(t+1)}$ and r_3 becomes colocated with r in $\mathcal{C}(t+1)$.

Now, assume that either r_1 or r_2 is extreme in $\mathcal{C}(t)$ and is activated during the round t. Without loss of generality let r_1 be extreme and assume it is activated during round t along with r. Now r_1 will see robots either in the positions $v_3(r_1)$ and $v_2(r_1)$ or on the positions $v_3(r_1), v_2(r_1)$ and $v_1(r_1)$ in $\mathcal{C}(t)$ (i.e., r_1 has view either Figure 2.3(e) or 2.3(f)). In both of these cases r_1 moves to $v_2(r_1)$ i.e., to the location of r_3 in $\mathcal{C}(t)$, along with r. So in $\mathcal{C}(t+1)$, r_1 and r_3 becomes colocated with r . Also by similar argument used in above cases, if r_2 does not move during round t, it stays adjacent to r in $G_{\mathcal{C}(t+1)}$, otherwise r_2 also becomes

colocated with r in $\mathcal{C}(t+1)$.

Case-II: Let us consider r is an extreme robot in $\mathcal{C}(t)$ that decides to move during round t. Then There are five possible views of r during round t.

Case-II(a): r only sees robots at $v_2(r)$ during look phase of round t (Figure 2.3(g)). So, only the robots on $v_2(r)$ are adjacent to r in $G_{\mathcal{C}(t)}$. Let r_1 be any robot on $v_2(r)$. Note that in $G_{\mathcal{C}(t)}$, r_1 is adjacent to r. Now in this case even if r_1 is activated during the round t, it either sees only r on $v_3(r_1)$ or sees robots on $v_3(r_1)$ and $v_1(r_1)$ during look phase of the round t. For both of the cases, r_1 does not move at round t. Now r moves to $v_2(r)$ (i.e., at the location of r_1 in $\mathcal{C}(t)$) during round t. So it is evident that in $\mathcal{C}(t+1)$ r becomes colocated with r_1.

Case-II(b): r only sees robots at $v_1(r)$ during the look phase of round t (Figure 2.3(b)). So, only robots on $v_1(r)$ are adjacent to r in $G_{\mathcal{C}(t)}$. Let r_1 be any robot on $v_1(r)$ in $\mathcal{C}(t)$. For any r_1 at $v_1(r)$, r_1 is adjacent to r in $G_{\mathcal{C}(t)}$. Note that r_1 is not an extreme robot and it sees r with $y-coordinate$ greater than zero. So, during round t even if r_1 is activated, it never moves. Now r moves to $v_1(r)$ (i.e., to the location of r_1 in $\mathcal{C}(t)$) during round t. So it is obvious that in $\mathcal{C}(t+1)$, r is on the same vertex along with r_1.

Case-II(c): r only sees robots at $v_3(r)$ and $v_2(r)$ during the look phase of round t (Figure 2.3(e)). So, only the robots on $v_3(r)$ and $v_2(r)$ are adjacent to r in $G_{\mathcal{C}(t)}$. Let at $\mathcal{C}(t)$, r_1 and r_2 be any robot on $v_3(r)$ and $v_2(r)$ respectively. Hence in $G_{\mathcal{C}(t)}$, r_1 and r_2 are adjacent to r. Now r moves to $v_2(r)$ (i.e., at the position of r_2 in $\mathcal{C}(t)$) during round t. Note that during the round t, even if r_2 is activated, it never moves as it is not an extreme robot and it sees r_1 with $y-coordinate$ greater than zero. So in $\mathcal{C}(t+1)$, r and r_2 will be colocated. Now it will be enough to show that either r_1 is adjacent to r in $G_{\mathcal{C}(t+1)}$ or, r_1 and r are colocated in $\mathcal{C}(t+1)$. Note that in round t, if r_1 does not move then r and r_1 remains adjacent in $G_{\mathcal{C}(t+1)}$ as $v(r)$ and $v(r_1)$ in $\mathcal{C}(t+1)$ are same as $v_2(r)$ and $v_3(r)$ in $\mathcal{C}(t)$ respectively and $v_2(r)$ and $v_3(r)$ are always adjacent for an extreme robot. So let us consider the cases where r_1 moves during the round t. First, assume that r_1 is not extreme in $\mathcal{C}(t)$ and it moves during round t. In this case r_1 must moves to the location

$v_1(r_1) = v_2(r)$ in $\mathcal{C}(t)$ (i.e., to the location of r_2 in $\mathcal{C}(t)$). Hence, in $\mathcal{C}(t+1)$, r,r_2 and r_1 are on the same vertex. Now let us assume that, r_1 is extreme in $\mathcal{C}(t)$. Then, it can see robots only on $v_2(r_1)$ and $v_1(r_1)$ in $\mathcal{C}(t)$ (i.e., r_1's view is same as the view in Figure 2.3(a)). In this case also r_1 moves to $v_1(r_1) = v_2(r)$ (i.e., to the location of r_2 in $\mathcal{C}(t)$). Thus with similar argument as above, in $\mathcal{C}(t+1)$, r,r_2 and r_1 are on the same vertex.

Case-II(d): r only sees robots at $v_2(r)$ and $v_1(r)$ during the look phase of round t (Figure 2.3(a)). So, only the robots at $v_2(r)$ and $v_1(r)$ positions are adjacent to r in $\mathcal{C}(t)$. Observe that for any r_1 at $v_1(r)$ and for any r_2 at $v_2(r)$, r is adjacent to r_1 and r_2 in $G_{\mathcal{C}(t)}$. Now, r moves to $v_1(r)$ during round t. Note that in $\mathcal{C}(t)$, r_1 is not extreme and sees r with $y-$ coordinate greater than zero. Hence r_1 does not move during round t. So, in $\mathcal{C}(t+1)$, r and r_1 are on the same vertex. Now it is enough to show that either r_2 is colocated with r in $\mathcal{C}(t+1)$ or, r_2 is adjacent to r in $G_{\mathcal{C}(t+1)}$. If r_2 does not move during round t then, in $\mathcal{C}(t+1)$, $v(r)$ and $v(r_2)$ are same as $v_1(r)$ and $v_2(r)$ in $\mathcal{C}(t)$. Now since $v_1(r)$ and $v_2(r)$ are always adjacent for an extreme robot r, it is evident that r and r_2 are adjacent in $G_{\mathcal{C}(t+1)}$. We now consider the cases where r_2 moves during round t. This is only possible if r_2 is extreme in $\mathcal{C}(t)$. Note that, during the look phase of round t, r_1 either sees robots at the positions $v_3(r_1)$ and $v_2(r_1)$ or sees robots at the locations $v_3(r_1), v_2(r_1)$ and $v_1(r_1)$ in $\mathcal{C}(t)$ (i.e., view of r_2 is same as either the view in Figure 2.3(e),or the view in Figure 2.3(f)). For both the views r_1 moves to $v_2(r_1) = v_1(r)$ (i.e, at the location of r_2 in $\mathcal{C}(t)$). Also r moves to $v_1(r) = v_2(r_1)$ (i.e., at the location of r_2 in $\mathcal{C}(t)$. So, in $\mathcal{C}(t+1)$, r, r_1 and r_2 are on the same vertex.

Case-II(e): r only sees robots say at the positions $v_1(r)$, $v_2(r)$ and $v_3(r)$ during the look phase of round t (Figure 2.3(f)). So, only the robots on the positions $v_1(r), v_2(r)$ and $v_3(r)$ are adjacent to r in $\mathcal{C}(t)$. Let, r is adjacent to r_1, r_2 and r_3 in $G_{\mathcal{C}(t)}$ for any r_1 at $v_1(r)$, r_2 at $v_2(r)$ and r_3 at $v_3(r)$ in $\mathcal{C}(t)$. Now, r moves to the position $v_2(r)$ in $\mathcal{C}(t)$ (i.e., to the location of r_2). Also note that r_2 is not extreme and has r_3 with $y-$ coordinate greater than zero in $\mathcal{C}(t)$. So r_2 does not move during round t. This implies in $\mathcal{C}(t+1)$, r and r_2 are colocated. Now it only remains to show that for any $r' \in \{r_1, r_3\}$, r' is either adjacent to r in

$G_{\mathcal{C}(t+1)}$, or colocated with r in $\mathcal{C}(t+1)$. Note that if r' does not move during round t, then $v(r')$ in $\mathcal{C}(t+1)$ is same as either $v_1(r)$ or, $v_3(r)$ in $\mathcal{C}(t)$ and $v(r)$ in $\mathcal{C}(t+1)$ is same as $v_2(r)$ in $\mathcal{C}(t)$. Now for an extreme robot r, both $v_1(r)$ and $v_3(r)$ are always adjacent to $v_2(r)$. So, r' remains adjacent to r in $G_{\mathcal{C}(t+1)}$. Now let r' moves during round t. Then note that $r' \neq r_1$. This is because, in $\mathcal{C}(t)$, r_1 is not extreme and r has $y-$coordinate greater than zero with respect to the local coordinate of r_1. So, r' must be r_3 in $\mathcal{C}(t)$. Note that if r_3 is not extreme but moves during round t then r_3 must move to $v_1(r_3) = v_2(r)$ in $\mathcal{C}(t)$. Now, if r_3 is extreme then it only sees robots at the positions $v_2(r_3) = v_1(r)$ and $v_1(r_3) = v_2(r)$ respectively during the look phase of the round t (i.e., view of r_3 is same as view in Figure 2.3(a)). So during round t, r_3 moves to $v_1(r_3) = v_2(r)$ in $\mathcal{C}(t)$ (i.e., the location of r_2 in $\mathcal{C}(t)$) . For both the cases when r' moves during round t, both r and $r'(= r_3)$ are colocated in $\mathcal{C}(t+1)$.

We have shown that, for any robot r , if $r' \in R$ is adjacent to r in $G_{\mathcal{C}(t)}$ then either r and r' are adjacent in $G_{\mathcal{C}(t+1)}$ or, r and r' are on the same vertex in $\mathcal{C}(t+1)$. From this we can conclude that if $G_{\mathcal{C}(t)}$ is connected then so is $G_{\mathcal{C}(t+1)}$. Now since the Initial configuration $\mathcal{C}(0)$ is connected, the graph $G_{\mathcal{C}}$ stays connected in each round. Hence $G_{\mathcal{C}(t)}$ is connected for any $t \geq 0$. □

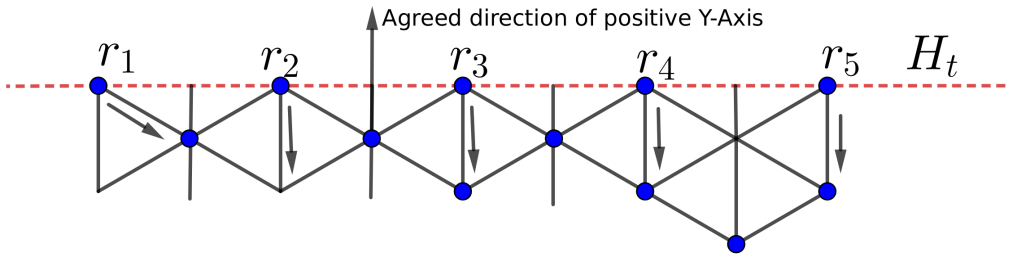


Figure 2.5: r_1, r_2, r_3, r_4 and r_5 are robots on H_t. Any robot on H_t will always have the view same as one of r_i, where $i \in \{1, 2, 3, 4, 5\}$. And for each of these 5 views, a robot always moves to another layer below H_t.

Lemma 2.2. *H_t of a configuration $\mathcal{C}$, always shift down in one epoch until the gathering is complete.*

Proof. Let r be a robot on H_t. If the gathering is not complete then r must see other robots on its adjacent vertices.

Now, there are two cases.

Case-I: If r is not extreme then upon activation r must see two robots on each of it's $v_2(r)$ position and no robot with $y - coordinate$ greater than zero (Figure 2.3(c) or, 2.3(d)). So upon activation r moves to $v_1(r)$ which is below H_t.

Case-II: If r is extreme, then there are three cases. Firstly if r sees a robot only on $v_2(r)$ upon activation (i.e., View in Figure 2.3g()), then it moves down to $v_2(r)$ which is below H_t. Secondly and thirdly, if r sees robot on only $v_1(r)$ (Figure 2.3(b)) or sees robots both on $v_1(r)$ and $v_2(r)$ (Figure 2.3(a)). For both second and third case, r moves to $v_1(r)$ which is also below H_t.

Since in one epoch, all robots on H_t must be activated once they must move below H_t. Hence, H_t of the configuration $\mathcal{C}$ always shifts down in one epoch. (Figure2.5). □

Lemma 2.3. *Robots on e_l or e_r which are not extreme do not move.*

Proof. Let r be a robot on e_l or on e_r which is not extreme. Note that a robot which is not extreme, only moves when it sees there is no robot above (i.e., no robots with $y - coordinate > 0$) and both of its $v_2(r)$ are occupied by some other robots. Now since r is on e_l or on e_r, at least one of it's $v_2(r)$ is empty. So r does not move. □

Lemma 2.4. *A robot r which is lowest on e_l or e_r never moves down to $v_1(r)$.*

Proof. We will proof this Lemma considering r on e_l. If r is on e_r the proof will be similar. Let r is the lowest robot on e_l. By Lemma 2.3, if r is not extreme it does not move. Now if r is extreme then no robot will move to $v_1(r)$ from a different vertical line as $v_1(r)$ is empty. So, r never moves to $v_1(r)$ as it can not see any robot on it's $v_1(r)$ position (Fig 2.6). So, r will never move to $v_1(r)$. □

Lemma 2.5. *Neither $d(e_l)$ nor $d(e_r)$ ever increase as long as the position of the corresponding vertical line is same.*

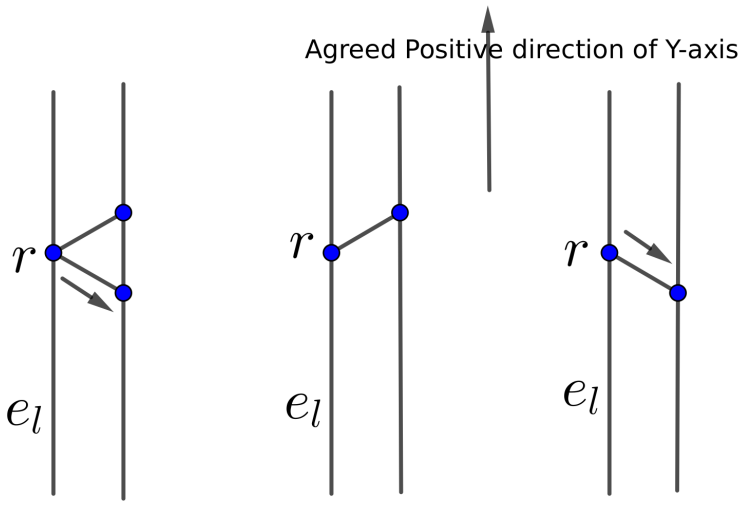


Figure 2.6: All possible view of the lowest extreme robot r on e_l. In each view r never moves directly below to $v_1(r)$.

Proof. We will prove this Lemma for e_l only. For e_r the proof will be similar. Observe that if $d(e_l)$ increase it can not be increased the by the lowest robot (say, r) on e_l (by Lemma 2.4). So the lowest robot can not increase $d(e_l)$. Also, no robot moves above the layer it is on. So, no robot on e_l moves up to increase $d(e_l)$.

Now it might be possible that $d(e_l)$ is increased by a robot that moves below the lowest robot r of e_l from the immediate right vertical line. Note that a robot moves from a vertical line to another vertical line only if it is extreme. Now, an extreme robot, if moves, never go to a position that is not occupied by any other robot before the movement according to algorithm 1(Fig 2.3). Since, to increase $d(e_l)$, a robot must move to a position that does not contain any other robot before the movement, no robot will come below r.

So, $d(e_l)$ never increases. Similarly we can say $d(e_r)$ never increases. Thus the result. □

Lemma 2.6. *If $w(\mathcal{C}(t_0)) > 0$ for some $t_0 \geq 0$ then there exists a round $t > t_0$ such that $w(\mathcal{C}(t)) < w(\mathcal{C}(t_0))$.*

Proof. Let $w(\mathcal{C}(t_0)) > 0$ for some $t_0 \geq 0$. Note that no robot from e_l moves left and no robots from e_r moves to its right. So $w(\mathcal{C}(t)) \leq w(\mathcal{C}(t_0))$, for all $t > t_0$. Now If possible let $w(\mathcal{C}(t)) = w(\mathcal{C}(t_0))$ for all $t > t_0$. This implies the vertical lines e_l and e_r never shifts to right and left respectively. Now by Lemma 2.5 we can say that $d(e_l)$ and $d(e_r)$ never increases. Also by Lemma 2.2 H_t shifts down always in one epoch. So from these two lemmas we can conclude that there exists $t_1 > t_0$ such that at the round t_1 either $d(e_l)$ or, $d(e_r)$ becomes 0. Note that when $d(e_l)$ (or, $d(e_r)$) is 0 then e_l (or, e_r) contains exactly one vertex v having robots. Let r be a robot on that vertex v. Note that r is extreme and by Lemma 2.1 since $G_{\mathcal{C}(t_1)}$ is connected r sees robots only on $v_2(r)$. So, r moves to $v_2(r)$ which is on the next vertical line on its right (or, left). So after a finite epoch either e_l shifts right or e_r shifts left and thus we arrive at a contradiction. Hence, there exists a round $t > t_0$ such that $w(\mathcal{C}(t)) < w(\mathcal{C}(t_0))$. □

Theorem 2.2. *Algorithm* 1-Hop 1-Axis Gather *solves the gathering problem on an infinite triangular grid for n oblivious, silent and myopic robots having 1-hop visibility and one axis agreement under semi-synchronous scheduler if in the initial configuration the visibility graph is connected.*

Proof. From Lemma 2.6 we can conclude that after a finite number of rounds there exists a round t_0 such that $w(\mathcal{C}(t_0))$ becomes zero. Observe that when $w(\mathcal{C}(t_0)) = 0$ all the robots are on a vertical line e_l. Also, note that when all the robots are on a single line then e_l is the same as e_r. In this situation if $d(e_l) = 0$ that means gathering is complete. So let us assume $d(e_l) > 0$. Note that in this scenario, since no non extreme robot r sees two robots on both of its $v_2(r)$ position and no extreme robot, say r', sees a robot in a location other than $v_1(r')$, no robot will move to a different vertical line from $e_l = e_r$. Now by Lemma2.2 and Lemma 2.5, H_t shifts down until there is only one grid point having robots on e_l (i.e $d(e_l) = 0$). So we can conclude that there exists a round $t > 0$ such that all robots are on a single vertex in $\mathcal{C}(t)$ and hence, gathering is solved. □

2.3.2 Complexity Analysis

First we observe in Theorem 2.3 that it will take at least $\Omega(n)$ epochs to gather n number of robots. The theorem is stated and proved formally in the following.

Theorem 2.3. *Any gathering algorithm on a triangular grid takes $\Omega(n)$ epoch.*

Proof. Let us consider the configuration in Figure 2.7. Let us define the height of the configuration as the distance between the topmost and lowest layer of the configuration. Note that in Figure 2.7 the height of the configuration is n (i.e., the number of robots). When considering the worst case, a robot can only be activated once in each epoch. Hence, the height of the configuration decreases by at most 2 units in each epoch. Now the robots will gather when the height of the configuration and width of the configuration both becomes 0. Since in each epoch, height decreases by 2 units, at least $\frac{n}{2}$ rounds will be needed to gather the n robots. Hence the result. □

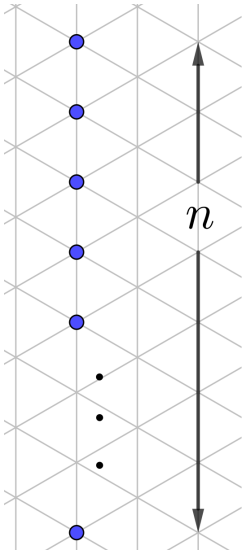


Figure 2.7: All of the n robots are on a straight line on the triangular grid. Height of the configuration is n.

Now we shall prove that the robots executing our proposed algorithm do not go downwards by much. First, we define the smallest enclosing rectangle for the initial configuration.

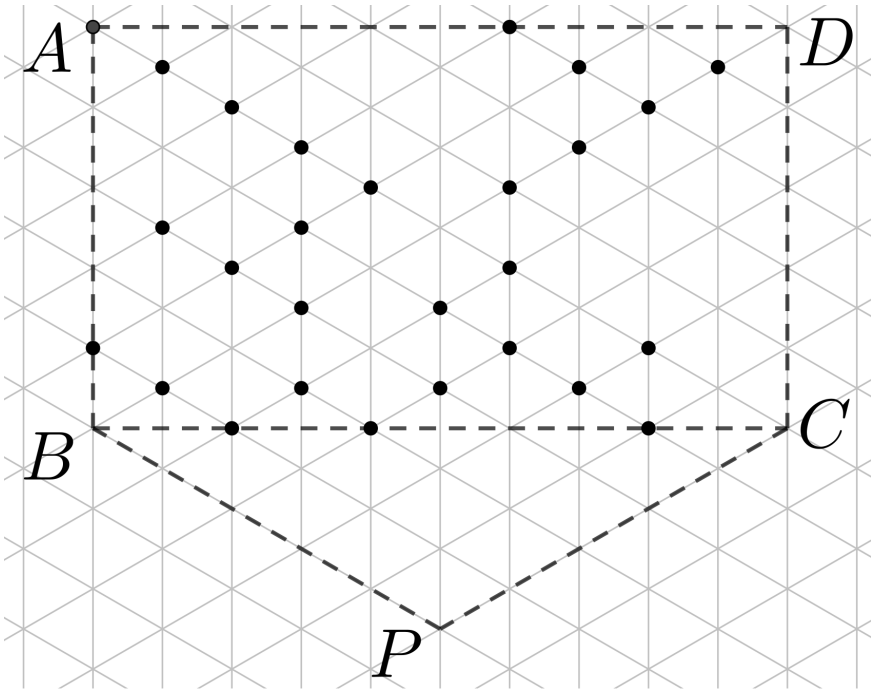


Figure 2.8: $ABCD$, smallest enclosing rectangle

Definition 2.10 ($\mathcal{SER}$)**.** *A rectangle $\mathcal{R} = ABCD$ is said to be the smallest enclosing rectangle ($\mathcal{SER}$) (Figure 2.8) of the initial configuration if it is the smallest in dimension satisfying the following:*

1. *All robots in the initial configuration are inside $\mathcal{R}$*
2. *All vertices of $ABCD$ are on some grid points*
3. *AB and CD side is parallel to the axis agreed by all the robots*
4. *BC is the lower side of the rectangle.*

Next, we define a polygon that shall contain all the robots throughout the algorithm.

Definition 2.11 (Bounding Polygon)**.** *Let $\mathcal{R} = ABCD$ be the $\mathcal{SER}$ of the initial configuration. Let P the point below BC line such that $\angle CBP = \angle BCP = \pi/6$. Then the polygon $\mathcal{P} = ABPCDA$ is said to be the Bounding Polygon.*

We show that no robot executing Algorithm 1 ever steps out of the bounding polygon (Lemma 2.7). Using Lemma 2.7, Theorem 2.4 proves that Algorithm 1 terminates within $O(n)$ epochs.

Lemma 2.7. *No robot executing Algorithm 1 ever steps out of the bounding polygon.*

Proof. Let $\mathcal{P} = ABPCDA$ be the bounding polygon (Figure 2.8). Note that, the point P is on some grid point. Opposite to our claim, let there be some robots that step out of $\mathcal{P}$ and k^{th} round is the earliest round when some robots stepped out $\mathcal{P}$. Let r be such a robot. Note that at the end of $(k-1)^{th}$ round, no robot is outside of $\mathcal{P}$. Firstly since no robot ever moves upward, so no robot can step out of $\mathcal{P}$ through the AD side and goes above to the AD side. Then r can step out of $\mathcal{P}$ through any side of $\mathcal{P}$ but AD. Note that, in this case, r must land on that side in some round before. Hence at the start of k^{th} round r must be on that side.

Let r has stepped out of $\mathcal{P}$ through AB side. If r is on any point but B then to cross the AB line and step out of $\mathcal{P}$ it has to change its vertical line. This is only allowed for an extreme robot according to our algorithm. So if r has to cross the vertical line while stepping out of $\mathcal{P}$ then r must be an extreme robot at k^{th} round. But according to our algorithm, since an extreme robot never occupies an empty grid point, this yields a contradiction. Even if r is at B and steps out of $\mathcal{P}$ without changing its vertical line then it must go down as a non extreme robot. But since the left $v_2(r)$ position of r is empty, so according to our algorithm, r wouldn't go down as a non extreme robot. Hence r can not step out of $\mathcal{P}$ through the AB line. With a similar argument, one can similarly show that r can not step out of $\mathcal{P}$ through the CD line.

Now let the robot r step out of $\mathcal{P}$ through the BP line or PC line. We already showed that r can not be at B or C at the start of k^{th} round. Let r be at some point on PB line or PC line. Now using the same argument as the previous case one can show that r can not be an extreme robot at the start of k^{th} round because r occupies an empty grid point in this round. Now we see r also cannot

be a non extreme robot at the start of k^{th} round. Because a non extreme robot only moves when both of its $v_2(r)$ position is nonempty. But this can not be true in this case for robot r. Hence this shows r cannot step out of $\mathcal{P}$ at all, which contradicts our assumption that some robots have stepped out of $\mathcal{P}$. □

Theorem 2.4. *The algorithm 1-hop 1-axis Gather takes at most $O(n)$ epochs to gather all the robots.*

Proof. Let $\mathcal{R} = ABCD$ be the $\mathcal{SER}$ of the initial configuration and $\mathcal{P} = ABPCDA$ be the bounding polygon. Since the total number of robots is n, from simple geometry $|BC| \leq (n+1)\frac{\sqrt{3}}{2}$ and so $|BC| \leq n+1$. Also the maximum distance from AD to P is $\frac{5}{4}(n+1)$. Therefore the maximum distance of H_t of the initial configuration from P is $\frac{5}{4}(n+1)$.
Now from Lemma 2.2 we can say that H_t shifts down at least half a unit in one epoch till the gathering is not done. And since the maximum distance of H_t of the initial configuration from P is $\frac{5}{4}(n+1)$, so within $2 \times \frac{5}{4}(n+1) = \frac{5}{2}(n+1)$ epochs the gathering must be complete. Hence the result follows. □

Theorem 2.5. *During Execution of the algorithm 1-hop 1-axis Gather a robot moves at most $O(n)$ times.*

Proof. Let $\mathcal{R} = ABCD$ be the $\mathcal{SER}$ of the initial configuration and $\mathcal{P} = ABPCDA$ be the bounding polygon. During execution of algorithm 1-hop 1-axis Gather a robot moves at most the height of the bounding polygon i.e., at most $\frac{5}{4}(n+1)$ for n many robots. Thus a robot can move at most $O(n)$ times. □

2.4 Conclusion

Gathering is a classical problem in the field of swarm robotics. The literature on the gathering problem is vast as it can be considered under many different robot models, scheduler models, and environments. Limited vision is very practical

when it comes to robot models. To practically implement any algorithm considering a robot swarm having full visibility is impossible. So, we have to transfer the research interest towards providing algorithms that work under limited visibility also. This paper is one achievement towards that goal.

In this paper, we have done a characterization of gathering on an infinite triangular grid by showing that it would not be possible to gather from any initial configuration to a point on the grid if the myopic robots having a vision of 1-hop do not have any axis agreement even under the FSYNC scheduler. Thus, considering one axis agreement we have provided an algorithm that gathers n myopic robots with a vision of 1-hop under the SSYNC scheduler within $O(n)$ epochs. We have also shown that the lower bound of time for gathering n robots on an infinite triangular grid is $\Omega(n)$. So our algorithm is time optimal.

For an immediate course of future research, one can think of solving the gathering problem by considering myopic robots on an infinite triangular grid making the algorithm collision-free (where no collision occurs except at the vertex of gathering) and under an asynchronous scheduler. Another interesting work would be to find out if there is any class of configurations for which gathering on a triangular grid will be solvable even without one axis agreement.

Chapter 3

Gathering with a Faulty Robot on a Finite Grid

In this chapter, we study the GATHERING problem under the assumption that there is one faulty robot in the system. Most previous works on GATHERING with faulty agents require the non-faulty robots to gather at a single point, which may not be the location of the faulty robot. However, this approach has limitations when dealing with faulty robots.

Consider a scenario where n robots are deployed in an environment. These robots are simple and have difficulty exchanging important information when they are no together. Therefore, their main objective is to gather at a single point where they can exchange the information necessary for their tasks. Suppose this information is stored at a specific point or a set of points in the environment. The robots need to gather at one of these specific points to exchange information.

Now, imagine there is a single point of resource in the environment, represented by a robot performing a different task with an independent algorithm, which can move freely until it encounters another robot. This situation can be modeled as a set of n robots, where one robot is "faulty" and moves arbitrarily in any direction until it meets at least one other robot. We call this faulty robot as *resource*.

Given this setup, we cannot define the GATHERING problem with a faulty robot as it was in previous works, because in those scenarios, the non-faulty agents might

not meet the faulty one and thus remain unaware of the information it carries. Therefore, the question we explore in this chapter is: "Can the set of robots gather at the location of this moving resource?" If so, "what is the minimum number of robots necessary and sufficient to achieve this?"

3.0.1 Why Rectangular Grid?

Given this context, it is evident that the environment must be a bounded region; otherwise, it would be impossible to reach the resource. Furthermore, in a bounded region on a plane, a finite number of point robots cannot meet at the resource's location, as there are infinitely many empty points where the resource can move to avoid being caught. Therefore, it is natural to consider this problem within a bounded network.

A finite grid is a widely used network in various fields and has numerous real-life applications, making it a suitable environment for this study. Additionally, this problem can be analogized to cops chasing and catching a robber on the run in a city's street network. Many cities, such as Manhattan, have a grid-like road network, which adds practical relevance to studying this problem on a finite grid.

Notably, if k robots with weak multiplicity detection can gather at the resource's location in some bounded networks, then any number of robots greater than k can gather. This is because once k robots meet with the resource, the resource becomes stationary, and the other robots can simply move to the resource's location. For this reason, we have focused on solving this problem while minimizing the number of robots.

In the following Section 3.1 of this chapter the model of robots, resource and the environment is described. Furthermore, some preliminaries and notations are described after formally defining the problem statement. In Section 3.2 of this chapter, the lower bound of time required and an impossibility result is provided. Then in Section 3.3 we present an algorithm that solves the proposed problem. And finally in Section 3.4 the chapter is concluded with some concluding remarks.

3.1 Gathering Model

In this section we describe the environment in which the problem is considered along with the robot and resource model.

3.1.1 Description of the environment

Let $G = (V, E)$ be a graph embedded on an euclidean plane where $V = \{(i, j) \in \mathbb{R}^2 : i, j \in \mathbb{Z}, 0 \leq i < n, 0 \leq j < m\}$ and there is an edge $e \in E$ between two vertices, say (i_1, j_1) and (i_2, j_2), only if either $i_1 = i_2$ and $|j_1 - j_2| = 1$ or, $j_1 = j_2$ and $|i_1 - i_2| = 1$. We call this graph a finite *grid* of dimension $m \times n$. Though the graph is defined here using coordinates, the robots has no perception of this coordinates which makes this grid unoriented. A *corner* vertex is a vertex of G of degree two. A vertex is called a *boundary* if either the degree of that vertex is three or the vertex is a corner. G has four corner vertex among which exactly one corner vertex has a *door*. This vertex having a door is called the *door vertex*. Robots can enter the grid by entering through that door. There is a movable *resource* (i.e., another robot with executing another independent algorithm on the grid), initially placed arbitrarily at a vertex g_0 (g_0 is not the door) of G.

3.1.1.1 Robot Model

The robots are considered to be autonomous, anonymous, identical and homogeneous. Also, the robots are considered to be point $\mathcal{OBLOT}$ robots (i.e., robots with no persistent memory). The robots can enter through the door one by one. A robot can distinguish if a vertex is on the boundary or a corner of the grid. Also, a robot can identify the door only if it is on the door vertex. Observe that, if the robots could distinguish the door vertex from any other vertex while located on some arbitrary vertex of the grid, then an orientation of the grid can be agreed upon by the robots. But since that is not the case here there is no such orientation of the grid on which the robots can agree. A robot can distinguish the resource from other robots. Each robot has its local coordinate system but

they do not agree on any global coordinate system.

The robots operate in a *LOOK-COMPUTE-MOVE* (LCM) cycle. In each of the cycles, a robot that was previously idle wakes and does the following phases,

***LOOK*:** In *LOOK* phase a robot takes a snapshot of its surroundings and gets the location of other robots and the resource according to its local coordinate system.

***COMPUTE*:** In this phase a robot performs an algorithm with the locations of resource and other robots as input and as an output of that algorithm it gets the location of a neighboring vertex called the destination point.

MOVE: In *MOVE* phase a robot moves to the destination point through the edge of G joining its current location and destination vertex. It is assumed that no two robots can cross each other through one edge without collision.

After completion of *MOVE* phase, the robot becomes idle until it is activated again.

3.1.1.2 Resource Model

The resource *res* is a movable entity, initially which is placed arbitrarily on a vertex (except the door) of G. The resource moves synchronously along with the activated robots if the scheduler is $\mathcal{FSYNC}$ or $\mathcal{SSYNC}$. Otherwise for $\mathcal{ASYNC}$ scheduler the resource also acts asynchronously. The resource can move one hop in one activation. The movement of the resource *res* is controlled by an adversary. So, after an activation, the resource can be on any one of the neighbours of it's previous position including it. We assume that resource will stay fixed if it meets with at least a robot. Otherwise, it can not stay fixed on a vertex forever. Let T_f be the upper bound of the number of rounds (in case of synchronous schedulers otherwise epochs) that *res* can stay fixed alone on a vertex of G. Also, it is assumed that the resource can not cross a robot on an edge without collision. Now if a robot and the resource collides on an edge then, the colliding robot would carry the resource to its destination vertex and then terminates.

3.1.2 Dynamic Gathering Problem

Let G be a finite grid of dimension $m \times n$. Suppose there is a door in a corner of the grid through which robots can enter the grid. The robots can only identify the door if they are located on it. Consider a movable resource that is placed arbitrarily on a vertex of G. Robots can see the resource. The resource will become fixed if at least one of the robots is on the same vertex with the resource. Now the problem asks the robots to move to the same location as the resource. We call this problem Dynamic Gathering.

Note that, if k robots are sufficient to solve the Dynamic Gathering then so is any $k' \geq k$ agents. This is because the first k robots that enters the grid can first gather at the resources location and makes it stationary then the remaining $k' - k$ robots can just simply move to the location of the stationary resource. Thus here in this chapter our main aim is to solve Dynamic Gathering with minimum number of robots. It is easy to see that when $k = 1$, it is impossible for the robot to move to the resource's location. So, here we consider the case when $k = 2$. We can show that it is impossible to solve Dynamic Gathering if the scheduler is $\mathcal{SSYNC}$ (See Theorem 3.2 in Section 3.2). So, here considering a fully synchronous (i.e., $\mathcal{FSYNC}$) scheduler we investigate the case where $k = 2$. We call this variant *Rendezvous at a known dynamic point* on a finite grid.

3.1.3 Notation and Definitions

For a robot r we denote the resource as res and the other robot as r'. Now we have the following definitions.

Definition 3.1 (Door boundary of a robot)**.** *If a robot r is located on a boundary of the grid on which the door vertex is also located then that boundary is called the door boundary of the robot r and is denoted as $BD(r)$.*

Definition 3.2 (Perpendicular Line of robot r)**.** *For a robot r on a boundary, the straight line perpendicular to $BD(r)$ passing through r is called the perpendicular line of robot r. It is denoted as $PD(r)$.*

Definition 3.3 (Distance from resource along $BD(r)$)**.** *Distance of the resource res along boundary $BD(r)$ is defined as the hop distance of robot r from the vertex v on $BD(r)$ such that the line joining v and res is perpendicular to $BD(r)$. We denote this distance as $dist(r)$ for a robot r on $BD(r)$.*

Definition 3.4 (InitGather Configuartion)**.** *A configuration $\mathcal{C}$ is called a* Init-Gather Configuartion *if:*

1. *two robots r and r' are not on same line.*
2. *there is a robot r such that r and the resource res are on a grid line (say L).*
3. *the perpendicular distance of the other robot r' to the line passing through res and perpendicular to L is at most one.*

In the following Fig. 3.1 and Fig. 3.2 we have mentioned the entities we have defined above.

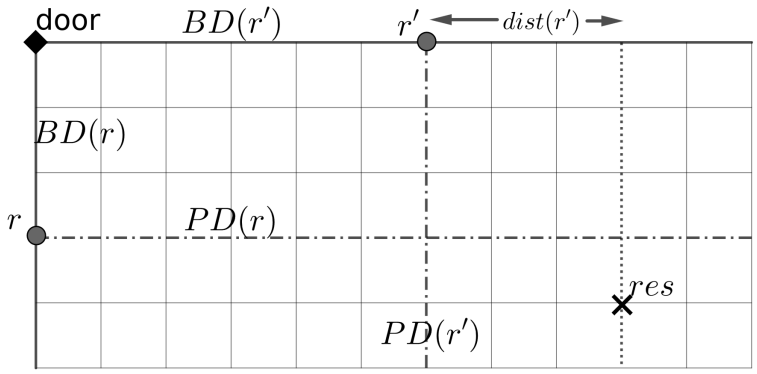


Figure 3.1: Diagram of a configuration mentioning $BD(r)$, $BD(r')$, $PD(r), PD(r')$ and $dist(r')$.

3.2 Lower Bound of Time and Impossibility

In this section, we will discuss the lower bound of time required to solve the problem of rendezvous on a known dynamic point on a finite grid of dimension

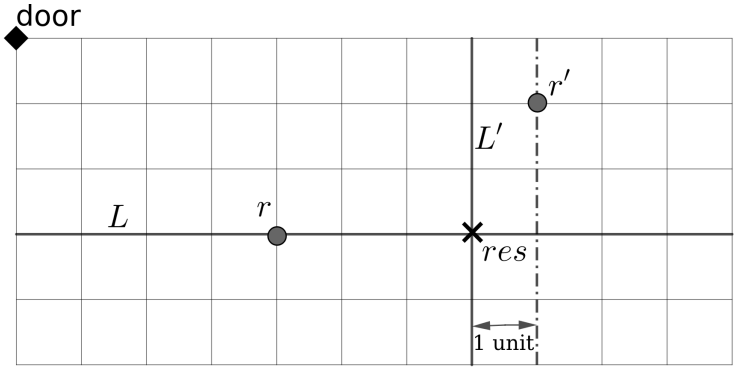


Figure 3.2: Diagram of an InitGather Configuration

$m \times n$. Also, we will prove an impossibility result which will justify our assumption of considering a fully synchronous scheduler to solve this problem. But first, let us define "*epoch*". An epoch is a time interval within which each robot in the system has been activated at least once. In the case of a fully synchronous scheduler, an epoch is equivalent to a round but for other schedulers, an epoch interval is finite but unpredictable. Now in the following theorem, we will discuss the time lower bound of solving rendezvous at a known dynamic point on a finite grid.

Theorem 3.1. *Any algorithm that solves rendezvous at a known dynamic point on a finite grid of dimension $m \times n$ takes $\Omega(m+n)$ epochs in the worst case.*

Proof. Let us consider the scheduler to be a fully synchronous scheduler. Thus an epoch is equivalent to a round. Consider the following diagram (Fig. 3.3) where after each T_f consecutive rounds, the resource changes its location from either P to Q or Q to P. This implies after entering from the door vertex the robots must meet the resource either in vertex P or in vertex Q. Now from the door vertex, the shortest path to P or Q is of length $m+n-1$. So to meet at either P or Q with the resource, each robot must travel through a path of length at least $m+n-1$. Now since in a round a robot can only move a path of length one, to travel a path of length $m+n-1$ at least $m+n-1$ round i.e epoch is necessary to solve this problem. Hence the result. □

Now we will discuss the impossibility result in the next theorem.

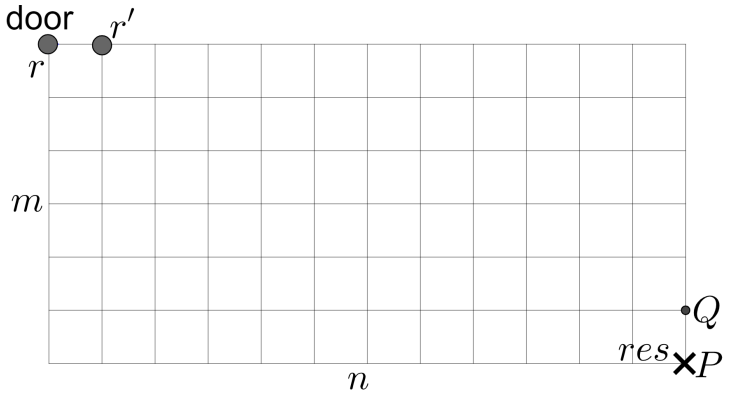


Figure 3.3: from the door vertex to reach P or Q the robot r needs to travel at least a path of length $m + n - 1$.

Theorem 3.2. *No algorithm can solve the problem of rendezvous on a known dynamic point on a finite grid of dimension $m \times n$ if the scheduler is semi-synchronous.*

Proof. Let there is an algorithm $\mathcal{A}$ such that after finite execution of which two robots on a finite grid of dimension $m \times n$ meet at the location of the dynamic resource. Let $m, n > 2$. Also, let t be the round such that after completion of which at least one robot reaches the location of the resource and terminates.

Let no robot is adjacent to the resource at the beginning of round t. This implies at the beginning of round t, the resource has at least two empty neighbor vertices. Now let the adversary activates only one robot during this round. Thus, even if the activated robot moves to one of the resource's empty adjacent vertex, another empty vertex remains empty. So even if the resource has to move during round t it can always find an empty vertex to move that remains empty after the completion of the round. Hence after completion of round t, no robot can move to the location of the resource. Thus we reach a contradiction. Now, let exactly one robot is adjacent to the resource res at the beginning of round t. Then, at least res has one empty vertex which is not reachable by the adjacent robot in one round. So, if the adversary activates only the adjacent robot, say r, and res moves to the empty vertex not reachable by r then again we reach a contradiction. Hence both the robots must be adjacent to the resource at the beginning of round

t. Now if the resource is not at the corner and both the robots are adjacent to the resource at the beginning of round t then, the resource must have an empty adjacent vertex that is not reachable by the robots in one round. Thus if the resource moves to that vertex during round t, we again reach a contradiction.

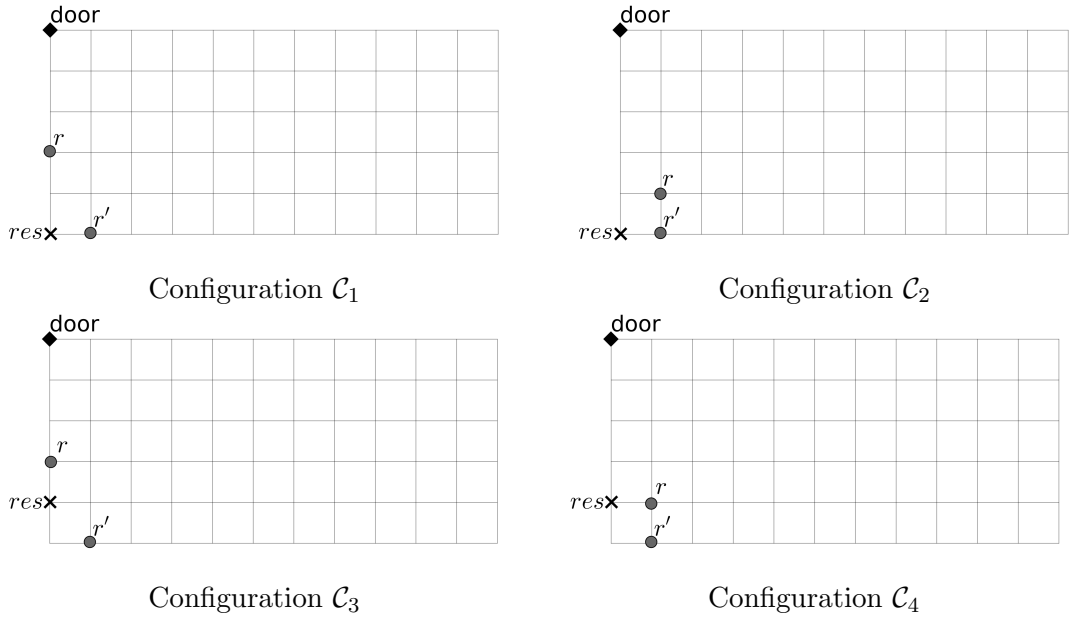


Figure 3.4: Some examples of configuration $\mathcal{C}_{corner-1}$.

Now if we can prove that the configuration (say, $\mathcal{C}_{corner}$) where the resource is at a corner and both the robots are adjacent to it, is never formed then we are done. Let the adversary always activates only one robot in a particular round. Now, if possible let at the beginning of round t, the configuration is $\mathcal{C}_{corner}$. This implies the configuration, say, $\mathcal{C}_{corner-1}$, that was formed just before $\mathcal{C}_{corner}$ must be one of $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$ or $\mathcal{C}_4$ (Fig.4). Since adversary is compelled to activate only one robot in a particular round, so in configuration $\mathcal{C}_{corner-1}$ one the robot must be adjacent to the corner vertex. Without loss of generality let r' be that robot. Note that, in $\mathcal{C}_1$ and $\mathcal{C}_2$ res did not move to form $\mathcal{C}_{corner}$ and, in $\mathcal{C}_3$ and $\mathcal{C}_4$ res had to move to form $\mathcal{C}_{corner}$.

Note that in all of these configurations, there is only one robot that is adjacent to the resource. If the adversary activates the adjacent robot then, in all of these configurations the resource can find an empty adjacent vertex that is not a corner

and remains empty even after the move of the resource. Thus from any of the four configurations $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$ and $\mathcal{C}_4$, $\mathcal{C}_{corner}$ is not formed. Hence we arrive at a contradiction. Thus $\mathcal{C}_{corner}$ will never be formed and hence the result. □

This justifies the necessity of a fully synchronous scheduler to solve this problem. In the next section assuming a fully synchronous scheduler, we have provided an algorithm that solves this problem of rendezvous on a known dynamic point on a finite grid.

3.3 A Solution for Dynamic Gathering

It is quite obvious to observe that without the help of the other robot, a robot can not independently reach the location of the resource if the resource is controlled by an adversary. So to solve this problem the two robots must work together collaboratively and push the resource toward a corner. This is the main idea that is used to develop the proposed algorithm. Now since there is no agreement on the coordinates of the robots and the robots are oblivious, the main challenge here is to agree on the direction for the robots to move.

The rendezvous algorithm Dynamic Rendezvous, proposed in this section is executed in three phases. Entry Phase, Boundary Phase and Gather Phase. In the Entry Phase, the robots move in the grid one by one through the door vertex. This phase ends when the robots are located on the two adjacent vertices of the door vertex. Then in Boundary Phase the robots move along their corresponding boundary to form a special kind of configuration called Init-Gather Configuration (Definition. 3.4). Then in Gather Phase the robots move maintaining the InitGather Configuration and pushing the resource to a corner.

In the first two phases, the agreement on the direction of movement for the robots is constructed from the fact that the robots know the location of the resource and identify vertices on the boundaries and corners of the grid. In these two phases, it is ensured that the robots are on the two boundaries of the grid of which the door

is a part and always remain on their corresponding boundaries. In the GATHER PHASE though, the robots move inside the grid leaving its boundary. In this situation as the robots are not on boundaries, they can not decide on a specific boundary for agreement. In this scenario, the agreement on the direction comes from the fact that at least one robot must be on a line along with the resource during each round of this phase. After this brief overview of the algorithm let us describe it in detail. The algorithm DYNAMIC RENDEZVOUS is as follows.

Algorithm 2: Dynamic Rendezvous

```
1 Input: A configuration C.
2 Output: A destination point of robot r.
3 if a robot, say r, is at a corner ∧ no robot terminates ∧ (there is no
    other robot on the grid ∨ another robot is adjacent to r ) then
4 |  Execute ENTRY PHASE;
5 else
6 |  if C is INITGATHER CONFIGURATION then
7 |  |  Execute GATHER PHASE;
8 |  else
9 |  |  Execute BOUNDARY PHASE;
```

The three phases are described in more detail in the following subsections.

3.3.1 Entry Phase

The first phase is called the ENTRY PHASE (Algorithm 3). During this phase, both the robots enter through the door vertex one by one into the grid G. A robot on the door vertex first checks if it can see another robot already on the grid. If it does not find any other robot on the grid, it moves to any of the two adjacent vertices of the door vertex. On the other hand, if there is already a robot on an adjacent vertex of the door vertex then, the robot on the door vertex moves to the other adjacent vertex of the door vertex. Note that, a robot on one adjacent vertex of the door vertex does not move and does not start executing any other phases if it sees another robot in the corner adjacent to it.

Algorithm 3: Entry Phase for robot r

```
1 if r is on door vertex then
2     if no other robot on boundary then
3         move through any edge on the boundary;
4     else
5         move through the edge on the boundary where there is no other
            robot;
6 else
7     if the other robot r' is at a corner then
8         does not move;
```

The Entry Phase ends when both robots are at the two distinct adjacent vertices of the door vertex in the corner. After the Entry Phase the robots will check if the configuration is an InitGather Configuartion or not. If the configuration is not an InitGather Configuartion then the robots execute the Boundary Phase, otherwise, they execute the Gather Phase.

3.3.2 Boundary Phase

The Boundary Phase starts after the end of the Entry Phase when the configuration is not an InitGather Configuration and no robots are at corners. In the initial configuration of Boundary Phase, both the robots are at the boundary and on the two distinct adjacent vertices of the door vertex. It is ensured in the algorithm of this phase (Algorithm 4) that no robot moves to corner during this phase. So, from this phase a robot can never initiate the Entry Phase again and also the corresponding boundary of a robot stays the same during this phase. Boundary Phase only terminates when the configuration becomes an InitGather Configuration or both the robots reach the location of the resource.

In this phase, a non terminated robot say r first checks if it can see any other robot on the grid. If it does not see any other robot on the grid that implies another robot, say r', has already reached the location of res. In this case, r simply moves towards the location of the resource r avoiding any corner vertices.

When a robot reaches the location of the resource it terminates. On the other hand, if r sees another robot r' on the grid then, r first finds out the adjacent vertex on its corresponding boundary which is nearest to the resource. Note that, there can be two such vertices only if r and the resource are on the same line $PD(r)$ but in this case r does not move. Now if v is unique then r calculates the distance from the resource res for the robots r and r' along $BD(r)$ and $BD(r')$ respectively. If both are non zero then, r finds out if v is a corner or not. If v is not a corner and the other robot r' is not adjacent to some corner on its corresponding boundary, then, r moves to v. Otherwise, if r' is adjacent to a corner on its boundary then, r moves to v only if distance of the resource along $BD(r)$ is not equal to one. This technique is required to avoid a livelock scenario. There can be another configuration where distance of the other robot r' along $BD(r')$ is zero but distance of r along $BD(r)$ is strictly greater than one. In this case the robot r simply moves to the vertex v. Note that in this case v can not be a corner vertex as $dist(r) > 1$. So, during this phase no robot ever moves to a corner and this phase terminates only when both the robot reaches the location of resource or an InitGather Configuration is achieved. Now, we have to ensure that the Boundary Phase terminates within finite rounds. But, before that we need to define a quadrant and proof some results which will be needed to proof the termination of Boundary Phase within $O(T_f \times \max\{m, n\})$ rounds.

Definition 3.5 (Quadrant)**.** *The grid G is divided into four segments by the two lines $PD(r)$ and $PD(r')$. Each of these segments are called a quadrant.*

The quadrants on the northeast, northwest, southeast, and southwest are denoted as R_{NE}, R_{NW}, R_{SE} and R_{SW} respectively (Fig. 3.5). Note that the intersection of any two quadrant is a section of either $PD(r)$ or $PD(r')$. At the beginning of the Boundary Phase, quadrant R_{NW} is a 2×2 grid, R_{SW} is a $(m-1) \times 2$ grid, R_{NE} is a $2 \times (n-1)$ grid and R_{SE} is a $(m-1) \times (n-1)$ grid. At the beginning of Boundary Phase, the resource res must be either inside or on one of R_{NE}, R_{SW} and R_{SE}.

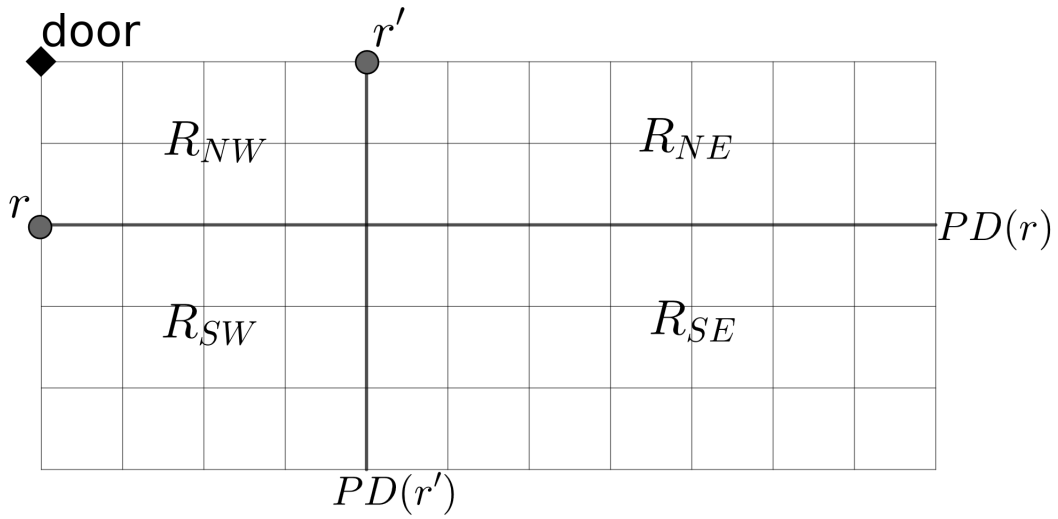


Figure 3.5: Four quadrants divided by $PD(r)$ and $PD(r')$

Algorithm 4: Boundary Phase for robot r

```
1  if on the same vertex with res then
2      terminate;
3  else
4      if r' is on the same vertex with res then
5          move to res along any shortest path avoiding corner in between;
6      else
7          v ← adjacent vertex on BD(r) which is near res along the
             boundary;
8          if dist(r) ≠ 0 and dist(r') ≠ 0 then
9              if v is not corner then
10                 if r' is not adjacent to a corner then
11                     move to v;
12                 else
13                     if dist(r) ≠ 1 then
14                         move to v;
15         else if dist(r') = 0 and dist(r) > 1 then
16             Move to v;
```

Lemma 3.1. *If* Boundary Phase *never terminates then, the resource must crosses or moves on to any one of $PD(r)$ or $PD(r')$ at least once within $O(T_f \times \max\{m, n\})$ rounds.*

Proof. If BOUNDARY PHASE never terminates then, no robot ever reaches the resource res and INITGATHER CONFIGURATION is never formed during execution of the BOUNDARY PHASE. If possible, let our claim is false, i.e., the resource res never moves onto and never crosses $PD(r)$ and $PD(r')$ during the execution of BOUNDARY PHASE. So res can never be on $PD(r)$ or on $PD(r')$ at the beginning of BOUNDARY PHASE. Now there are three cases depending on the location of res at the beginning of the BOUNDARY PHASE. The cases are as following:

At the beginning of BOUNDARY PHASE the res is on some vertex of,

I. $R_{SE} \setminus \{PD(r) \cup PD(r')\}$

II. $R_{NE} \setminus \{PD(r) \cup PD(r')\}$

III. $R_{SW} \setminus \{PD(r) \cup PD(r')\}$

Case I: Let res is on some vertex of $R_{SE} \setminus \{PD(r) \cup PD(r')\}$ at the beginning of the BOUNDARY PHASE. According to our assumption res must stay on some vertex of $R_{SE} \setminus \{PD(r) \cup PD(r')\}$ for infinitely many consecutive rounds.

Let at the beginning of some round t, the dimension of R_{SE} is $m' \times n'$, where $m', n' > 2$. then according to algorithm 4, the dimension of R_{SE} decreases to $(m'-1) \times (n'-1)$ (Fig. 3.6). So, within $\min\{m-3, n-3\}$ rounds dimension of R_{SE} becomes either $2 \times n_1$ or, $m_1 \times 2$, where $2 \leq m_1 < m$ and $2 \leq n_1 < n$.

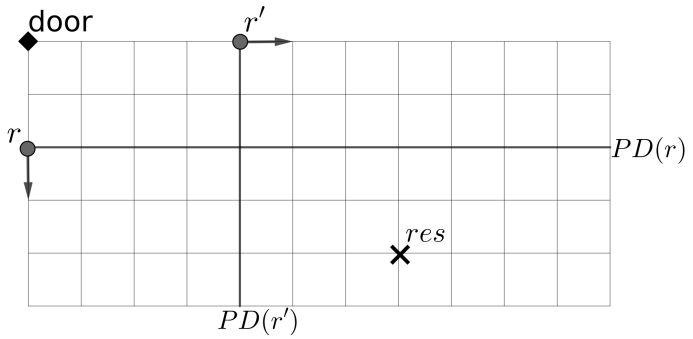


Figure 3.6: Both height and width of R_{SE} decreases.

without loss of generality let, dimension of R_{SE} is $2 \times n_1$ and $2 < n_1 < n$ at the beginning of some round t'. In this scenario exactly one robot is adjacent to a cor-

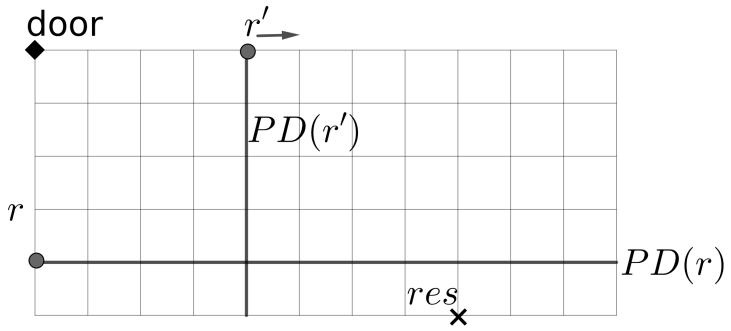


Figure 3.7: Height of R_{SE} remains same but width decreases.

ner. Without loss of generality let r be the robot whose adjacent vertex on $BD(r)$ is corner. In this scenario $dist(r) = 1$ as res is on $R_{SE} \setminus \{PD(r) \cup PD(r')\}$. Now, if $dist(r') > 1$ then, after completion of round t' dimension of R_{SE} decreases to $2 \times (n_1 - 1)$ as r' moves towards res along $BD(r')$. On the other hand if $dist(r') = 1$ at the beginning of round t' then, none of r and r' moves according to algorithm 4. In the worst case, after T_f round during the round $t' + T_f$ the resource res must move. Note that, during this round res can only move parallel to $PD(r)$ and away from r as otherwise the configuration becomes an InitGather Configuration contrary to our assumption. Observe that when res completes the move, $dist(r')$ becomes strictly greater than one and r' moves in the next round and decreases the dimension of R_{SE} to $2 \times (n_1 - 1)$ (Fig. 3.7). So, In the worst case at some round t'', within
$(T_f+1)[\max\{m-3, n-3\} - \min\{m-3, n-3\}] = (T_f+1)[|m-n|]$ rounds the dimension of R_{SE} becomes 2×2. At the beginning of round $t''+1$ the configuration is as follows (Fig. 3.8),

1. res is at the corner which is diagonally opposite to the door vertex.
2. r and r' are adjacent to a corner which is not the door vertex on $BD(r)$ and $BD(r')$ respectively.

In this configuration none of r and r' moves and whenever res moves the configuration becomes an InitGather Configuration contrary to our assumption that Boundary Phase never terminates.

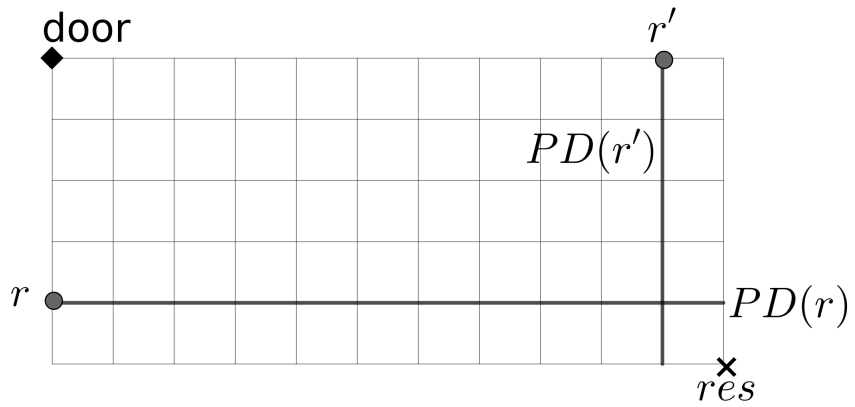


Figure 3.8: In this configuration r and r' does not move. Whenever res moves the configuration becomes an InitGather Configuration

Now for the remaining two cases with similar argument it can be shown that the configuration becomes an InitGather Configuration that leads to a contradiction. So our assumption that res never crosses or moves onto any of $PD(r)$ and $PD(r')$ was wrong. Thus, if Boundary Phase doesn't terminate then, res must move onto or crosses any one of $PD(r)$ or $PD(r')$ within $(T_f + 1)[|m - n|] + T_f$ rounds in the worst case, which is asymptotically equals to $O(T_f \times \max\{m, n\})$.

□

Lemma 3.2. *If* Boundary Phase *never terminates and the res has moved onto or crosses $PD(R)$ ($R \in \{r, r'\}$) at some round (say t), then $dist(R) \leq 1$ from round t on wards.*

Proof. If Boundary Phase never terminates then, no robot ever reaches the resource res and InitGather Configuration is never formed during execution of the Boundary Phase. Without loss of generality, let res have crossed or moved onto $PD(r)$ at round t. So at the beginning of round $t + 1$, $dist(r)$, must be less or equal to one.

Case I: Let res be on $PD(r)$ at the beginning of round $t + 1$. Then $dist(r) = 0$. Now during round $t + 1$, res either moves along $PD(r)$ (horizontally in Fig. 3.5) or, Perpendicular to $PD(r)$ (vertically in Fig. 3.5) or does not move at all. Now

if res moves parallel to $PD(r)$ or does not move at all, then $dist(r)$ remains the same after completion of round $t+1$ according to the algorithm of BOUNDARY PHASE. On the other hand, If res moves Perpendicular to $PD(r)$ during round $t+1$, then $dist(r)$ becomes one after the completion of round $t+1$.

Case II: Let res crosses $PD(r)$ at round t. Then at the beginning of the round $t+1$, $dist(r) = 1$. Now if res moves parallel to $PD(r)$ or does not move at all during round $t+1$ then after the completion of the round, $dist(r)$ either stays one or decreases to zero (Fig. 3.9, Fig. 3.10). Now let res moves Perpendicular to $PD(r)$ during round $t+1$, then if res moves towards $PD(r)$ then $dist(r)$ either remains same as one (as r can move along $BD(r)$, res crosses $PD(r)$) or becomes zero in case r does not move along $BD(r)$ (Fig. 3.11, Fig. 3.12). On the other hand, if res moves away from $PD(r)$ during round $t+1$, then $dist(r)$ remains one after completion of round $t+1$ as r also moves during round $t+1$ towards the direction of res along $BD(r)$.

So after completion of round $t+1$, $dist(r)$ is still less or equal to 1. Now with similar arguments, it is easy to see that if after completion of round $t+i$, $dist(r) \leq 1$, then $dist(r) \leq 1$ after completion of round $t+i+1$ for some natural number i. Hence by Mathematical induction, we can conclude the lemma.

□

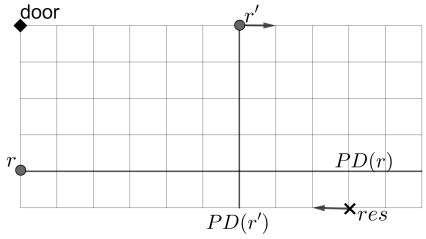


Figure 3.9: $dist(r)$ remains one.

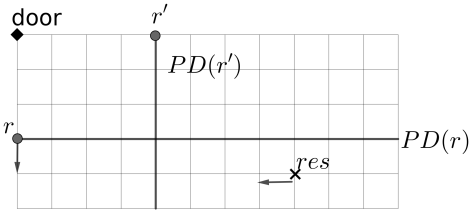


Figure 3.10: $dist(r)$ becomes zero.

Lemma 3.3. *If* BOUNDARY PHASE *never terminates and res has moved onto or crossed $PD(R)$ where $R \in \{r, r'\}$ at some round t, then res never crosses $PD(R')$ from round t on wards (Here $R' = r$ if $R = r'$ and $R' = r'$ if $R = r$).*

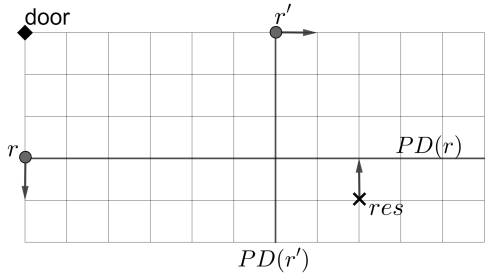


Figure 3.11: $dist(r)$ remains one as res crosses $PD(r)$.

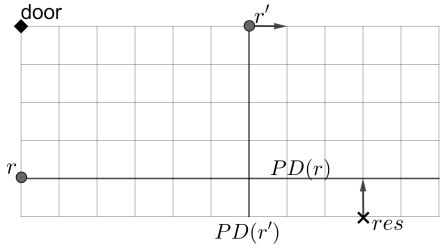


Figure 3.12: $dist(r)$ becomes zero as r does not move.

Proof. If Boundary Phase never terminates then, no robot ever reaches the resource res and InitGather Configuration is never formed during execution of the Boundary Phase. Without loss of generality let res move onto or crosses $PD(r)$ at some round t_0 and $BD(r')$ is the boundary of G at the north (Fig. 3.5). By Lemma 3.2 ,for any round $t > t_0$, $dist(r)$ remains less or equals to one. We have to show that from round t on wards res never crosses or moves onto $PD(r')$. If possible let res moves onto or crosses $PD(r')$ at some round $t_1 > t_0$. Then res must moves parallel to $PD(r)$ or doesn't move at all during round t_1.

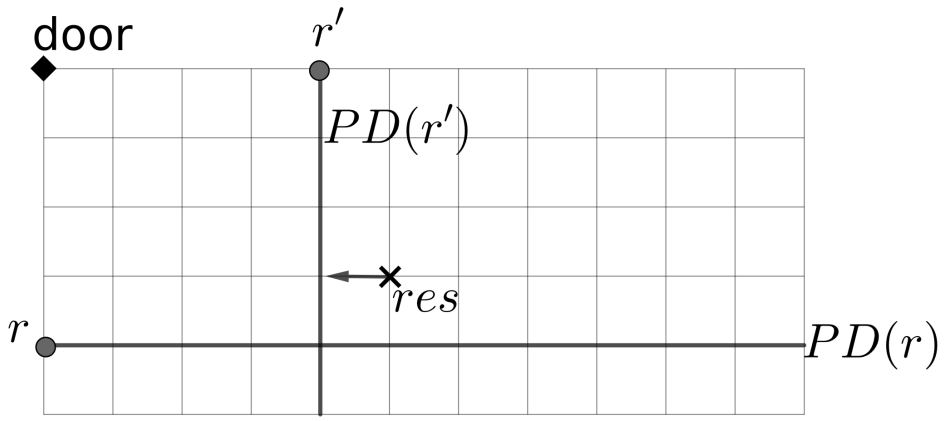


Figure 3.13: res moves onto $PD(r')$ and creates an InitGather Configuration.

Note that, if $dist(r) = 0$ at the beginning of round t_1 then, after completion of the round $dist(r)$ remains zero and $dist(r')$ becomes less than or equals to one. So, after completion of round t_1 the configuration becomes an InitGather Configuration contrary to our assumption. So,let $dist(r) = 1$ at the begin-

ning of round t_1. Now, after completion of the round, if res moves onto $PD(r')$ i.e., $dist(r')$ becomes zero then, the configuration again becomes an InitGather Configuration as $dist(r)$ either remains one or becomes zero. So, res can only cross $PD(r')$ during round t_1. After crossing $PD(r')$, $dist(r')$ becomes one. Now, for res to cross $PD(r')$ without achieving an InitGather Configuration is possible only when r does not move otherwise $dist(r)$ becomes zero and we arrive at an InitGather Configuration contrary to the assumption. Observe that, r does not move during round t_1 only if it is located adjacent to a corner on $BD(r)$. Now at the beginning of round t_1, $dist(r')$ is also one (as it is assumed that it crosses $PD(r')$ during round t_1) (Fig. 3.13). In this scenario r' doesn't move during the round t_1 according to algorithm 4. So, res can't cross $PD(r')$ without moving onto it. As described earlier this leads to an InitGather Configuration which is a contradiction. So, for Boundary Phase to never terminate, if res crosses $PD(R)$ during round t_0 then from round t_0 on wards it never crosses or moves onto $PD(R')$.

□

Now using the above lemmas we prove the following theorem.

Theorem 3.3. *For a grid of dimension $m \times n$, the* Boundary Phase *terminates within $O(T_f \times \max\{m, n\})$ rounds.*

Proof. If possible, let us assume Boundary Phase never terminates. This implies no robot ever moves to the location of res and InitGather Configuration is never formed during the execution of Boundary Phase. From the above three lemmas (i.e., Lemma 3.1, Lemma 3.2 and Lemma 3.3) it can be said that, res must crosses or moves onto $PD(R)$ where $R \in \{r, r'\}$ at some round t for the first time ,within $T_f \times O(\max\{m, n\})$ rounds. Then from round t on wards, $dist(R)$ always remains less or equals to one and res never crosses $PD(R')$ where $R' = r$ if $R = r'$ and $R' = r'$ if $R = r$.

Now without loss of generality let, res has moved onto or crossed $PD(r)$ at some round t for the first time. Also, let, $BD(r')$ is the boundary on the north of

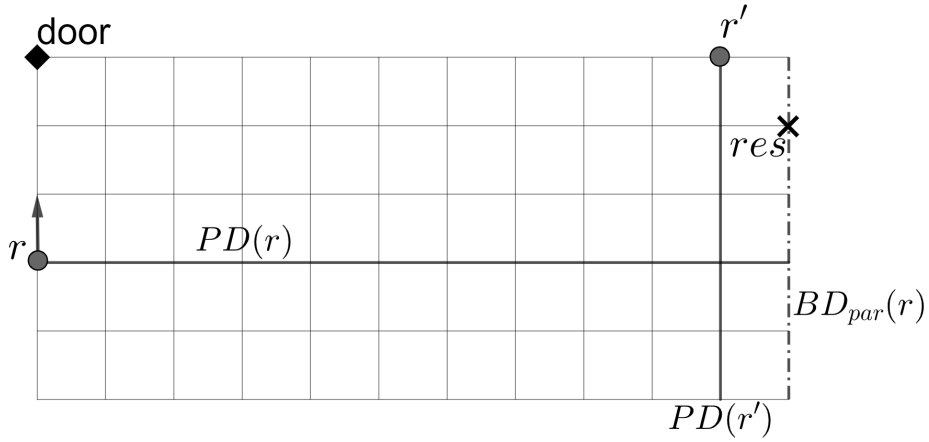


Figure 3.14: dimension of R_{NE} decreases from $m' \times 2$ to $(m'-1) \times 2$

the grid (Fig. 3.5). Then from round t on wards res must lie inside $(R_{NE} \cup R_{SE}) \setminus PD(r')$ (i.e., $dist(r') \geq 1$ from round t on wards). So, in the worst case, r' must move away from the door along $BD(r')$ in each T_f consecutive rounds once, from round t on wards. Let after completion of round t the dimension of $(R_{NE} \cup R_{SE})$ is $m' \times n'$, where the length $n' < n$. Note that after $(n'-2) \times (T_f+1)$ rounds $(R_{NE} \cup R_{SE}) \setminus PD(r')$ is the boundary of the grid, say $BD_{par}(r)$, which is parallel to $BD(r)$ (Fig. 3.14). In this scenario res must be on some vertex of $BD_{par}(r)$. Note that in this configuration r' is adjacent to a corner so it does not move and res can not be on $PD(r)$ as otherwise it would be a InitGather Configuration and Boundary Phase terminates. Now in this configuration res can be either on some vertex of $R_{NE} \setminus \{PD(r) \cup PD(r')\}$ or on some vertex of $R_{SE} \setminus \{PD(r) \cup PD(r')\}$ and also remains so in the upcoming rounds. Let without loss of generality res is on some vertex of $R_{NE} \setminus \{PD(r) \cup PD(r')\}$ at the beginning of some round t. Let dimension of R_{NE} is $m' \times 2$ at the beginning of round t (Fig. 3.14). Then by the similar argument used to prove *Case I.* of Lemma 3.1 we can conclude that if no robot reaches the location of res, within at most $(T_f+1) \times (m'-2) + T_f$ more rounds the configuration becomes an InitGather Configuration, which is a contradiction. Hence our assumption that Boundary Phase never terminates is wrong. And as calculated the Boundary Phase terminates in $O(T_f \times \max\{m,n\}) + (T_f+1) \times (m+n-6) +$

T_f rounds in the worst case which is asymptotically equals to $O(T_f \times \max\{m, n\})$.

□

3.3.3 Gather Phase

Gather Phase starts if none of the two robots reaches the location of res after the termination of the Boundary Phase. Throughout the execution of this phase, the configuration will remain an InitGather Configuration (Lemma 3.5). So, in each round, a robot will lie on the same line (say L) along with res, and the perpendicular distance of the other robot to the line passing through res and perpendicular to L must be at most one. During this phase, if a robot is in the same location with res, it terminates and the other robot moves to the location of res along any shortest path. On the other hand when none of the robots are on the same vertex with res, the robot on L checks if res is adjacent to it. If res is not on its adjacent vertex, it moves towards res along L. Otherwise, if res is on its adjacent vertex then it moves towards res along L only if it sees res is on a corner and the other robot, say r', is also on another adjacent vertex of res. Now if the robot is not on any line along with the resource res, (i.e., perpendicular distance of the robot to the line through res and perpendicular to L is one) then, the robot will move parallel to L towards res. The pseudo code of Gather Phase is as follows.

Before proving the correctness of the Gather Phase, let us discuss some notations used in the following proofs.

Let, r be the robot which is on the same line along with the resource res in an InitGather Configuration. This line is denoted as L. We denote the line passing through res and perpendicular to L as L'. Now by L^1 ans L^{-1} we denote the lines parallel to L' and one hop distance apart from L'. L^1 is the line furthest to r compared to L^{-1}. Observe that r' can be on any one of L^{-1}, L', L^1 in an InitGather Configuration.

Algorithm 5: Gather Phase for robot r

```
1  if r is on same vertex with res then
2      terminate;
3  else
4      if r' is on the same vertex with res then
5          move to res along any shortest path avoiding door vertex;
6      else
7          if r is on a line L with res then
8              if res is not adjacent to r then
9                  move towards res along L;
10             else
11                 if res is at a corner and r' is adjacent to res then
12                     move towards res along L;
13         else
14             move parallel to L towards res;
```

Now we can prove the following lemma.

Lemma 3.4. *By executing Dynamic Rendezvous algorithm two robots r and r' never moves to same line from an* InitGather Configuration.

Proof. Let at the beginning of some round t, the configuration is an InitGather Configuration. Observe that r and r' are not on the same line at the beginning of the round t. If possible let after completion of round t, r and r' moves to the same line. Without loss of generality let, r be the robot on the same line L along with res. Then, r' can be on any one of L^{-1}, L', L^1 at the beginning of round t. Now, we have three cases depending on the location of r'.

Case I: Let r' is on L' at the beginning of round t. Then for being on the same line after the completion of the round either r moves to L' or, r' moves to L. Without loss of generality, let r moves to L' during round t. This implies at the beginning of round t, r is adjacent to res on L (Fig. 3.15). Now according to the algorithm 5, r doesn't move during round t, a contradiction.

Case II: Let r' is on L^1 at the beginning of round t. Then during round t, r' moves parallel to L to L'. So to be on the same line after completion of the round,

r must move onto L' during this round (Fig. 3.16). But for the same reason used in *Case I*, r can not move during round t which leads to a contradiction again.

Case III: Let r' is on L^{-1} at the beginning of round t. According to algorithm 5, r' moves to L' during this round. So for being on the same line r must move to L' too during round t. For this to happen, r must be on the line L^{-1} along with r' at the beginning of round t (Fig. 3.17). But that is a contradiction as at the beginning of round t the configuration is an InitGather Configuration.

For all of this cases we reach a contradiction assuming that after completion of round t, r and r' moves to the same line. Hence the lemma.

□

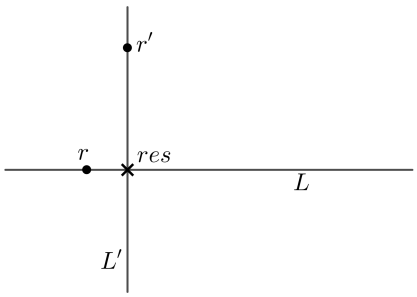


Figure 3.15: r' is on L'. r doesn't move to L'.

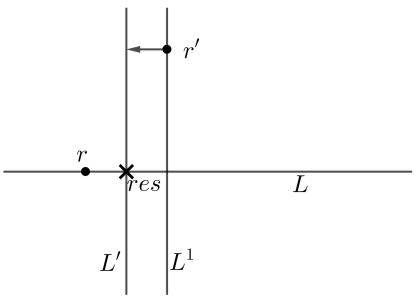


Figure 3.16: r' is on L^1.r doesn't move to L'.

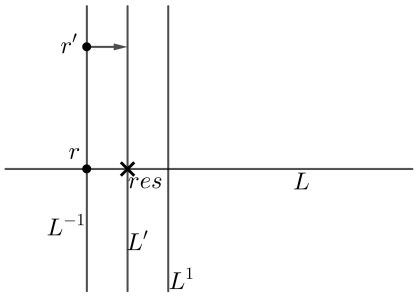


Figure 3.17: This configuration is not possible.

Next we have the following lemma.

Lemma 3.5. *If no robots terminate, an* InitGather Configuration *remains an* InitGather Configuration *after one execution of the algorithm Dynamic Rendezvous.*

Proof. Let at the beginning of some round t the configuration is an InitGather Configuration. Let r be the robot on the same line L along with res. Now, the other robot r' can be on any one of L^{-1}, L' and L^1 at the beginning of round t. By the Lemma 3.4, it can be made sure that, r and r' does not move onto same line after completion of round t. So, to prove this lemma it is sufficient to show that, after completion of round t, one of r and r' is on the same line, say L_d, along with res and the perpendicular distance of the other robot to the line L'_d is at most one where, L'_d is the line perpendicular to L_d and passing through res. Now based on the movement of res during round t, we have three cases as following.

I. res moves along L.

II. res moves along L'.

III. res does not move.

Case I: Let res moves along line L during round t. Now even if r moves in round t, it stays on L according to algorithm 5. Now, let r' can be on L'or, L^1 or, L^{-1} at the beginning of round t. Irrespective of the position, r' reaches either on L^1 or on L^{-1} in the new configuration after completion of the round t (Fig. 3.18 and Fig. 3.19). Thus for this case the configuration remains an InitGather Configuration.

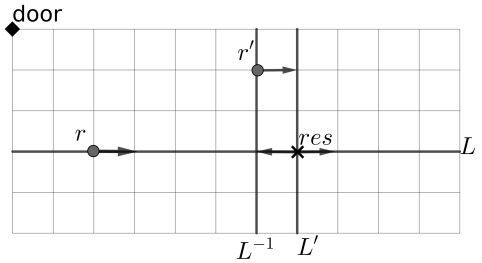


Figure 3.18: r' moves to L' and res moves to either L^{-1} or L^1 during round t. r stays on L along with res.

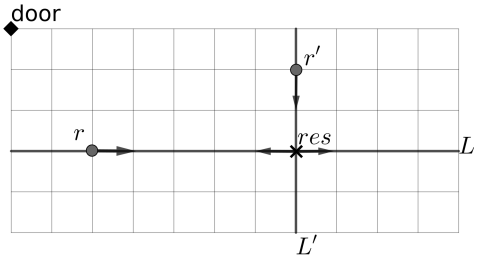


Figure 3.19: r' stays on L' but res moves to either L^{-1} or L^1 during round t. r stays on L with res.

Case II: Let res moves along L' during the round t. In this case irrespective of the location of r' at the beginning of round t (i.e., L^{-1}, L', L^1), it moves onto L'

along with res in the new configuration after completion of the round. Let L'' be the line perpendicular to L' on which res moves after completion of round t. Also note that during round t, r stays on L even if it moves. Now since, L'' is parallel to L and one hop away from L, the perpendicular distance of r to L'' becomes one after completion of the round t (Fig. 3.20 and Fig. 3.21). Thus the new configuration remains an INITGATHER CONFIGURATION after completion of round t.

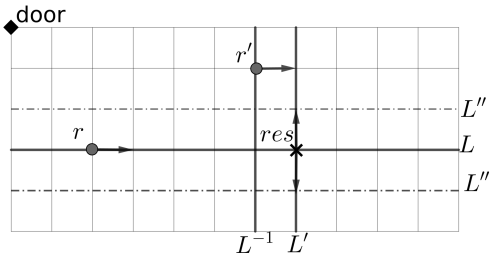


Figure 3.20: res moves to L'' along L', r stays on L and r' moves to L' from L^{-1}.

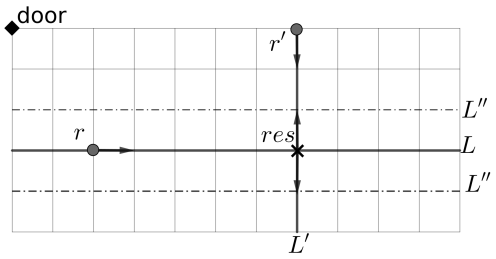


Figure 3.21: res moves to L'' along L', r stays on L and r' stays on L'.

Case III: Let res does not move during round t. Then, irrespective of the location of r' at the beginning of round t, it reaches L' along with res after completion of the round. Also, even if r moves it stays on L along with res after completion of round t. So, after completion of round t, in the new configuration r stays on L along with res and r' stays on L'. Hence the new configuration is again an INITGATHER CONFIGURATION.

For all of the above cases, the configuration remains an INITGATHER CONFIGURATION and we have the lemma.

□

Let at the beginning of a particular round during the GATHER PHASE a robot r is on the same line L along with the resource, res. Let us define two lines, firstly, L_1 passing through r and perpendicular to L, and secondly L_2, passing through the vertex of the other robot r' and parallel to L. Note that the lines L_1 and L_2 divides the entire grid into one or more rectangles. The rectangle inside of which

the resource res is located is called the "Containing Rectangle" and it is denoted as R_{Con} (Fig. 3.22). Observe that, at the beginning of the first round of GATHER PHASE, L_1 is $BD(r)$ and L_2 is $BD(r')$ and $R_{Con} = G$.

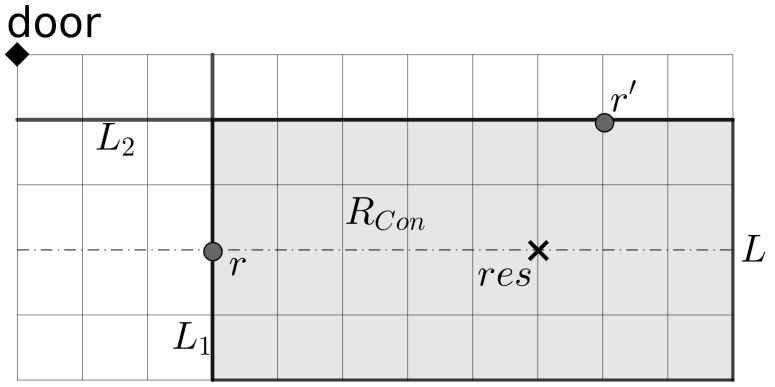


Figure 3.22: shaded region is R_{Con}

Lemma 3.6. *The resource res never moves onto L_1 or L_2 during the* GATHER PHASE *without colliding with any robot.*

Proof. Initially at the beginning of GATHER PHASE, resource res must not be on $BD(r)$ or $BD(r')$. Otherwise, res must have crossed or moved onto both $PD(r)$ and $PD(r')$ during the BOUNDARY PHASE, which is not possible due to claim (3). So, at the beginning of GATHER PHASE, res is not on any of L_1 or L_2.

If possible let during some round t of GATHER PHASE, res moves onto either L_1 or L_2 for the first time, without colliding with any robots. Let, res is on the same line L with r at the beginning of round t. Now, r must be on the vertex $L_1 \cap L$ and r' is on any of the three vertices, $L_2 \cap L^{-1}$, $L_2 \cap L'$ and $L_2 \cap L^1$ at the beginning of round t (Fig. 3.23). Now we have two cases,

Case I: Let res moves onto L_1 during round t. For this to happen, res must be on the adjacent vertex of $L_1 \cap L$ (i.e, location of r) at the beginning of round t and it must move along L during round t.So, res reaches $L_1 \cap L$ after completion of the round. Now since r is adjacent to res on L at the beginning of round t, it does not move during round t (Algorithm 5) and stays on $L_1 \cap L$ after completion

of the round. So after completion of round t, res collides with r contrary to our assumption.

Case II: Let res moves onto L_2 during round t. For this to happen, res must move along L' and reaches $L' \cap L_2$ after completion of the round. Now, irrespective of the position of r' at the beginning of round t, it reaches $L_2 \cap L'$ (Algorithm 5) after completion of the round (Fig. 3.23). Since both res and r' reaches $L_2 \cap L'$ after completion of round t, they collides contradicting our assumption.

Since in both cases we reach contradiction our assumption must be incorrect. So, res never moves onto L_1 or L_2 without colliding with a robot during GATHER PHASE. □

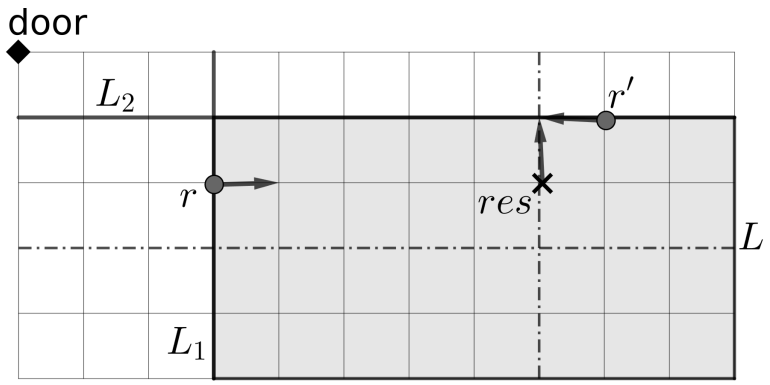


Figure 3.23: res collides with r' if moves onto L_2

Corollary 3.3.1. *During the* GATHER PHASE*, the resource res never moves outside R_{Con} without colliding with a robot.*

Proof. If res moves out of R_{Con} then it must cross either L_1 or L_2 without moving onto them.

Now the resource, res never crosses L_1 without colliding with r. Also, for the same reason, res never crosses L_2 while r' is also on the same line along with res. So let us assume r' is not on the same line along with res at the beginning of some round t during which res crosses L_2. So, during round t r' must move along L_2 . So, the line L_2 does not shift after the completion of round t. This implies

res must move onto L_2 to cross it which is not possible due to Lemma 3.6. Hence res never moves out of R_{Con}. □

Let at the beginning of the first round of GATHER PHASE, R_{Con} be a $m_1 \times n_1$ grid. Let the height and width of R_{Con} be m_1 and n_1 respectively where, both $m_1 > 2$ and $n_1 > 2$. We will prove that within $T_f + 1$ rounds either both m_1 and n_1 decrease or one of m_1 and n_1 decreases and the other one stays the same.

Lemma 3.7. *During the* GATHER PHASE*, if both height and width of the R_{Con} be more than two and none of the robot terminates then, within $T_f + 1$ rounds either both height and width of R_{Con} decreases or one of height or width decreases and the other remains same.*

Proof. Let at the beginning of some round t during the GATHER PHASE, r be a robot on the line L along with the resource, res. The lines L_1 (line passing through r and perpendicular to L)and L_2 (line passing through the other robot, r' and parallel to L) and the boundaries that do not contain the door vertex forms a rectangle R_{Con}. We have proved that res always remains contained within $R_{Con} \setminus \{L_1 \cup L_2\}$ and never moves out of it if none of the robots are terminated. Let the dimension of R_{Con} at the beginning of round t be $m_1 \times n_1$ where both m_1 and n_1 are greater than two. Thus even if res is at a corner at the beginning of round t, both r and r' are not adjacent to res. Thus during round t, no robot moves to the location of res.

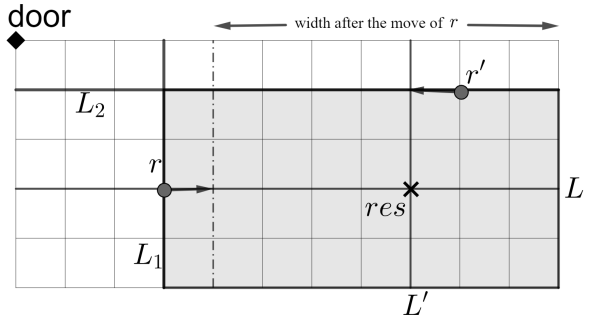


Figure 3.24: Only width of R_{Con} decreases and height remains same.

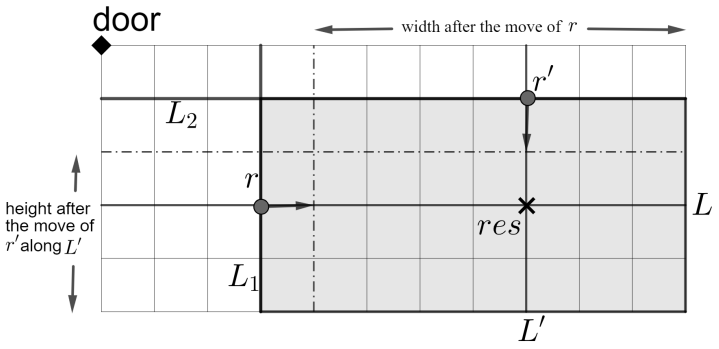


Figure 3.25: both height and width of R_{Con} decreases.

Case I: Let at the beginning of round t, r is not adjacent to res. Also, without loss of generality let the length of the side of R_{Con}, which is parallel to L at the beginning of round t is the width of R_{Con}. Now according to the algorithm, r moves along L towards res i.e towards the direction of the interior of R_{Con}. Hence L_1 shifts towards the interior of R_{Con}. So the width of R_{Con} decreases during round t. Now, if r' is not on the line L' (line passing through res and perpendicular to L) or, adjacent to res on the line L' at the beginning of round t then, r' moves along L_2 (Fig. 3.24) or does not move at all. In both of these cases, the height of R_{Con} remains the same after the completion of the round. on the other hand if at the beginning of round t, r' is on L' along with res and not adjacent to r' then, r' moves along L' towards the direction of res (Fig. 3.25). Note that in this case L_2 also shifts towards the interior of R_{Con} and decreases the height of R_{Con} after completion of round t.

So we have shown that if r, a robot on a line L with res is not adjacent to res then either both height and width decrease or only width decreases in one round.

Case II: Let, res is adjacent to r on L at the beginning of round t then, r will not move along L. It is assumed that res will not stay at the same location for more than T_f consecutive rounds. Note that since both height and width are more than two, res gets an empty vertex to move. Now in the worst case during the round $t+T_f$, res must have moved either along L or along L'. Note that res can not move towards r along L as it would end up colliding with r.

Case II(a): Let during the round $t+T_f$, res moves along L opposite to r then,

at the beginning of round $t + T_f + 1$, r and res are not adjacent along L. Hence during this round, either only the width of R_{Con} decreases and height remains the same, or both height and width of R_{Con} decrease by a similar argument as in *case I*.

Case II(b): Now let us consider the case where r is adjacent to res on L at the beginning of round $t + T_f$ but res moves perpendicular to L i.e., along L' during the round $t + T_f$.

If at the beginning of round $t+T_f$, r' was on L' then, by the same argument as in *Case I* and *Case II(a)* we can conclude that in the worst case, after completion of round $t+2T_f+1$, the height of R_{Con} must decrease while width either remains same or also decreases.

Let us now consider r' is not on L' at the beginning of round $t + T_f$. In this case, after completion of round $t + T_f$, r' and res must be on the line L' and hence according to the same argument as above cases in the worst during round $t + 2T_f + 1$ either height and width of R_{Con} both decrease or height decreases while the width remains same.

□

By Lemma 3.5 we can conclude that, after one execution of the Gather Phase, if no robot is terminated then in the next round, Gather Phase will be executed again. Also, from the Lemma 3.7 it is evident that if the dimension of the R_{Con} is $m_1 \times n_1$ where both $m_1 > 2$ and $n_1 > 2$ then, in the worst case in every $2T_f + 1$ round, the height or the width of the configuration decreases and none of them ever increase. So within $O(T_f \times (m + n))$ there will be a round (say t_0) when either the height or the width of R_{Con} becomes two. Without loss of generality let the dimension of R_{Con} be $m_1 \times 2$, at the beginning of round t_0, where $m_1 > 2$. Now we claim the following lemma.

Lemma 3.8. *If the dimension of R_{Con} is $m_1 \times 2$ (resp. $2 \times n_1$) and no robot terminates then, within $(T_f+1)(m_1-1)$ (resp. $(T_f+1)(n_1-1)$) rounds dimension of R_{Con} becomes 2×2.*

Proof. Without loss of generality let the dimension of R_{Con} be $m_1 \times 2$ at the beginning of some round (say t_0) where, $m_1 > 2$. This implies exactly one of the height or width of R_{Con} is two. Without loss of generality let the width is two and height be $m_1 > 2$. Thus R_{Con} consists of exactly two lines perpendicular to the width. One of these two lines (say, L) is a boundary of G which does not contains the door vertex and the other one is the line parallel and adjacent to it (Say L_2). Since no robots are terminated, the configuration at the beginning of round t_0 is an InitGather Configuration (Lemma 3.5). So, each of these two lines L and L_2 contains exactly one robot. Let without loss of generality r be on the line L and r' is on L_2. Now by Lemma 3.6, res must be on L and below L_1. Also, the distance of r' to the line perpendicular to L and passing through res (say L') is at most one as the configuration is an InitGather Configuration at the beginning of round t_0. So if during round t_0, res moves perpendicular to L it must collides with r' and r' terminates contrary to the assumption. So let us consider res either move along L or does not move at all during round t_0 (Fig. 3.26). Now, in the worst case within $T_f + 1$ rounds the height of R_{Con} must decrease by one unit. Also, since $m_1 > 2$, at the beginning of round t_0, if res is at a corner, r is not adjacent to res and hence r' can only move parallel to L_2. So unless $m_1 = 2$, width of R_{Con} remains two. Thus in the worst case, the height of R_{Con} becomes two while the width still remains two in $(T_f + 1)(m_1 - 1)$ rounds. Hence the lemma. □

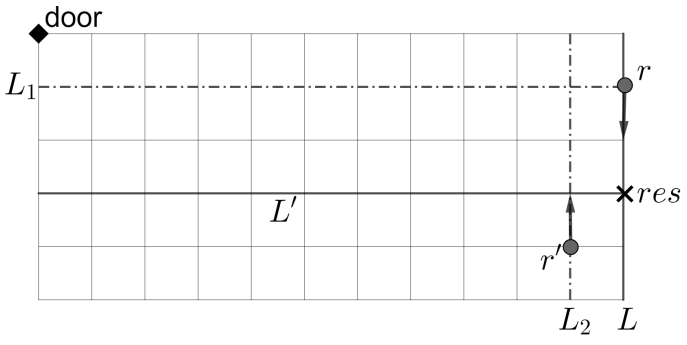


Figure 3.26: Dimension of decreases from $m_1 \times 2$ to $(m_1 - 1) \times 2$ where $m_1 > 2$.

Now we have proved that if no robot terminates then, within $O(T_f \times (m + n))$

rounds there is a round t_1 such that at the beginning of it, R_{Con} is a 2×2 rectangle on the bottom right corner of the grid G (Fig. 3.27). At the beginning of round t_1, res must be at the corner of the Grid diagonally opposite to the door vertex (by Lemma 3.6). Here two robots r and r' must be on two different adjacent vertices of res as the configuration is an InitGather Configuration (Lemma 3.5). Hence by the algorithm of Gather Phase r and r' both move to the vertex of res during the round t_1, while res has no other edges to move out as it is on the corner. So, both robot reaches the location of res and terminates. From this discussion, we can conclude the following Theorem.

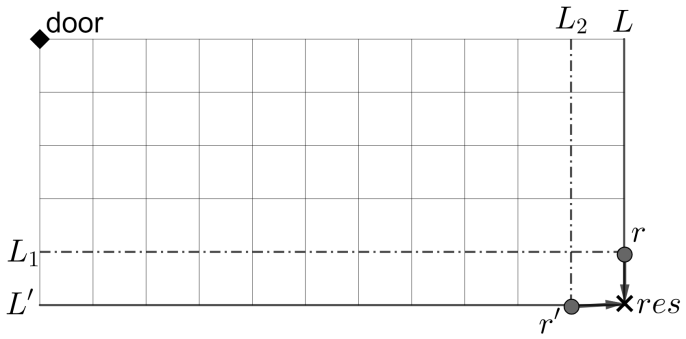


Figure 3.27: R_{Con} has dimension 2×2.

Theorem 3.4. *For a grid of dimension $m \times n$, the* Gather Phase *terminates within $O(T_f \times (m+n))$ rounds.*

Now since the termination of Gather Phase implies termination of the whole algorithm we can conclude with the following theorem.

Theorem 3.5. *Algorithm* Dynamic Rendezvous *terminates within $O(T_f \times (m+n))$ rounds.*

3.4 Conclusion

Gathering is a classical problem in the field of swarm robotics. Rendezvous is a special case of gathering where two robots gather at a single point in the environment. All the previous works on gathering considered the meeting point

to be not known by the robots but here we have considered the robots to know the meeting point but the meeting point can move in the environment until a robot reaches it. To the best of our knowledge, it is the first work that considers a dynamic meeting point. In this work, we have shown that it is impossible for two robots to gather at a known dynamic meeting point on a finite grid if the scheduler is semi-synchronous. Then considering a fully synchronous scheduler we have provided a distributed algorithm Dynamic Rendezvous which gathers the two robots on the known dynamic meeting point called the resource, within $O(T_f \times (m + n))$ rounds where $m \times n$ is the dimension of the grid and T_f is the upper bound of the number of consecutive rounds the resource can stay at a single vertex alone. We have also provided a lower bound of time i.e., $\Omega(m + n)$ to solve this problem considering a $m \times n$ grid. So, if $T_f \leq k$ for some constant k then our algorithm is time optimal.

For future courses of research, one can think of solving this problem on other different networks such as tree, ring, etc. In ring networks, solving this problem with limited visibility can be really interesting. Also, One can think of finding out the minimum number of robots needed to gather at a known dynamic meeting point for different schedulers in different networks.

Chapter 4

Line Formation by Fat Luminous Robots on an Infinite Grid

Line Formation is one of the building blocks for the formation of other complex geometric pattern formation problems by a swarm of robots. In this problem, a swarm of robots have to move in such a way that all robots locate themselves on a single line. This has been earlier studied as a sub-problem to many formation problems such as arbitrary pattern formation on grid-based terrains [2, 60, 69]. In this problem the common approach is to first agree on a global coordinate system, then as an intermediate step, they form a line from which the robots move to their designated target positions according to the agreed global coordinate system. The line formation acts as a signal that the global coordinate they agree on now is fixed and it will remain fixed until the formation is complete.

This problem has been studied earlier in different environments under different assumptions. In the plane, it was studied for oblivious robots in [86]. In grid it was first studied in [13] as a sub-problem to forming arbitrary patterns with point and transparent robots. It was then studied for opaque point robots on a grid in [2] as a sub-problem to the formation of a circle on a grid. In this work, the authors assumed the robots to be equipped with a persistent light having 7 distinct colors and having one axis agreement.

In this chapter, we study the Line Formation problem as a foundational step for forming a circle on a grid (cf. Chapter 5). The algorithm presented in Chapter 5

creates a circle starting from a specific configuration called "P1FC" (see Definition 4.2). Our line formation algorithm is derived as a byproduct of forming a P1FC. The algorithm outlined in this chapter (Algorithm 6) forms a P1FC but also achieves a line formation as an intermediate configuration. This algorithm can easily be adapted to a line formation algorithm with slight modifications for proper termination once the line is formed. These modifications are discussed in Section 4.3. To facilitate a better understanding of the next chapter, we consider the formation of P1FC in this chapter.

In Section 4.1 of this chapter we first describe the model of the robot and scheduler considered for solution of this problem. Furthermore, some definitions are provided as preliminaries before formally defining the problem of P1FC Formation. Then in Section 4.2, an algorithm, FormP1FC is presented that can form a P1FC from any arbitrary configuration. In Section 4.2.1, the correctness of algorithm FormP1FC is analysed . In Section 4.3 it is shown how with a slight modification to FormP1FC, it can be used as a line formation algorithm with only 4 colors. Finally, we conclude this chapter in Section 4.4

4.1 Robot Capabilities, Scheduler and Grid Environment

In this section, we first describe the models considered. Then, some definitions and notations are provided that have been used throughout the entire paper.

4.1.1 Model

Infinite Grid: An infinite grid is an infinite geometric graph $\mathcal{G} = (V, E)$ where vertices are points placed on $\mathbb{R}^2$ with coordinates $\{(a, b) : a \in \mathbb{Z} \text{ and } b \in \mathbb{Z}\}$. Also, two vertices are adjacent iff the Euclidean distance between them is one unit.

Robot Model: This work considers a set $\{r_1, r_2, \ldots r_k\}$ of k robots that are initially placed arbitrarily on k vertices of an infinite grid $\mathcal{G}$. The robots are

considered to be autonomous, anonymous, homogeneous, and identical. The robots are not point. In this work, the robots are considered to be disks of radius at most $\frac{1}{2}$ unit.

Axis and Unit Length Agreement: Each robot has a local coordinate system where the origin is its position of itself. There is no global agreement on the coordinates however the robots agree on the direction and orientation of the x-axis which is parallel to one of the grid lines and also on unit length. That implies the robots agree on left, right, and also on the distance between any two points, but do not agree on up and down.

Visibility Model: In this work, the robots have unlimited but obstructed visibility. A robot r_i can see another robot r_j if and only if there is a point on the perimeter of r_i, say p_i, and another point on the boundary of r_j, say p_j such that the line segment $\overline{p_i p_j}$ does not intersect at a point on any other robot. From this visibility model, it follows that if r_i can see r_j then r_j also sees r_i.

Memory and communication: In this work, the robots are considered to be of the $\mathcal{LUMI}$ model. Thus the robots are able to remember some finite previous states and can communicate with visible robots using finite bit messages broadcasted using lights. Each of the robots is equipped with a light that can have 5 distinct colors from the set $Col = \{$`off`, `chord`, `moving1`, `diameter`, `done`$\}$ one at a time. Upon activation from the idle state, a robot executes according to the Look-Compute-Move cycle (LCM cycle) as described below.

LCM cycle: Upon activation with a color $C_1 \in Col$, a robot r first executes the Look phase. During this phase, the robot takes a snapshot of its surroundings and gets the positions of other visible robots according to its own coordinate system. Using this information as input, the robot then executes the algorithm in the Compute phase. As an output of the algorithm, the robot gets a color $C_2 \in Col$ and a grid point at most one hop away from its current position. During the Move phase, the robot first changes its color to C_2 from C_1 (if different) and then moves to the new grid point (if different). After the Move phase is executed, the robot again returns to the idle state until it is activated again. A robot can move only along the edges of the grid, i.e., a robot can only move to one of the

four adjacent vertices of its current position by moving once. The move is also considered to be rigid and instantaneous, i.e., a robot is always seen on the grid points.

Scheduler Model: The scheduler considered in this work is the most general asynchronous scheduler ($\mathcal{ASYNC}$). There is no agreement on rounds. The time taken by a robot to execute the Look phase, Compute phase, Move phase, and the time a robot remains idle is finite but unpredictable.

4.1.2 Notations and Definitions

We have used some notations throughout the paper. A list of these notations is mentioned in the following table.

Notation	Description
$\mathcal{L}_1$	First vertical line on left that contains at least one robot.
$\mathcal{L}_i$	i-th vertical line on the right starting from $\mathcal{L}_1$.
$\mathcal{L}_V(r)$	The vertical line on which the robot r is located.
$\mathcal{L}_H(r)$	The horizontal line on which the robot r is located.
$\mathcal{L}_I(r)$	The left immediate vertical line of robot r which has at least one robot on it.
$\mathcal{R}_I(r)$	The right immediate vertical line of robot r which has at least one robot on it.
$H_L^O(r)$	Left open half for the robot r.
$H_L^C(r)$	Left closed half for the robot r (i.e $H_L^O(r) \cup \mathcal{L}_V(r)$).
$H_B^O(r)$	Bottom open half for the robot r.
$H_B^C(r)$	Bottom closed half for the robot r (i.e $H_B^O(r) \cup \mathcal{L}_H(r)$).
$H_U^O(r)$	Upper open half for the robot r.
$H_U^C(r)$	Upper closed half for the robot r (i.e $H_U^O(r) \cup \mathcal{L}_H(r)$).

Definition 4.1 (***Configuration***)**.** *Let $\mathcal{G} = (V, E)$ be an infinite grid. Let $f : V \to \{0, 1\} \times (Col \cup \{NULL\})$ be a function such that*

$$f(a,b) = \begin{cases} (0, NULL) & \text{if there is no robot on (a,b)} \\ (1, c) & \text{if a robot is on } (a,b) \text{ with color } c \in Col \end{cases}$$

Then we call the pair $(\mathcal{G}, f)$ a configuration which is denoted as $\mathcal{C}$. A configuration at time t is denoted as $\mathcal{C}(t)$.

Definition 4.2 (**P1FC**). *We call a configuration a P1FC if the following conditions hold*

1. $\mathcal{L}_2$ *has exactly two robots, say* r_1 *and* r_2*, with color* `diameter`*. All other robots are on either* $\mathcal{L}_1$ *or* $\mathcal{L}_3$ *with color* `chord`*.*

2. *All robots on* $\mathcal{L}_1$ *and* $\mathcal{L}_3$ *are strictly between* $\mathcal{L}_H(r_1)$ *and* $\mathcal{L}_H(r_2)$*.*

Definition 4.3. *Let* t_b *be the time when a robot moves first from the initial configuration. We say that* $\mathcal{L}_1$ *is fixed at a time* $t_f > t_b$ *if both of the following conditions hold:*

1. $\forall t \in [t_b, t_f)$, $\mathcal{C}(t)$ *has a robot that is not on* $\mathcal{L}_1$*.*

2. *From time* t_f *onwards, no robot from* $\mathcal{L}_1$ *moves left until all robots move to* $\mathcal{L}_1$*.*

Definition 4.4 (***Terminal Robot***). *In a configuration* $\mathcal{C}$*, a robot* r *is called a terminal robot on* $\mathcal{L}_V(r)$ *if there is no robot either above or below* r *on* $\mathcal{L}_V(r)$*.*

Definition 4.5 (P1FC Formation). *Let a set of* k *fat opaque robots of the same radius be placed arbitrarily on the vertices of an infinite grid* $\mathcal{G}$*.* P1FC Formation *problem is said to be solved if* $\mathcal{C}(t)$ *satisfies the following condition.*

$\exists t > 0$ *such that* $\mathcal{C}(t)$ *is a P1FC and* $\mathcal{C}(t') = \mathcal{C}(t), \forall t' \geq t$*. .*

4.2 P1FC Formation

In this section, we introduce an algorithm called FormP1FC, which solves the P1FC formation problem in finite time. In the next chapter, we will examine the Circle Formation problem and provide an algorithm that forms a circle, given the initial configuration is a P1FC. Thus, we can use the FormP1FC algorithm introduced in this chapter as phase 1 for the circle formation algorithm in Chapter 5, to form a circle on a grid from any arbitrary asymmetric configuration. Within this context, a robot is considered to be in phase 1 if it has not yet formed a P1FC.

Algorithm 6 can be easily modified to serve as a line formation algorithm. (Described in Section 4.3)

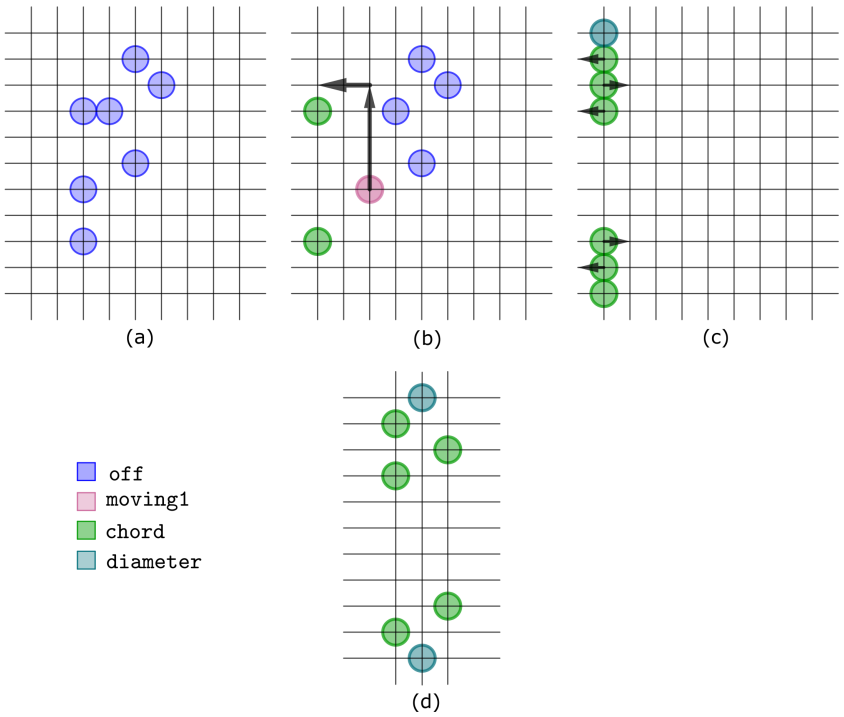


Figure 4.1: Outline of FormP1FC: (a) Initial configuration $\mathcal{C}(0)$. **(b)** Two terminal robots of $\mathcal{L}_1$ in $\mathcal{C}(0)$ moved two hop and changes color to `chord`. A terminal robot of color `off` on the nearest vertical line of $\mathcal{L}_1$ changes color to `moving1` and moves to $\mathcal{L}_1$. **(c)** All robots of color `off` moves to $\mathcal{L}_1$ in a similar way and forms a single line. The terminal robots on this line change color to `diameter` eventually. (here in this specific example, due to asynchrony only one terminal robot changes color to `diameter` first). Next, all non-terminal robots move either left or right. (**d**) P1FC is formed.

4.2.0.1 Brief Description of the Algorithm

Before going into details of the algorithm, let us first have the pseudocode of the algorithm FormP1FC.

Algorithm 6: FORMP1FC

```
1 r ← myself
2 if r.color = off then
3 |   Execute OFFP1FC(r)
4 else if r.color = moving1 then
5 |   Execute MOVING1P1FC(r)
6 else if r.color = chord then
7 |   Execute CHORDP1FC(r)
```

If a robot r has color `off` it executes OFFP1FC(r). Otherwise if the robot r has color `moving1` then it executes MOVING1P1FC(r). Also, a robot r executes CHORDP1FC(r) if the color of the robot is `chord`.

Algorithm 7: OFFP1FC(r)

```
1 if r sees no robot with color diameter then
2 |   if there is no robot on H_L^O(r) and r is terminal then
3 |   |   r.color ← moving1;
4 |   |   move left;
5 |   else if L_I(r) is at least two hop away and all visible robots on L_I(r) has color
  |     chord then
6 |   |   if r is terminal then
7 |   |   |   r.color ← moving1;
```

Initially, all the robots have color `off` and are placed arbitrarily on the grid. There can be at most two robots that are terminal on $\mathcal{L}_1$ of the initial configuration. Upon activation, these robots will see their left open half is empty and one of their upper or bottom halves of their corresponding vertical line is empty. For this view, the robots change their color to `moving1` from `off` and moves left shifting the line $\mathcal{L}_1$ (See line 2-4 of Algorithm 7). Except the terminal robots on $\mathcal{L}_1$ of the initial configuration other robots of color `off` change their color to `moving1` only when they are terminal on their corresponding vertical line and see all robots on their left immediate vertical line has color `chord` and $\mathcal{L}_I(r)$ is at least two hop away from $\mathcal{L}_v(r)$ (See line 5-7 of Algorithm 7). So, unless the terminal robots on $\mathcal{L}_1$ of the initial configuration change their color to `chord` no other robots do anything even if they are activated.

Algorithm 8: MOVING1P1FC(r)

```
1  if r sees no robot with color diameter on L_V(r) then
2      if all visible robots on L_I(r) has color chord and r sees no robot of color chord
         on L_V(r) then
3          if H_U^C(r) ∩ L_I(r) and H_B^C(r) ∩ L_I(r) both are non empty then
4              if there is a robot r' on L_V(r) then
5                  move opposite to r'
6              else
7                  move according to positive Y− axis;
8          else
9              move left;
10     else if (r is singleton on L_V(r) and all visible robots on L_I(r) has color
         moving1) or, (distance of R_I(r) having a robot with color off = 1 and H_L^O(r) is
         empty. then
11         move left;
12     else if H_L^O(r) is empty then
13         if sees a robot with color chord on L_V(r) then
14             r.color ← chord;
15         else if distance of R_I(r) having a robot of color off ≥ 2 then
16             r.color ← chord;
17 else
18     r.color ← diameter;
19     terminate;
```

Let r_1 and r_2 be two terminal robots on $\mathcal{L}_1$ in the initial configuration. If $\mathcal{L}_1$ of the initial configuration contains more than two robots then there must be another robot except r_1 and r_2, say r, with color `off`. Now upon activation let r_1 have moved left once with color `moving1`. Now when it is activated again, it will see r on its $\mathcal{R}_I(r_1)$ having color `off` and also $\mathcal{R}_I(r_1)$ is one hop away from $\mathcal{L}_V(r_1)$. In this view if $H_L^O(r_1)$ is empty then, r_1 moves left again shifting the $\mathcal{L}_1$ further left (See line 10-11 of Algorithm 8). The target of this move is to make the distance between $\mathcal{L}_1$ and the first vertical line that contains a robot of color `off` more than one. Let when r_1 moved left further r_2 was on $\mathcal{L}_V(r)$ with a pending movement due to asynchrony. Then, it will move left and see all robots on $\mathcal{L}_I(r_2)$ has color `moving1` and it is singleton on $\mathcal{L}_V(r_2)$. For this view r_2 moves to left again and moves to $\mathcal{L}_1$ along with r_1 (See Line 10-11 of Algorithm 8). A robot with color `moving1` can change its color to `chord` for two possible views. For the first one, it has to be on $\mathcal{L}_1$ and has to see another robot with color

`chord` on $\mathcal{L}_1$ (Line 13-14 of Algorithm 8). For the other one, it has to see its right immediate vertical line, which is at least two hops away and has at least a robot of color `off` (Line 15-16 of Algorithm 8). So when r_1 moved left twice from its initial position upon its next activation activation, if r_2 is still in $\mathcal{L}_v(r)$ and has not yet changed its color to `moving1`, r_1 changes its color to `chord` otherwise r_2 reaches $\mathcal{L}_V(r_1)$ and eventually both of them change their color to `chord` before any other robot does anything. Now a robot, say r', which has changed its color to `moving1` from `off` on $\mathcal{L}_k$ by seeing all robots of color `chord` on $\mathcal{L}_I(r')$ must be terminal on $\mathcal{L}_k$ for some $k > 1$. Now, if upon activation r' sees all visible robots on $\mathcal{L}_I(r')$ has color `chord` and no robot of color `chord` on $\mathcal{L}_V(r')$ (This condition is to stop robots of color `moving1` to move further left from $\mathcal{L}_1$), then it can have either both of $H^C_U(r') \cap \mathcal{L}_I(r')$ and $H^C_B(r') \cap \mathcal{L}_I(r)$ non-empty or, empty. For this case, if there is another robot on $\mathcal{L}_V(r')$, then r' moves along $\mathcal{L}_V(r')$ opposite to that robot, otherwise it moves along its positive Y-axis. After finite moves one of $H^C_U(r') \cap \mathcal{L}_I(r')$ and $H^C_B(r') \cap \mathcal{L}_I(r)$ must become empty for r' when it moves left (See Algorithm 8, line 2-9).

If a robot of color `moving1` sees a robot with color `diameter` on its own vertical line, it changes its color to `diameter` (Line 17-19 of Algorithm 8).

Algorithm 9: ChordP1FC(r)

```
1  if r sees no robot with color diameter then
2      if There is a robot with color chord on L_V(r), there is no robot on H^O_L(r), H^O_R(r)
         and H(r) where H(r) ∈ {H^C_B(r), H^C_U(r)} then
3          r.color ← diameter;
4          terminate;
5  else if r sees a robot with color diameter on L_V(r) then
6      if r is terminal then
7          r.color ← diameter;
8          terminate;
9      else
10         Execute ChordMove();
```

Now a robot with color `chord` changes its color to `diameter` when it sees at least one robot of color `chord` on its corresponding vertical line and sees no other robot on its left and right open halves and one of the upper or bottom closed halves (See line 1-4 of Algorithm 9). A robot, say r_c, with color `chord` executes

ChordMove when it is not terminal on $\mathcal{L}_V(r_c)$ and sees at least one robot of color `diameter` on $\mathcal{L}_V(r_c)$ (Line 9-10 of Algorithm 9). On the other hand if r_c is terminal on $\mathcal{L}_V(r_c)$ and sees a robot of color `diameter` on $\mathcal{L}_(r_c)$ then it changes its color to `diameter` from `chord` (Line 5-8 of Algorithm 9).

We now describe the ChordMove subroutine.

ChordMove **Subroutine:** A robot r with color `chord` executes the subroutine ChordMove when it sees at least one robot with color `diameter` and is not terminal on $\mathcal{L}_V(r)$. while executing ChordMove, if a robot, say r, sees only one robot r_1 with color `diameter` then it checks if there is any other robot between the horizontal lines passing through r_1 and r i.e., $\mathcal{L}_H(r_1)$ and $\mathcal{L}_H(r)$ respectively. If there are no robots in the above-mentioned region then r moves left. Now if there are robots between the mentioned region, then the following procedure takes place. If r_N is the nearest of r which is in the mentioned region and if r_N is on $H_L^O(r)$ then r moves right otherwise r moves left.

Now, if a robot r sees both the robots, say r_1 and r_2, with color `diameter` then, r finds its direction to move as stated above considering both the regions between $\mathcal{L}_H(r_1)$, $\mathcal{L}_H(r)$ and $\mathcal{L}_H(r_2)$, $\mathcal{L}_H(r)$. If for both the direction considering both the regions are same then r moves according to that direction otherwise it moves left. Note that after this procedure is complete, $\mathcal{L}_V(r_1)$ has only two robots r_1 and r_2 both having the color `diameter` and $\mathcal{L}_I(r_1)$ and $\mathcal{R}_I(r_1)$ contains all the robots with color `chord`. Also observe that, difference between number of robots on $\mathcal{L}_I(r_1)$ and $\mathcal{R}_I(r_1) \leq 2$. Note that, a robot, say r, with color `chord` does nothing after it has already moved once executing ChordMove until the whole procedure is complete. This is because after r has moved once, it sees r_1 or r_2 not on $\mathcal{L}_V(r)$ and $\mathcal{L}_V(r_1)$ has other robots except r_1 and r_2 until all non-terminal robots on $\mathcal{L}_V(r_1)$ executes this subroutine exactly once.

4.2.1 Correctness Analysis

The correctness of Algorithm 6 is divided into two parts. In the first part, the robots first form a line where all robots have either the color `chord` or `moving1`.

To be specific only terminal robots can have the color `moving1` on that line. Now at least one of the terminal robots eventually changes its color to `diameter`. Then in the second part, the non-terminal robots with color `chord` move left or right once, reaching either $\mathcal{L}_1$ or $\mathcal{L}_3$ and thus forming a P1FC eventually.

So, first, we have to show that all robots must move to a single line eventually where all robots have color either `chord` or `moving1`. We prove this by ensuring that all robots with color `off` eventually change their color to `moving1` (Lemma 4.3). Then all robot with color `moving1` moves to $\mathcal{L}_1$ (Lemma 4.2). For this, we also ensured that $\mathcal{L}_1$ becomes fixed after a finite time (Lemma 4.1), otherwise, a potential livelock situation may occur.

In the following, we have stated some observations that proved some claims that will be needed to prove the above-mentioned lemmas.

Observation 4.1. *A robot r can see all robots of $\mathcal{L}_I(r)$ (resp. $\mathcal{R}_I(r)$) if $\mathcal{L}_I(r)$ (resp. $\mathcal{R}_I(r)$) is at least two hop away from r.*

Observation 4.2. *If r be a robot of color `off` executing* FormP1FC *such that $H^O_R(r)$ is non empty, then all robots on $H^O_R(r)$ must have color `off`*

Observation 4.3. *After a move by any robot from the initial configuration and before any robot changes its color to `chord`, $\mathcal{L}_1$ can have at most two robots of color `moving1` and all other robots has color `off`.*

Claim 1. *Let r be a robot with color `moving1` such that $\mathcal{R}_I(r)$ is one hop away from $\mathcal{L}_V(r)$ and there is at least one robot with color `off` on $\mathcal{R}_I(r)$. If r is activated in this configuration, then r always sees a robot having color `off` on $\mathcal{R}_I(r)$ during the execution of* FormP1FC.

Proof. Let r be a robot with color `moving1`. Let $\mathcal{R}_I(r)$ be one hop away from $\mathcal{L}_V(r)$ which has a robot, say r' of color `off` on it. If possible r does not see any robot with color `off` on $\mathcal{R}_I(r)$. That is r does not see r' upon activation, say at a time $t > 0$. Let $\mathcal{L}_H(r)$ and $\mathcal{L}_H(r')$ be two horizontal lines passing through r and r' respectively. Then the above assumption is true only when $\mathcal{L}_V(r)$ and $\mathcal{L}_V(r')$ each contains at least one robot between $\mathcal{L}_H(r)$ and $\mathcal{L}_H(r')$. Let r_1 and r_2

be two such robots on $\mathcal{L}_V(r)$ and $\mathcal{L}_V(r')$ respectively. without loss of generality, let r can see r_2 and r' can see r_1. Note that r_2 must be of color `moving1` at time t. Also r_1 either have color `off` or color `moving1` at time t (r_1 can not be of color `chord` at time t as r' is on $\mathcal{R}_I(r_1)$ having color `off` at time t).

Now, if r_1 has color `off` then in the interval $(0, t)$, $\mathcal{L}_V(r_1)$ has not changed. Now in this interval, all robots on $\mathcal{L}_V(r_1)$ must have color `moving1` or `off`. So in the interval, $(0, t)$, r_2 can never see all robots with color `chord` on $\mathcal{L}_I(r_2)$ and thus can not change its initial color `off` to `moving1` in the mentioned interval. So, r_2 can not be of color `moving1` at time t which is a contradiction to our assumption.

So, let r_1 has color `moving1` at time t. Now, since at time t, r_2 has color `moving1`, there is a time $t_1 < t$ when r_2 has color `off`, is terminal on $\mathcal{L}_V(r_2)$ and either sees $H_L^O(r_2)$ is empty or sees all robots on $\mathcal{L}_I(r_2)$ having color `chord`. This implies r and r_1 must have moved to $\mathcal{L}_I(r_2)$ after time t_1. Thus, at time t_1 all of r, r_1 and r_2 were along with r' on $\mathcal{L}_V(r_2) = \mathcal{L}_V(r')$. Now since, r and r_1 moves to $\mathcal{L}_I(r_2)$, they must have changed their color to `moving1` from initial color `off`. There are three cases. Firstly, let both r and r_1 get activated and see that they are terminal on $\mathcal{L}_V(r')$ and change their color to `moving1` on or after time t_1. This is not possible as in this case r, r_1 and r_2 all have to be terminal on $\mathcal{L}_V(r')$ at time t_1 which is not possible. Secondly, let both r and r_1 be activated and see themselves terminal on $\mathcal{L}_V(r')$ and change their color to `moving1` before t_1. This case is also impossible as at time t_1 since r and r_1 are still on $\mathcal{L}_V(r')$, r_2 can not see itself as a terminal robot which is a contradiction. So first, let us assume r changed its color before time t_1 and r_1 changed its color after time t_1 and before time t. We claim that, for r_1 to change its color after time t_1, r must have to move left from $\mathcal{L}_V(r')$. This is because, at time t_1, r and r_2 both are terminal on $\mathcal{L}_V(r')$ so until r moves r_1 can not become terminal and thus can not change its color. Now after time t_1 if r_1 is activated before time t, even if it is terminal now it will not change its color to `moving1` as $\mathcal{L}_I(r_1)$ is exactly one hop away (less than two hop away). Thus, even if r_1 is activated it does not change its color from `off` until time t. So at time t, r can see r_2 with color `off` contrary to our assumption. Now, if r_1 had changed its color to `moving1` before time t_1 and r

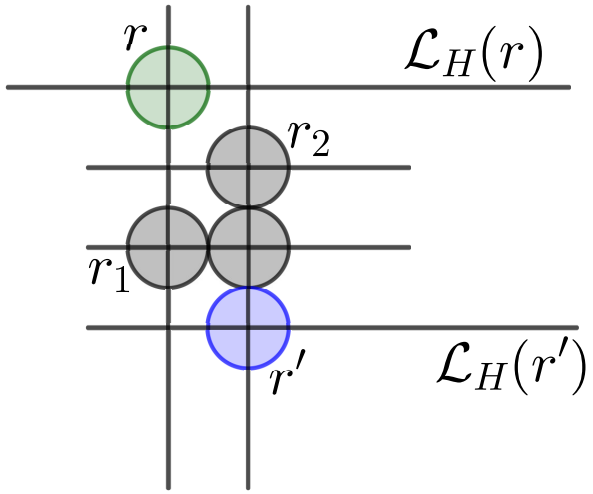


Figure 4.2: r has color `moving1`, r' has color `off` at time t and r can not see r'.

after time t_1, then by similar argument it can be said that at time t, r stays at $\mathcal{L}_V(r')$. So, in this case, if $\mathcal{R}_I(r)$ is non-empty, all robots on $\mathcal{R}_I(r)$ will have color `off` (Observation 4.2). Thus, r must see at least one robot having color `off` on $\mathcal{R}_I(r)$ even if it is one hop away contrary to our assumption. Thus, for a robot r if $\mathcal{R}_I(r)$ has a robot with color `off` and $\mathcal{R}_I(r)$ is one hop away then r always sees at least one robot of color `off` on $\mathcal{R}_I(r)$.

□

Claim 2. *Before $\mathcal{L}_1$ is fixed, no robot in the configuration can have color `chord`.*

Proof. Let there be a time t when $\mathcal{L}_1$ is not fixed but the configuration at time t has a robot, say r with color `chord`. without loss of generality let r be the first robot that changes its color from `moving1` to `chord`. So, there must be a time $t_0 < t$ when r is activated on $\mathcal{L}_1$ with color `moving1` and changes its color to `chord`. Note that at time t_0, $\mathcal{L}_1$ is not fixed. Also, since there are more than 2 robots in the system, there must be at least one robot with color `off` which is not on $\mathcal{L}_1$ for the whole duration $[t_b, t_0)$. Thus there must be a robot on $\mathcal{L}_1$ with color `moving1` which moves left after t_0. Let r' be that robot. Note that r' can not be r as a robot with color `chord` does not move until $\mathcal{L}_1$ is fixed. Now, if r' is

activated after time t_0 for executing the LCM cycle where it moves left from $\mathcal{L}_1$ then, upon activation, it must have seen a robot r_1 with color `off` on $\mathcal{R}_I(r') = \mathcal{L}_2$ which is one hop away from $\mathcal{L}_1$. But this is not possible because if r_1 is on $\mathcal{L}_2$ after time t_0 it must have been there at time t_0 also. So at time t_0, r does not change its color to `chord` upon activation contrary to our assumption. So, let r' be activated at a time $t' < t_0$ for executing the LCM cycle where it moves left from $\mathcal{L}_1$ after time t_0. This is only possible if at time t', $\mathcal{R}_I(r') = \mathcal{L}_2$ had a robot, say r_1, with color `off`. Now since r' moves left after time t_0, at time t_0 upon activation r must have seen a robot with color `off` on $\mathcal{R}_I(r) = \mathcal{L}_2$ which is one hop away from $\mathcal{L}_2$. Thus r doesn't change its color to `chord` upon activation at time t_0. This is also a contradiction. Thus, before $\mathcal{L}_1$ is fixed, no robot changes its color to `chord`. □

Claim 3. *During the execution of Algorithm 6, between the time of first move by any robot from the initial configuration and the time when all robots move to a single line for the first time, a robot on $\mathcal{L}_1$ have color either `moving1` or `chord`.*

Proof. Let $t_b > 0$ be the time when the first move by a robot happened from the initial configuration. Also, let $t_f > t_b$ be the time when all robots move to a single line after time t_b for the first time (t_f can be infinite if all robots never move to a single line). Now, $\mathcal{L}_1$ can not have a robot with color `diameter` in the time interval (t_b, t_f) as, in this interval, no robot sees both its left and right open halves empty. Also, since at time t_b at least one leftmost terminal robot moves left after changing its color to `moving1`, $\mathcal{L}_1$ also shifts left at time t_b. Note that, after this move, $\mathcal{L}_1$ does not have any robot with color `off`. So in the interval (t_b, t_f), $\mathcal{L}_1$ can not have any robot with color `off` as robots with color `off` never moves left to reach $\mathcal{L}_1$ (algorithm 6) and no robot of different color change their color to `off` while executing Algorithm 6. So, within the time interval (t_b, t_f), $\mathcal{L}_1$ can have robots of color either `moving1` or of color `chord`. □

Lemma 4.1. *$\mathcal{L}_1$ can not shift left infinitely often without all robots being on $\mathcal{L}_1$.*

Proof. Let t_b be the time when the first robot moves from the initial configuration. After t_b, suppose $\mathcal{L}_1$ shifts left infinitely often, while there remains at least one

robot not positioned on $\mathcal{L}_1$ after t_b. This implies there is a robot, say r, which moves left from $\mathcal{L}_1$ infinitely often. Notably, robot r must have the color `moving1`, and it retains this color without change. Now r can move left from $\mathcal{L}_1$ only if it sees a robot, say r', of color `off` on $\mathcal{R}_I(r)$ which is one hop away from $\mathcal{L}_1$ (Figure 4.3). Let $t_0 > t_b$ be a time when r is activated on $\mathcal{L}_1$ and observes r' on $\mathcal{R}_I(r)$, situated at a distance of one unit from $\mathcal{L}_1$. In this case, r moves left and shift $\mathcal{L}_1$ to left along with it. Note that after this move is completed, no robot on $\mathcal{L}_1$ will ever see another robot of color `off` on $\mathcal{L}_2$ (as robots with color `off` never move left with same color according to Algorithm6). Consequently, for all subsequent times $t_1 > t_0$, if r is reactivated at t_1 on $\mathcal{L}_1$ with the color `moving1`, it will not perceive any robot with the color `off` on $\mathcal{L}_2$. As a result, it will not move left, contradicting our initial assumption. Hence, the leftward shift of $\mathcal{L}_1$ cannot continue infinitely without all robots being positioned on the same line. □

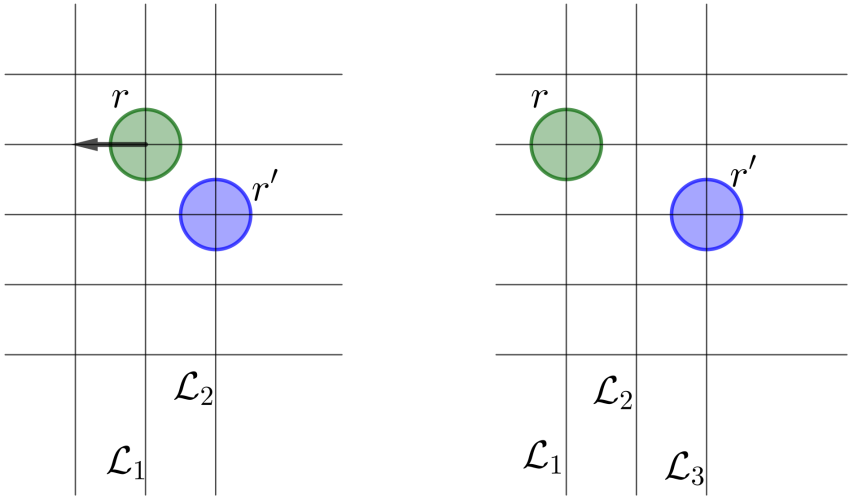


Figure 4.3: in the configuration on left, r can see r' with color `off` on $\mathcal{L}_2$ so it moves left. in the configuration on right, r can not see any robot on $\mathcal{L}_2$ with color `off` so it does not move

Claim 4. *Excluding $\mathcal{L}_1$ there can not be more than two robots with color `moving1` in during the execution of Algorithm 6.*

Proof. Suppose there are three robots, denoted as r_1, r_2, and r_3, all with color

`moving1`, positioned on $\mathcal{L}k_1, \mathcal{L}k_2$, and $\mathcal{L}k_3$ at time t, where $1 < k_1 \leq k_2 \leq k_3$. Initially, r_1, r_2, and r_3 must all be located on the same vertical line. Otherwise, the rightmost robot among them in the initial configuration cannot transition its color from `off` to `moving1`, unless the leftmost robot(s) among the others reach $\mathcal{L}_1$ and alter their color to `chord`. However, this contradicts the observation that all three robots have the color `moving1` at time t. Therefore, let's assume that initially all of them are positioned on $\mathcal{L}_k$ with the color `off`. It is important to note that none of them moves left from $\mathcal{L}_k$ unless all of them are activated for their corresponding LCM cycle, in which they change their color to `moving1`. This implies that there exists a time $t' < t$ when all of r_1, r_2, and r_3 are located on $\mathcal{L}_k$, and each of them either possesses the color `moving1` or becomes activated in their corresponding LCM cycle, during which they change their color to `moving1`. This implies r_1, r_2 and r_3 all are terminal on $\mathcal{L}_k$ at time t' but this is not possible. Hence, excluding $\mathcal{L}_1$, there can not be more than two robots with color `moving1` during the execution of Algorithm 6. □

Claim 5. *A robot with color `moving1` on $\mathcal{L}_1$ must be terminal on $\mathcal{L}_1$ while executing Algorithm 6.*

Proof. From Observation 4.3 and Claim 2 it is evident that before $\mathcal{L}_1$ is fixed, it can have at most two robots and those are of color `moving1`. So before $\mathcal{L}_1$ is fixed, any robot with color `moving1` on $\mathcal{L}_1$ must be terminal on $\mathcal{L}_1$.

Let $\mathcal{L}_1$ become fixed at a time t_0. Now let us assume r be a robot with color `moving1` on $\mathcal{L}_1$ that is not terminal on $\mathcal{L}_1$ at a time $t > t_0$. This implies there must be another robot, say r', which without loss of generality is directly below r on $\mathcal{L}_1$ at the time t and r has color `moving1`. Now, r and r' must have moved to $\mathcal{L}_1$ from $\mathcal{L}_2$. Note that, in the initial configuration, r and r' can not be on two different vertical lines otherwise the rightmost robot among them can not change its color from `off` to `moving1` until the other one reaches $\mathcal{L}_1$ and change color to `chord`. Also, even when they are on the same vertical line and one of them, say r, already moves left before the other one i.e., r' wakes to change its color from `off` to `moving1`, it can't do so unless r reaches $\mathcal{L}_1$ and change its color to `chord`. So, without loss of generality let us assume before one of r and r' moves from $\mathcal{L}_k$,

the other robot must have been activated and seen $\mathcal{L}_1$ where all robots have color `chord`. This ensures that r and r' will change their color to `moving1` from `off`. Let in the initial configuration, r and r' were on the same line which is the line $\mathcal{L}_k$ at the time t_0 and $k \geq 3$ ($\mathcal{L}_2$ at time t_0 can not have any robot in the initial configuration). Also observe that, for r and r' to reach $\mathcal{L}_1$ they must move there from $\mathcal{L}_2$ (The lines $\mathcal{L}_i$ are denoted for the time when $\mathcal{L}_1$ becomes fixed). Since in the initial configuration, r and r' were not on $\mathcal{L}_2$ of the current configuration, they must have color `moving1` while on $\mathcal{L}_2$. Suppose r is located on $\mathcal{L}_2$ while r' is on $\mathcal{L}_j$ with $j \geq 2$ and has color either `moving1` or is in the transitional LCM cycle where it would change its color to `moving1` on $\mathcal{L}_j$. Now for the former case, if r is activated and it moves to $\mathcal{L}_1$ while r' is idle then upon activation r' can not move to $\mathcal{L}_{j-1}$ until r changes its color to `chord`. For the latter case also, before the next activation of r', if r moves to $\mathcal{L}_1$, r' can not move to $\mathcal{L}_{j-1}$ unless r changes its color to `chord`. In either of these cases, when r' reaches $\mathcal{L}_1$, r must be of color `chord`, which contradicts our assumption.

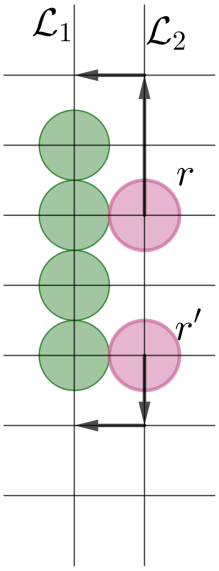


Figure 4.4: r and r' from $\mathcal{L}_2$ moves to $\mathcal{L}_1$ in such a way that at $\mathcal{L}_1$ all other robots are between r and r'. Thus r' can not be directly below or above r.

Therefore, for both r and r' to reach $\mathcal{L}_1$ with the color `moving1`, there must exist a time $t_1 > t_0$ and $t_1 < t$ when both of them are on $\mathcal{L}_2$ with the color `moving1`.

Furthermore, neither of them moves to $\mathcal{L}_1$ while the other one is inactive. Note that r' must have changed its color to `moving1` from `off` after seeing all robots with color `chord` as $\mathcal{L}_1$ of the initial configuration can not have more than two terminal robots of color `off`. This implies $\mathcal{L}_1$ must have at least one robot of color `chord` at the time when r' moves on it after time t_0. Now since both r and r' are activated on $\mathcal{L}_2$ before any one of them moves left, they must see each other on $\mathcal{L}_2$ and move opposite of each other on $\mathcal{L}_2$ until one robot has moved above $\mathcal{L}_H(r_1)$ and the other moves below $\mathcal{L}_H(r_2)$, where r_1 is the uppermost robot on $\mathcal{L}_1$ and r_2 is the lowest robot on $\mathcal{L}_1$ before r and r' moves on to $\mathcal{L}_1$. Thus now when they move left on $\mathcal{L}_1$ they must have at least one robot with color `chord` between them. So at time t, r' can not be directly below r. Thus A robot with color `moving1` on $\mathcal{L}_1$ must be terminal on $\mathcal{L}_1$. □

Claim 6. *During execution of Algorithm 6, if the fixed $\mathcal{L}_1$ has a robot r with color `moving1` which sees at least one robot that is not on $\mathcal{L}_1$ upon activation then r changes its color to `chord` eventually.*

Proof. If $\mathcal{L}_1$ has more than two robots then r must see a robot with color `chord` on $\mathcal{L}_V(r) = \mathcal{L}_1$ upon activation (Claim 5 and Claim 3) (Figure. 4.5(a)). So, it must change its color to `chord`. Therefore, let $\mathcal{L}_1$ have at most two robots when r is activated on the fixed $\mathcal{L}_1$ at a time, say t_0. If another robot has color `chord` on $\mathcal{L}_1$ when r is activated at t_0, by the same reason r changes its color to `chord`. So let us assume that if there is another robot, say r', on $\mathcal{L}_1$ at time t_0 then it has color `moving1` (Figure 4.5(b)). In this configuration, there is no robot of color `chord`. So all other robots except r and r' are of color `off` in this configuration, thus they are at their initial positions. For this case, upon activation, r must see at least one robot with color `off` on $\mathcal{R}_I(r)$ which is at least two hop away from $\mathcal{L}_1$ and thus r changes its color to `chord`. Now at time t_0 upon activation if r is singleton on $\mathcal{L}_1$ then there can be at most another robot with color `moving1` on $\mathcal{L}_j$ ($j > 1$) (Figure. 4.5(c)). Now if there is no other robot with color `moving1` at time t_0 then r sees $\mathcal{R}_I(r)$ has a robot of color `off` and it is at least 2 hop away from $\mathcal{L}_1$. Thus, in this case, r changes its color to `chord`. Otherwise, the other robot, say r' with color `moving1` is either singleton on some $\mathcal{L}_j$ or not ($j > 1$).

Also in between $\mathcal{L}_1$ and $\mathcal{L}_j$, there is no other robot at time t_0. This is because, $\mathcal{L}_j$ at time t_0 must be either the vertical line $\mathcal{L}_1$ of the initial configuration or is strictly left of $\mathcal{L}_1$ of the initial configuration and since there is no robot with color `chord` on $\mathcal{L}_1$, except r and r' all robots has color `off` at time t_0 (this ensures all robots except r and r' never moved from their initial position until time t_0). If, at time t_0, r' is singleton on some $\mathcal{L}_j$ ($j > 1$) then upon activation r' moves left to $\mathcal{L}_{j-1}$. Note that if r' is not on $\mathcal{L}_1$, r on $\mathcal{L}_1$ does nothing even when activated. So r' eventually reaches $\mathcal{L}_1$. Now when r' reaches $\mathcal{L}_1$, by the above argument when r activates next, it changes the color to `chord` as it sees $\mathcal{R}_I(r)$ has a robot with color `off` which is at least two hop away from $\mathcal{L}_1$. Now let at time t_0, r' was not singleton on $\mathcal{L}_j$, then there must be another robot, say r_1, with color `off` on $\mathcal{L}_j$ which is seen by r at time t_0. For this case also, r changes its color to `chord`. □

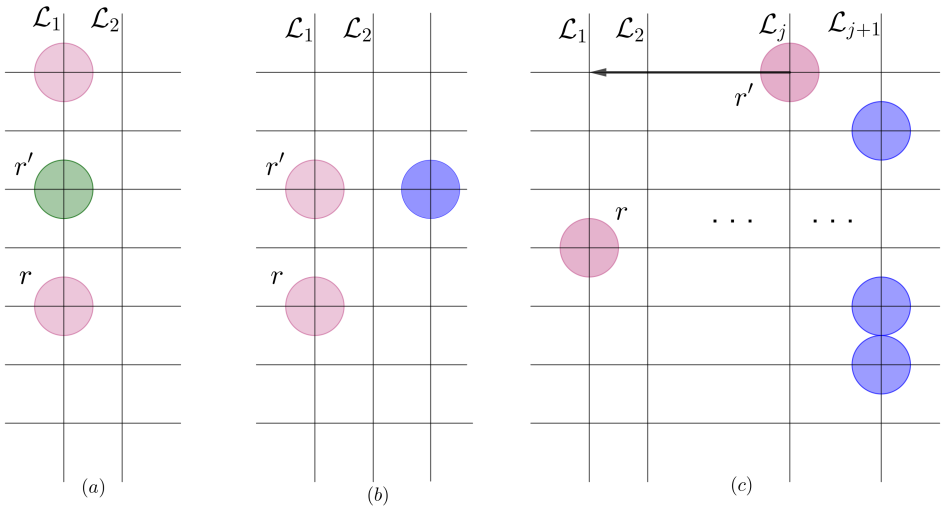


Figure 4.5: **(a)** $\mathcal{L}_1$ has more than two robots including r. Then r sees r' of color `chord` on $\mathcal{L}_1$. **(b)** There are exactly two robots, r and r' on $\mathcal{L}_1$ and both of them has color `moving1`. **(c)** r is singleton on $\mathcal{L}_1$ with color `moving1` and r' is another robot with color `moving1` on $\mathcal{L}_j$. r' moves to $\mathcal{L}_1$ and transforms into case (b). Here the blue color denotes color `off`

Lemma 4.2. *During execution of Algorithm 6, a robot with color* ***moving1*** *on $\mathcal{L}_j$ eventually moves to $\mathcal{L}_{j-1}$ where $j \geq 2$ if $\mathcal{L}_1$ is fixed.*

Proof. We will prove this using mathematical induction on j.

Base case: In the base case we first establish that a robot r with color `moving1` on $\mathcal{L}_2$ moves to $\mathcal{L}_1$ eventually. For that, let r be a robot on $\mathcal{L}_2$ with color `moving1` at a time t. If possible, let r never reaches $\mathcal{L}_1$. Note that there can be at most one robot, say r', other than r which is not on $\mathcal{L}_1$ and has color `moving1` at time t(Claim 4). Now let at time t, r' is on $\mathcal{L}_k$ for some $k \geq 2$.

Case 1: Suppose r' is on $\mathcal{L}_2$ along with r (i.e., $k = 2$) at time t. Note that $\mathcal{L}_2$ can not have any other robot of color `off` as it is strictly to the left of the $\mathcal{L}_1$ in the initial configuration. So all robots on the right of $\mathcal{L}_2$ must be of color `off` and do not do anything even if they are activated as whenever they are activated they never see any robot with color `chord` on their left immediate vertical lines. Now at time t, $\mathcal{L}_1$ either has all robots with color `chord` or, has all robots of color `chord` except at most two robots on the terminal. For the latter case, by Claim 6 all robots on $\mathcal{L}_1$ will have color `chord` eventually at a time, say t', where $t' > t$ (Figure 4.6(a)). Until then r and r' do nothing even if they are activated. after t' when r' activates next it sees $\mathcal{L}_1$ has all robot with color `chord` and thus move left to $\mathcal{L}_1$. with similar argument after moving to $\mathcal{L}_1$ eventually r' end up with color `chord`. next when r is activated on $\mathcal{L}_2$ it must see that all robot on $\mathcal{L}_1$ has color `chord` and thus moves left contrary to the assumption.

Case 2: Next suppose r is singleton on $\mathcal{L}_2$ and r' is singleton on some $\mathcal{L}_k$ where $k > 2$. Note that between $\mathcal{L}_2$ and $\mathcal{L}_k$ there are no other robots (Claim 4 and Observation 4.2). Note that all other robots on the right of $\mathcal{L}_k$ at time t have color `off` and upon activation they do nothing from time t onwards as they never see all robots with color `chord` on their left immediate vertical line as r never moves to $\mathcal{L}_1$. In this case whenever r' is activated it sees r singleton on $\mathcal{L}_2$ and moves left to $\mathcal{L}_{k-1}$ (Figure 4.6(b)). This way eventually r' reaches $\mathcal{L}_2$ along with r. This is the same configuration as described in case 1. So, again it would reach a contradiction.

Case 3: Let, r is singleton on $\mathcal{L}_2$ and r' is not singleton on $\mathcal{L}_k$ at time t where $k > 2$. Then if r' has a pending move, it moves to $\mathcal{L}_{k-1}$, otherwise if r' activates after time t, it does nothing. For the former case, after r' moves to $\mathcal{L}_{k-1}$, it either falls into the case 1 or, the case 2. Now, for the latter case, if r' does nothing on

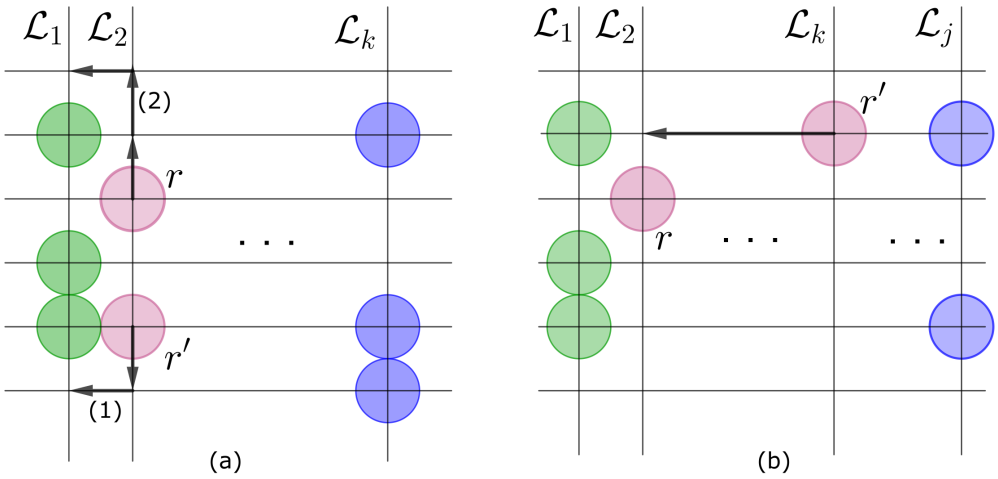


Figure 4.6: **(a)** r and r' are on $\mathcal{L}_2$ with color `moving1`. If r does not move left, r' moves left upon seeing all robots on $\mathcal{L}_1$ of color `chord` and in the next activation changes color to `chord`. Next r moves to $\mathcal{L}_1$. **(b)** r with color `moving1` is singleton on $\mathcal{L}_2$ and r' with color `moving1` is also singleton on $\mathcal{L}_k$ $(k > 2)$. If r do not move to $\mathcal{L}_1$, r' moves and reaches $\mathcal{L}_2$ and converts to the case in (a).

$\mathcal{L}_k$ then r always remains singleton on $\mathcal{L}_2$ and no new robot moves onto $\mathcal{L}_1$ from time t onwards. So, at time t if not all robots on $\mathcal{L}_1$ is of color `chord`, eventually they all become of color `chord (Claim 6)`. So, upon the next activation, r must move to $\mathcal{L}_1$. This is again a contradiction to our assumption.

Case 4: In this case, let us assume r is only robot in the configuration of color `moving1` on $\mathcal{L}_2$ at time t. If from t onwards no new robot changes its color to `moving1` then eventually there will be a time, say $t' > t$ when all robots on $\mathcal{L}_1$ have color `chord`. Also, no new robot moves to $\mathcal{L}_1$ after t'. Thus, when r activates next it must move to $\mathcal{L}_1$. So, let us now assume there is a time when a new robot r' changes its color to `moving1`. Then as described in the previous cases, r will move to $\mathcal{L}_1$ contradicting our assumption. So, There will be a time when r moves to $\mathcal{L}_1$ from $\mathcal{L}_2$.

Hypothesis: For some $j > 2$ and for any $i \leq j$, a robot having a color `moving1` on $\mathcal{L}_i$ moves to $\mathcal{L}_{i-1}$ where $i \geq 2$.

Inductive step: Let r be a robot on $\mathcal{L}_{j+1}$ with color `moving1` at a time t, where $j \geq 2$. Now by Claim 4, there can be at most another robot, say r' with color

`moving1` on $\mathcal{L}_k$ at time t ($k > 1$). If possible let r never move to $\mathcal{L}_j$. Now there are two cases.

Case 1: For the first case let us assume $k \leq j + 1$ (Figure 4.7(a)). First assume that r is singleton on $\mathcal{L}_{j+1}$ at time t. Then $k < j+1$. By Observation 4.2 between $\mathcal{L}_1$ and $\mathcal{L}_{j+1}$ there are no other robot except r' on $\mathcal{L}_k$ at time t. Note that by induction hypothesis r' eventually moves to $\mathcal{L}_1$ and changes its color to `chord`, say at a time $t' > t$. Before that no robot on $H_R^O(r)$ does anything even if they are activated as they can not change their color from `off` to `moving1` because they can never see $\mathcal{L}_1$ as their left immediate vertical line due to r being on $\mathcal{L}_{j+1}$. So, after t', whenever r activates it sees $\mathcal{L}_I(r) = \mathcal{L}_1$ where all robots have color `chord`. Thus eventually it moves left to $\mathcal{L}_j$. Also, before t' if r is activated and sees r' on some $\mathcal{L}_{k'}$ where $1 < k' \leq k$ then it moves left to $\mathcal{L}_j$ contrary to the assumption. So, let r is not singleton on $\mathcal{L}_{j+1}$ at time t. Let there are p robots on $\mathcal{L}_{j+1}$ at time t (Figure 4.7(a)). Now if r' is on some $\mathcal{L}_k$ where $1 < k < j + 1$ then, it must be singleton on $\mathcal{L}_k$. Now, until r' reaches $\mathcal{L}_1$ and changes its color to `chord`, no other robot on the right of r' does anything. By the induction hypothesis, it can be ensured that r' will reach $\mathcal{L}_1$, and eventually all robots on it would change their color to `chord` at a time say, t'. Now since $p > 1$, there must exist another robot, say r_1, having color `off` on $\mathcal{L}_{j+1}$ which is terminal on it at time t'. Also at time t' there is no other robot in between $\mathcal{L}_1$ and $\mathcal{L}_{j+1}$ and no other robot except r_1 moves in between $\mathcal{L}_1$ and $\mathcal{L}_{j+1}$ from time t' onwards unless r_1 reaches $\mathcal{L}_1$ and changes its color to `chord`. This is because only r_1 can change its color to `moving1` from `off`, after time t' and until it reaches $\mathcal{L}_1$ and changes color to `chord` (by Claim 4). This implies, after t' whenever r_1 is activated it will change its color to `moving1` from `off` and will eventually move to $\mathcal{L}_j$ seeing all robots on $\mathcal{L}_1$ having color `chord`. Now by the hypothesis r_1 will reach $\mathcal{L}_1$ and change its color to `chord` eventually at a time, say t_1. Now at time t_1, $\mathcal{L}_{j+1}$ has $p - 1$ robots. if $p - 1 = 1$ then r becomes singleton on $\mathcal{L}_{j+1}$ and as described above it will eventually move to $\mathcal{L}_j$. otherwise there always will be a robot on $\mathcal{L}_{j+1}$ which is terminal and has color `off`. For this case, the terminal robot will eventually reach $\mathcal{L}_1$ as described above and will change its color to `chord`. This implies eventually number of robots on $\mathcal{L}_{j+1}$ will decrease until only r remains.

For this case, r moves to $\mathcal{L}_j$ eventually, contrary to the assumption.

Case 2: Let $k > j+1$ (Figure 4.7(b)). This implies r is singleton on $\mathcal{L}_{j+1}$ also, there are no robots between $\mathcal{L}_1$ and $\mathcal{L}_k$ except r. First assume r' is singleton on $\mathcal{L}_k$ (Figure 4.7(b)). Then upon seeing only r on $\mathcal{L}_I(r')$ it will first move left until it reaches $\mathcal{L}_{j+1}$. Now as argument-ed for the previous case 1, r' will reach $\mathcal{L}_1$ eventually and change its color to `chord`. Note that since r is singleton on $\mathcal{L}_{j+1}$, no robot from $H_R^Q(r)$ does anything upon activation. Also, there are no robots in between $\mathcal{L}_1$ and $\mathcal{L}_{j+1}$ and all robots on $\mathcal{L}_1$ has color `chord`. Note that now when r will be activated it will always see $\mathcal{L}_I(r) = \mathcal{L}_1$ where all robots have color `chord` and thus eventually it will move to $\mathcal{L}_j$. This is again a contradiction. Thus, let us assume at time t, r' is not singleton on $\mathcal{L}_k$. For this case, r' will not do anything even if it is activated. Thus, eventually r will see all robots with color `chord` on $\mathcal{L}_I(r)$ (i.e., $\mathcal{L}_1$). Thus r will move to $\mathcal{L}_j$.

So considering another robot, r' having color `moving1` on $\mathcal{L}_k$ ($k > 1$) at time t we always reach a contradiction. So, let at time t, r is the only robot with color `moving1` on $\mathcal{L}_{j+1}$ ($j \geq 2$). For some pending move, another robot may change its color to `moving1` later, say at $t' > t$. Now before t' if $\mathcal{L}_1$ has all robot with color `chord` and r is activated, it will move to $\mathcal{L}_j$. Otherwise, we reach the same configuration described in case 1 and case 2. So, in this case, also, we have a contradiction. Thus, r will eventually move to $\mathcal{L}_j$. □

Lemma 4.3. *After $\mathcal{L}_1$ is fixed, let $\mathcal{L}_k$ be the first vertical line with a robot r of color `off` at a time t. Then r eventually changes its color to `moving1` while executing Algorithm 6.*

Proof. Let after $\mathcal{L}_1$ is fixed, t be a time when $\mathcal{L}_k$ is the first vertical line that contains a robot r having color `off`.This implies, all robots between $\mathcal{L}_1$ and $\mathcal{L}_k$ (if exists) has color `moving1` at time t. Now let from t onwards r never changes its color to `moving1`. Let there be $p \geq 1$ robots at time t on $\mathcal{L}_k$. Now we have two cases.

Case 1: Let $p = 1$. That is r is singleton on $\mathcal{L}_k$ at time t (Figure 4.8(a)). Then robots on $H_R^Q(r)$ must be of the color `off` at time t (Observation 4.2) and does

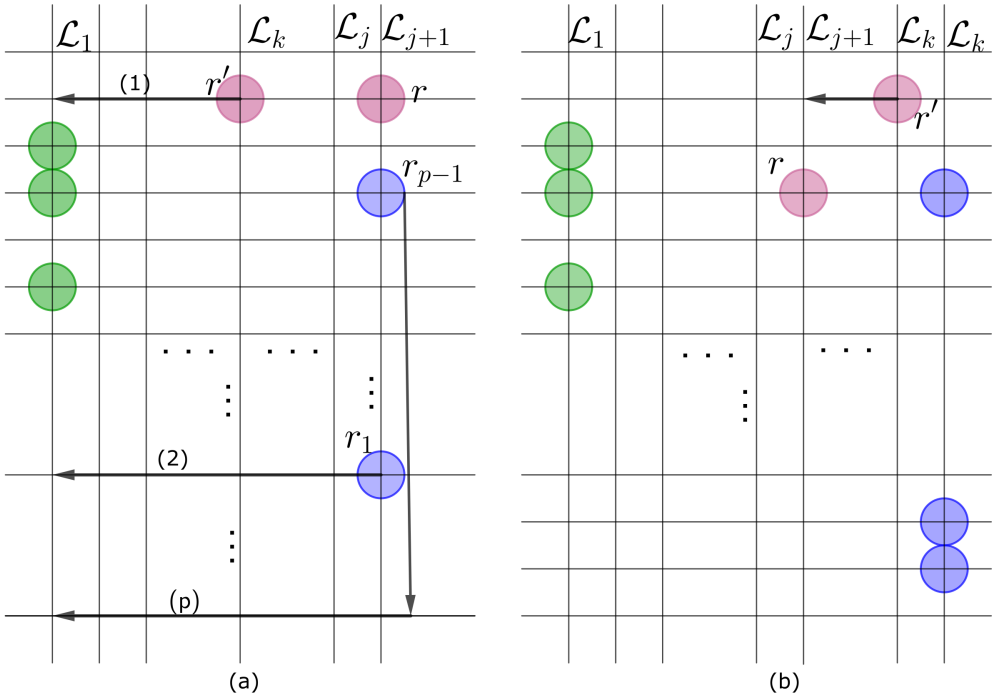


Figure 4.7: (a) r' moves to $\mathcal{L}_1$ first and changes color to `chord`. Next r_1 changes color to `moving1` and do the same as r'. Eventually r_{p-1} also moves to $\mathcal{L}_1$ and changes color to `chord`. Next, whenever r activates it moves and eventually reaches $\mathcal{L}_j$. **(b)** Here if r does not move, singleton r' moves left. After reaching $\mathcal{L}_{j+1}$ it transforms into case I.

nothing upon activation from time t onwards as r does not change its color. Now, by lemma 4.2, all robots in between $\mathcal{L}_1$ and $\mathcal{L}_k$ eventually moves to $\mathcal{L}_1$ and change their color to `chord` at a time, say t_0. And within this time no new robot changes its color to `moving1` from `off`. Thus from t_0 onwards, if r does not change its color, the configuration remains unchanged. Now, when r is activated after time t_0, it sees all robots on $\mathcal{L}_I(r) = \mathcal{L}_1$ has color `chord` and thus changes its color to `moving1`, contrary to our assumption.

Case 2: Let $p > 1$ (Figure 4.8(b)). For this case, we will show that the number of robots on $\mathcal{L}_k$ eventually decreases until it becomes one. Which we already discussed in case 1. First observe that no robot on $H_R^O(r)$ at time t, moves to $\mathcal{L}_k$ as they will remain of color `off` (observation 4.2) from time t onwards if r does not change its color to `moving1`. So the number of robots on $\mathcal{L}_k$ never increases

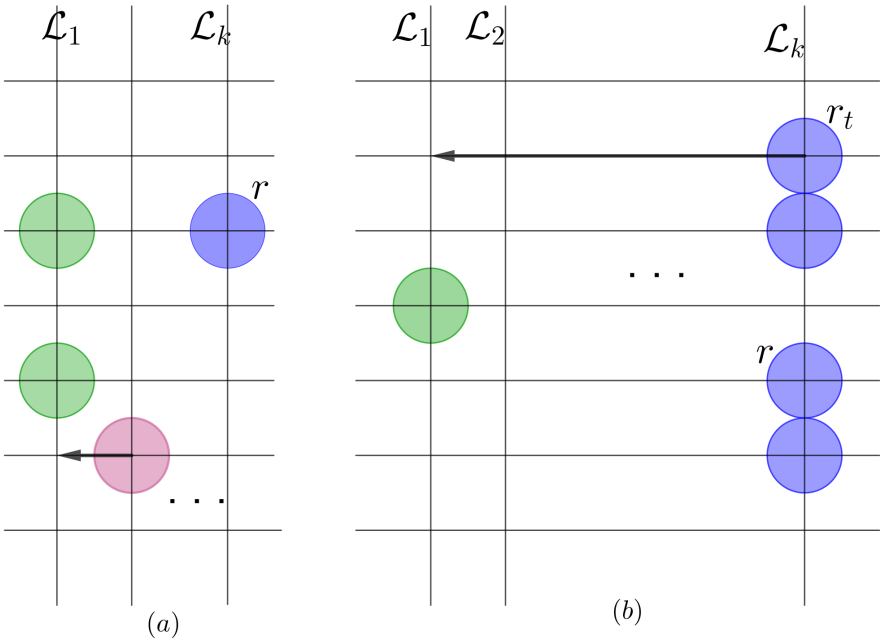


Figure 4.8: **(a)** r with color `off` is singleton on $\mathcal{L}_k$ the robots between $\mathcal{L}_1$ and $\mathcal{L}_k$ moves to $\mathcal{L}_1$ and changes color to `chord`. Next, whenever r activates it changes color to `moving1`. **(b)** r is not singleton on $\mathcal{L}_k$. All robots between $\mathcal{L}_1$ and $\mathcal{L}_k$ has already moved to $\mathcal{L}_1$ and changed color to `chord`. Next, whenever r_t activates it changes color `moving1` and moves left decreasing the number of robots on $\mathcal{L}_k$ and eventually transforming into the case shown in (a).

from time t onwards. Now let the number of robots on $\mathcal{L}_k$ never decrease i.e., it remains the same. This implies, no robot from $\mathcal{L}_k$ changes its color to `moving1` from time t onwards. Now by Lemma 4.2 all robots in between $\mathcal{L}_1$ and $\mathcal{L}_k$ have color `moving1` at time t and they will eventually move to $\mathcal{L}_1$ and change their color to `chord` at a time, say $t_1 \geq t$. Now let r_t be the terminal robot on $\mathcal{L}_k$ which is activated first after time t_1. Now upon activation, it must see all robots on $\mathcal{L}_I(r_t) = \mathcal{L}_1$ has color `chord` and thus changes its color to `moving1` contrary to our assumption. Thus the number of robots on $\mathcal{L}_k$ will decrease and eventually have only r. So according to case 1, we will again reach a contradiction. As for both cases contradictions are achieved, our assumption that r never changes its color to `moving1` from time t onwards is false. So there is a time when r changes its color to `moving1`. □

So using Lemma 4.1 we guarantee that within a finite time after the first move by any robot from the initial configuration $\mathcal{L}_1$ will be fixed. Let $\mathcal{L}_k$ be the first vertical line on the right of $\mathcal{L}_1$ where a robot of color `off` exists ($k > 2$). So all robots in between $\mathcal{L}_1$ and $\mathcal{L}_k$ has color `moving1`. Thus by Lemma 4.2 eventually all these robots will move to $\mathcal{L}_1$. Now observe that all robots on $\mathcal{L}_k$ will have color either `moving1` or `off`. Now using Lemma 4.2 it can be said that all robots of color `moving1` on $\mathcal{L}_k$ will move to $\mathcal{L}_1$ eventually. Also by Lemma 4.3 all robots of color `off` on $\mathcal{L}_k$ change their color to `moving1` and move to $\mathcal{L}_1$ eventually. Thus Within finite time all robots of $\mathcal{L}_k$ move to $\mathcal{L}_1$. Now if there are no other robots on the right of $\mathcal{L}_k$ then we are done. Otherwise the first vertical line on the right of $\mathcal{L}_1$ containing a robot of color `off`, shifts right. Eventually, there will be one such line that does not have any other robot on its right and all other robots will be on $\mathcal{L}_1$. Now as described above all other robots of that line will also eventually move to $\mathcal{L}_1$. In this moment all non-terminal robots on $\mathcal{L}_1$ will have color `chord` and the terminal robots either will have color `chord` or `moving1` (Claim 5). From this above discussion, we can have the following theorem.

Theorem 4.1. *There exists a time t when all robots move to a single line with non-terminal robots having color* ***`chord`*** *and terminal robots having color either* ***`chord`*** *or,* ***`moving1`*** *by executing the Algorithm 6 from any initial configuration assuming one axis agreement, under asynchronous scheduler.*

We now proof the following theorem which states that a configuration where all robots are on a single line with all non-terminal robots having color `chord` and terminal robots having color either `chord` or, `moving1` will eventually change into a P1FC. This theorem ensures the termination of Algorithm 6.

Theorem 4.2. *Let in a configuration $\mathcal{C}$, all robots are on a single line where each non-terminal robots have color* ***`chord`*** *and the terminal robots have color either* ***`chord`*** *or* ***`moving1`****. Then in finite time, the configuration will change into a P1FC.*

Proof. By the theorem 4.1, there exists a time t when all robots will be on a single line. All robots on that line which are not terminal must have color `chord`

at time t and the terminal robots can have color either `moving1` or, `chord` at time t. Let both terminal robots, say r_1 and r_2, has color `moving1` at time t. Then among r_1 and r_2, whichever is activated first, say r_1 without loss of generality, must see a robot of color `chord` on $\mathcal{L}_V(r_1) = \mathcal{L}_1$ and changes its color to `chord`. The guarantee that r_1 will see a robot of color `chord` comes from the fact that the non-terminal robots with color `chord` on $\mathcal{L}_1$ do not move out of $\mathcal{L}_V(r_1)$ until they see a robot with color `diameter` on $\mathcal{L}_V(r_1)$. So there must exist a time $t_1 > t$ when all robots are on $\mathcal{L}_1$ and all but at most one terminal robot say r_2 has color `chord` and r_2 has either color `chord` or `moving1`. Now there are two cases. In the first case, we assume all robots have color `chord` at t_1, and in the second case we assume all robots except r_2 have color `chord` and r_2 has color `moving1`.

Case I: Let us consider the case when all robots on $\mathcal{L}_1$ has color `chord` at time t_1. Now after t_1 whichever robot of r_1 and r_2 is activated first, changes its color to `diameter`. Note that both terminal robots can have color `diameter` too. So, there exists a time $t_2 > t_1$ when all robots are on $\mathcal{L}_1$, all non-terminal robots have color `chord` and at least one terminal robot has color `diameter`.

Case I(a): Now if both the terminal robots r_1 and r_2 have color `diameter` at time t_2 (Figure 4.9(a)), then the non terminal robots on $\mathcal{L}_1$ that can see r_1 or r_2 with color `diameter` moves either left or right once by executing ChordMove subroutine. After a robot moves to the left or right of $\mathcal{L}_V(r_1)$ by executing ChordMove, the next non-terminal robot on $\mathcal{L}_V(r_1)$ can now see at least one robot of color `diameter` and thus execute the ChordMove subroutine. This way all non-terminal robots on $\mathcal{L}_V(r_1)$ will execute Chordmove at least once. Note that a robot that has executed ChordMove once does not move again until P1FC is achieved. This is because the robot does not execute Algorithm 6 as it will see a robot of color `diameter` which is not on its own vertical line. Now Let t_3 be a time the last non-terminal robot on $\mathcal{L}_V(r_1)$ moved right or left after executing ChordMove. Then we can ensure that only r_1 and r_2 are on $\mathcal{L}_2$ with color `diameter` at time t_3 and all other robots are on $\mathcal{L}_1$ and $\mathcal{L}_3$ with color `chord` and also they are strictly between $\mathcal{L}_H(r_1)$ and $\mathcal{L}_H(r_2)$. Hence at t_3, the configuration becomes a P1FC.

Case I(b): Now let at time t_2, only r_1 has color `diameter` and r_2 has color `chord` (Figure 4.9(b)). Now if r_2 is activated before any of the non-terminal robots move by executing ChordMove it changes its color to `diameter`. For this case, we can show that eventually P1FC will be achieved (using a similar argument as Case I(a)). So, let before r_2 is activated a robot which is not terminal on $\mathcal{L}_1$ and sees r_1 executes ChordMove and moves left. Then r_2 never changes its color to `diameter` until all non-terminal robots move left or right of $\mathcal{L}_V(r_1) = \mathcal{L}_V(r_2)$ by executing ChordMove. When all such robots move, r_2 then can see r_1 with color `diameter` and then it changes its color to `diameter`. Note that after r_2 changes its color `diameter` the configuration becomes a P1FC.

Case II: Let at time t_1 all but one terminal robot, say r_2 (without loss of generality), on $\mathcal{L}_1$ has color `chord`. Let r_2 has color `moving1` at time t_1. Now, if r_1 is activated and sees a robot of color `chord` on its vertical line and sees no robot on both of its left and right open halves then r_1 changes its color to `diameter`. Let this happens at a time $t_4 > t_1$. Then similar to the above discussion all non-terminal robots on $\mathcal{L}_V(r_1)$, that see r_1 after time t_4, on their own vertical line, execute ChordMove and move left or right. If at least one such non-terminal robot has already executed ChordMove then even if r_2 is activated after that, with color `moving1` it does not change its color as it sees its left open half non-empty.

Case II(a): If r_2 is activated before any robot moves from $\mathcal{L}_V(r_1) = \mathcal{L}_V(r_2)$ then it sees a robot with color `chord` on $\mathcal{L}_V(r_2)$ and sees $H^O_L(r_2)$ empty. Thus r_2 in this case changes its color to `chord` at a time say $t_5 > t_1$. In the configuration, if $t_5 > t_4$ then at time t_5, all robots have color `chord` except r_1, which has color `diameter`. This is similar to the case I(b). Thus in this case eventually the configuration will become a P1FC. So, let $t_5 \leq t_4$. If $t_5 < t_4$, then at t_5, all robots will be on a single line with color `chord`. This is similar to the case I and thus eventually the configuration will change to a P1FC. Now if $t_5 = t_4$, then at t_5, r_1 will have color `diameter` and r_2 will have color `chord` which is similar to the case I(b). Thus again the configuration will eventually become a P1FC.

Case II(b): Now let after t_1, r_2 is activated for the first time when at least one

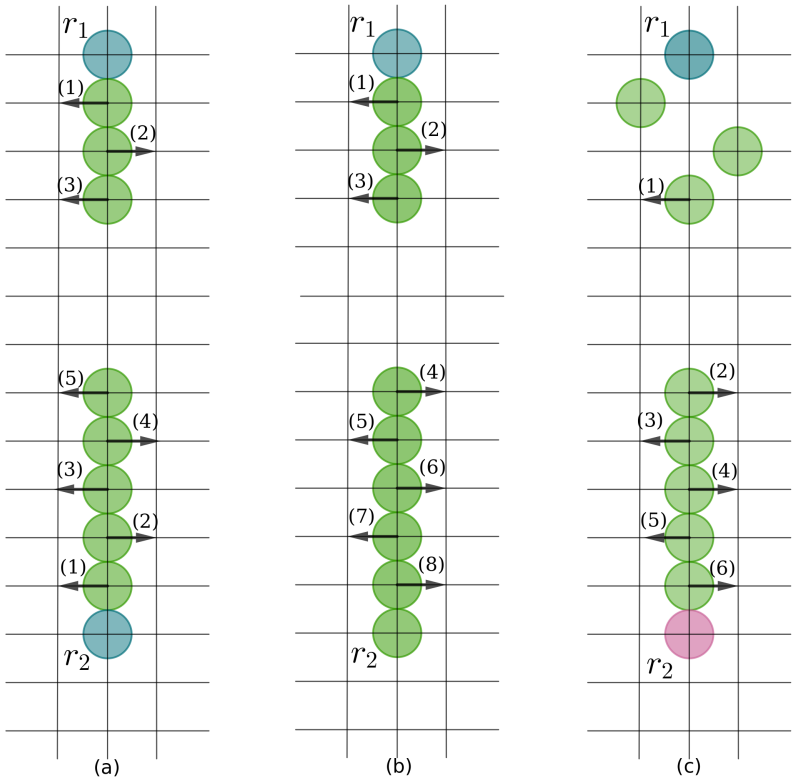


Figure 4.9: **(a)** Both terminal robots have color `diameter` in $\mathcal{C}(t_2)$. **(b)** r_1 has color `diameter` and r_2 has color `chord` in $\mathcal{C}(t_2)$. r_2 does not change color `diameter` until all non terminal robots of color `chord` execute ChordMove exactly once and moves to left or right. **(c)** r_2 still has color `moving1` while some non terminal robots from the line already executed ChordMove. In this case also r_2 changes color to `diameter` when all non-terminal robots move left or right. The numbers in the bracket denote the order in which the robots move.

non-terminal robot already executed ChordMove (Figure 4.9(c)). Then r_2 does not change its color even if it is activated as it does not have $H_L^O(r_2)$ non-empty. Now When all non-terminal robots execute ChordMove once then on $\mathcal{L}_V(r_2)$ there are only two robots r_1 with color `diameter` and r_2 with color `moving1`. Now when r_2 is activated again, it sees r_1 with color `diameter` on $\mathcal{L}_V(r_2)$ and changes its color to `diameter`. After this, the configuration again becomes a P1FC.

For all of the above cases, it can be ensured that A configuration where all robots are on a single line with all non-terminal robots having color `chord` and terminal robots having color either `chord` or, `moving1` will change into a P1FC within finite time. □

4.3 Modification to Algorithm 6 for Line Formation

The P1FC formation algorithm FormP1FC(Algorithm 6) can be easily modified to use it as a line formation algorithm. Let us rename the color `chord` as `line`. In this modified version the color `diameter` is not necessary. So the checks with the"if" statements in Line 1 of Algorithm 7, Algorithm 8 and Algorithm 9 is not there in the modified version. Also, remove the "Else" block starting at line 17 upto Line 19 of Algorithm 8 and "Else if" block of Algorithm 9. And finally, Whenever a robot r finds that *r.color* is `line` it terminates. In this modified version when all robots terminate they are on a single vertical line on the grid. Also, note that the subroutine ChordMove() is not executed by any of the robots in this modified line formation algorithm.

4.4 Concluding Remarks

In this chapter, we took a closer look at the Line Formation problem, an essential step in forming a circle on a grid and a foundation for the algorithm discussed in Chapter 5. We introduced and analyzed the FormP1FC algorithm, showing how it enables robots to transition from any starting configuration to a P1FC

configuration. We also argued how this algorithm can be adapted for simple line formation with only minor modifications and with one less color used in forming P1FC. This work provides a solid basis for robotic coordination in grid-based environments as well as it works as a foundation for solving more complex formation problems in grid-based terrain. We discuss one such problem in the next chapter.

There are several scopes for the betterment of this work in the future. First, we do not know yet if the 3 colors used in our line formation algorithm are necessary or not. So it would be really interesting to find out about that. Also, the necessity of one-axis agreement to solve line formation has not been discussed yet even for point robots on grid-based terrain.

Chapter 5

Circle Formation on Infinite Grid by Luminous, Fat Robots

In this chapter, we explore the circle formation problem on an infinite grid-based discrete terrain. This problem has been extensively studied in the Euclidean plane for many years [49, 74] and continues to be a subject of recent research [52, 77]. Its first investigation in grid-based terrains was presented in [2]. The robots considered in that work are modeled as opaque points. However, in practical scenarios, robots possess dimensions, even if they are minimal. Therefore, we examine the problem introduced by Adhikary et al. in [2] under the fat opaque robot model. The next section elaborates on the model used, along with relevant definitions and terminologies.

5.1 Robot Abilities for Circle Formation and Related Definitions

In this chapter, we adopt the same model and notations introduced in Chapter 4. A circle on a grid-based terrain differs from one on a plane as the points on the circle have to be grid points. This requires us to first define a circle on an infinite grid. This can be done in various ways. In this work, we define an approximated circle on the grid corresponding to a given circle on the plane. The circumference of this approximated circle on the grid is defined as follows.

Definition 5.1 (***Grid Circumference***)**.** *Let $\mathcal{CIR}$ be a circle on the plane on which the infinite grid is embedded. Let L_H be a horizontal grid line that intersects the circle $\mathcal{CIR}$ on at most two points A and A'. Now,*

- *If A and A' are grid points, we say that only A and A' on L_H are on the circumference of $\mathcal{CIR}$.*
- *Otherwise, if A and A' are not grid points, we call a grid point (a, b) on the circumference of $\mathcal{CIR}$ on L_H if the line joining the grid points (a, b), $(a + 1, b)$ or (a, b), $(a - 1, b)$ contains exactly one of A and A'.*

The set of all such grid points on the circumference of the circle is called the Grid Circumference.

For simplicity by the term "on the circumference of the circle" we will always mean on some grid point which is in the set Grid Circumference.

Now, let us define the problem (CF_FAT_GRID) formally.

Definition 5.2 (***Problem Statement of*** CF_FAT_GRID)**.** *Let a finite set of fat robots of the same radius be initially located on distinct vertices of an infinite grid $\mathcal{G}$. We say that the Circle formation by Fat robots on an infinite grid (CF_FAT_GRID) is solved if the following condition is satisfied.*

> *$\exists t > 0$ such that in $\mathcal{C}(t)$ all robots are on the grid circumference of a circle and $\forall t' \geq t$ $\mathcal{C}(t') = \mathcal{C}(t)$.*

5.2 Circle Formation

The circle formation algorithm $CIRCLE\_FG$, described in this section solves the CF_FAT_GRID problem stated earlier in Definition 5.2. This Algorithm works in two phases. In first phase a robot r execute the algorithm FORMP1FC(r) (Algorithm 6 in Chapter 4) to make a P1FC configuration. Then in the second

phase, it executes FORMCIRCLE(r) and eventually forms the required circle. The pseudocode is given below.

Algorithm 10: *CIRCLE_FG*

```
1  r ← myself;
2  Initialize r.phase = 0;
3  if r.color = off then
4  |   if r sees no robot of color diameter then
5  |   |   r.phase = 1
6  |   else
7  |   |   r.phase = 2
8  else if r.color = moving1 then
9  |   if r sees no robot of color diameter ∨ r sees a robot of color diameter on L_v(r)
   |    then
10 |   |   r.phase = 1
11 |   else
12 |   |   r.phase = 2
13 else if r.color = chord then
14 |   if r sees no robot of color diameter ∨ r sees at least one robot with color
   |    diameter on L_v(r) ∨ r sees at least one robot with color diameter and at least
   |    one agent with color chord on either L_I(r) or R_I(r) then
15 |   |   r.phase = 1
16 |   else
17 |   |   r.phase = 2
18 if r.phase = 1 then
19 |   Execute FORMP1FC(r)
20 else if r.phase = 2 then
21 |   Execute FORMCIRCLE(r)
```

In the *CIRCLE_FG* algorithm, a robot first decides the phase it is currently in and then according to that either executes FORMP1FC or, FORMCIRCLE.

A robot with color `off` decides it is in phase 1 if it does not see any other robot having color `diameter` otherwise, it decides to be in phase 2. A robot having color `chord` decides that it is in phase 1 if any of the following scenarios occur in its view. Either it sees no robot having color `diameter` or, it sees at least one robot having color `diameter` on the same vertical line or, it sees at least one robot having color `diameter` and at least another robot having color `chord` on its left or, right immediate vertical line occupied by a robot. On the other hand, the robot decides to be in phase 2. A robot having color `moving1` decides to be in phase 1 if either it sees no robot having color `diameter` or, sees a robot with color

`diameter` but on the same vertical line and for any other view it decides that it is in phase 2. Robots having color `diameter` and `done` do nothing ever so they don't have to distinguish between the phases. FORMP1FC is already described in Algorithm 6 in the previous chapter. Here in the following subsection, we describe the subroutine FORMCIRCLE used in Algorithm $CIRCLE\_FG$.

5.2.1 Description of subroutine FORMCIRCLE(r)

A robot r executes subroutine FORMCIRCLE(r) only after the P1FC configuration has already been formed. Let at time t the configuration become a P1FC for the first time. Let r_1 and r_2 be the only robots on $\mathcal{L}_2$ and they have color `diameter`. Note that these two robots are already terminated While they changed their color to `diameter` during the execution of FORMP1FC subroutine. Now, these two robots help the other robots to agree on the circle to be formed. The line segment of $\mathcal{L}_V(r_1)$ between r_1 and r_2 will be agreed by other robots as a diameter of the circle to be formed. So, if a robot sees both r_1 and r_2 it knows the circle, say $\mathcal{CIR}$. While executing the algorithm FORMCIRCLE, a robot r with color `chord` executes the subroutine CHORDFORMCIRCLE(r) (Algorithm 12). If the robot has color `off` it executes the subroutine OFFFORMCIRCLE(r) (Algorithm 13) and executes MOVING1FORMCIRCLE(r) (Algorithm 14), if the robot has color `moving1`. A robot having color `done` terminates immediately.

Now, observe that in a P1FC, each horizontal line between $\mathcal{L}_H(r_1)$ and $\mathcal{L}_H(r_2)$ contains at most one robot. Now each of these horizontal lines has exactly two grid points on the circumference of the circle $\mathcal{CIR}$, one on the left of the agreed diameter and the other on the right. So, for any robot r, if r terminates at a grid point on $\mathcal{L}_H(r)$ which is on the circumference of $\mathcal{CIR}$, we say that circle is formed. The target of executing the subroutine FORMCIRCLE is to make sure that any robot r terminates at the grid point on $\mathcal{L}_H(r)$ which is on the circumference of $\mathcal{CIR}$ and on the left (resp. right) of the agreed diameter if in the P1FC, r is on the left (resp. right) of the agreed diameter.

A robot r executes FORMCIRCLE after the configuration becomes a P1FC. In this

scenario. There are exactly two robots with color `diameter`, say r_1 and r_2 on the same vertical line, and all other robots have color `chord` on $\mathcal{L}_I(r_1)$ and $\mathcal{R}_I(r_1)$. Difference between number of robots on $\mathcal{L}_I(r_1)$ and $\mathcal{R}_I(r_1)$ is at most two. Note that the vertical line $\mathcal{L}_V(r_1)$ divides the circle on two halfs $H_L^O(r_1)$ and $H_R^O(r_1)$. Here, we describe FORMCIRCLE for the robots only in $H_L^O(r_1)$. The algorithm for the robots in $H_R^O(r_1)$ will be similar.

Algorithm 11: FORMCIRCLE(r)

```
1  Procedure FormCircle(r)
2      d = distance between two robots with color diameter
3      robot with color diameter is on right (resp. left) open half of r
4      c = midpoint of two robots with color diameter
5      if r.color = chord then
6          Execute chordFormCircle(r)
7      else if r.color = off then
8          Execute offFormCircle(r)
9      else if r.color = moving1 then
10         Execute moving1FormCircle(r)
11     else if r.light = done then
12         terminate;
```

Note that due to the procedure CHORDMOVE in the first phase before P1FC is formed, all robots with color `chord` must be strictly between the horizontal lines passing through r_1 and r_2. Observe that, during the first phase, if a robot r of color `chord` sees at least one robot of color `diameter` then there are two possibilities. Either r sees at least one robot of color `diameter`, say r_1, on $\mathcal{L}_V(r)$ or, it sees r_1 on $\mathcal{R}_I(r)$ (resp. $\mathcal{L}_I(r)$) while there must be another robot of color `chord` on $\mathcal{R}_I(r)$ (resp. $\mathcal{L}_I(r)$). In that case r executes FORMP1FC(r). We have shown that (Lemma 5.1) during execution of FORMCIRCLE(r) (i.e., when r has $phase = 2$) a robot r of color `chord` always sees at least one robot of color `diameter` which can not be on $\mathcal{L}_V(r)$ (as r never moves to the vertical line where the robots of color `diameter` are located). Now if r sees a robot r_1 of color `diameter` on $\mathcal{R}_I(r)$ (resp.$\mathcal{L}_I(r)$) then, it must see the other robot r_2 of color `diameter` on $\mathcal{R}_I(r)$ (resp. $\mathcal{L}_I(r)$) too. But the difference from Phase 1 is that, here $\mathcal{L}_V(r_1)$ does not have any other robots of color `chord`. Thus a robot of color `chord` can always distinguish between Phase 1 and Phase 2.

Now, a robot that is terminal on $\mathcal{L}_I(r_1)$, say r with color `chord`, must see both the robots r_1 and r_2. In this case, it moves toward its left (Line 6,7 in Algorithm 12). Before this move r changes its color to `off` only if the horizontal distance of r from r_1 or r_2 is $\lceil \frac{d}{2} \rceil - 1$ (Line 3, 4 in Algorithm 12). Observe that due to this rule after a finite time all robots which were initially on $\mathcal{L}_I(r_1)$ with color `chord`, reach a vertical line which is $\lceil \frac{d}{2} \rceil$ distance away from r_1 and r_2 with color `off`.

Algorithm 12: CHORDFORMCIRCLE(r)

```
1 if r sees only two robots with color diameter on R_I(r) (resp. L_I(r)) then
2     if r is terminal then
3         if horizontal distance from robot with color diameter = ⌈d/2⌉ − 1 then
4             r.color = off
5             move horizontally away from robot with color diameter
6         else
7             move horizontally away from robot with color diameter
```

Now we claim that a robot with color `off` always sees at least one of r_1 or r_2 (Lemma 5.2). By this condition, a robot with color `off`, can identify whether it is in Phase 1 or Phase 2. Let us name the vertical line $\mathcal{L}_V(r_1)$ as v_0, and v_i be the i-th vertical line on the left of v_0. So, after a finite time, all robots will be on $v_{\lceil \frac{d}{2} \rceil}$ strictly between $\mathcal{L}_H(r_1)$ and $\mathcal{L}_H(r_2)$ having color `off`. In this configuration, all the robots can see both r_1 and r_2 and thus can calculate the point c which is equidistant from both r_1 and r_2 on v_0. There can be at most two robots on $v_{\lceil \frac{d}{2} \rceil}$ which are nearest to c. Also, if there are two such robots then there can be no other robots strictly between them on $v_{\lceil \frac{d}{2} \rceil}$. Thus a nearest robot to c on $v_{\lceil \frac{d}{2} \rceil}$ can understand if it is nearest to c. Such a robot, say r, first moves right after changing the color to `moving1` (Line 3,4 in Algorithm 13) and does not move further until it sees no other robots on its left (c.f in Algorithm 14 a robot r with color `moving1` after seeing a robot r' with color `diameter` on $\mathcal{R}_I(r)$ can only move if it doesn't see any robot having color `off` or, `moving1` on $\mathcal{L}_I(r)$). Now , after r moves, another robot, say r', on $v_{\lceil \frac{d}{2} \rceil}$ moves right only if it sees r with color `moving1` on its immediate right vertical line $v_{\lceil \frac{d}{2} \rceil - 1}$ and sees no other robot on the left of v_0 strictly between $\mathcal{L}_H(r)$ and $\mathcal{L}_H(r')$ (Line 5-7 in Algorithm 13). Thus there exists a time when all robots on the left of v_0 are on $v_{\lceil \frac{d}{2} \rceil - 1}$ and have color `moving1`. We call this configuration a $(\lceil \frac{d}{2} \rceil - 1)-$*Left Sub Circle Configuration*

($(\lceil \frac{d}{2} \rceil - 1)-LSCC$). More formally,

Definition 5.3 ($j-$ Left Sub Circle Configuration Left (j-LSCC))**.** *A configuration where r_1 and r_2 with color* ***diameter*** *be the only two robots on v_0 is called a j- Left sub circle configuration (Figure 5.1) if*

1. *All robots of color* ***moving1*** *on the left of v_0 are the only robots on vertical line v_j.*
2. *There are no robots strictly between v_j and v_0.*
3. *All robots on the left of v_0 are strictly between $\mathcal{L}_H(r_1)$ and $\mathcal{L}_H(r_2)$ and each horizontal line contains at most one robot.*
4. *All robots on left of v_j (if any) must be of color* ***done***

Let r be a robot on the left of v_0 that sees both r_1 and r_2 with color `diameter`. Then it can find out the point c which is equidistant from r_1 and r_2 on v_0. Let $\mathcal{CIR}$ be the circle with center at c and radius $\frac{d}{2}$ where d is the length between r_1 and r_2 along v_0. Let $\mathcal{L}_\perp(r)$ be the line through r and perpendicular to v_0. Let C_r be the point of intersection of $\mathcal{CIR}$ and $\mathcal{L}_\perp(r)$. We say that a robot r is on $\mathcal{CIR}$ if $0 \leq distance(r, C_r) < 1$ and r is on left of C_r on $\mathcal{L}_\perp(r)$.

In a $j-$ LSCC configuration we can divide the robots of color `moving1` in two classes namely, $\mathcal{IN}_j$ and $\mathcal{OUT}_j$. We say that a robot of color `moving1` in a $j-$LSCC is in $\mathcal{IN}_j$ if it is strictly inside the circle $\mathcal{CIR}$ otherwise, it is in $\mathcal{OUT}_j$ (i.e., when the robot is either strictly outside or on the circle $\mathcal{CIR}$).

Algorithm 13: offFormCircle(r)

```
1 if r sees at least one robot of color diameter then
2     if r sees exactly two robots on R_I(r) (resp. L_I(r)) with color diameter and r is
        nearest to the center c then
3         r.color = moving1;
4         move towards the robot of color diameter;
5     else if r sees r', a robot of color moving1 on R_I(r)(resp. L_I(r)) such that no
        robots between L_H(r) and L_H(r') then
6         r.color = moving1;
7         move horizontally towards the robot of color diameter;
```

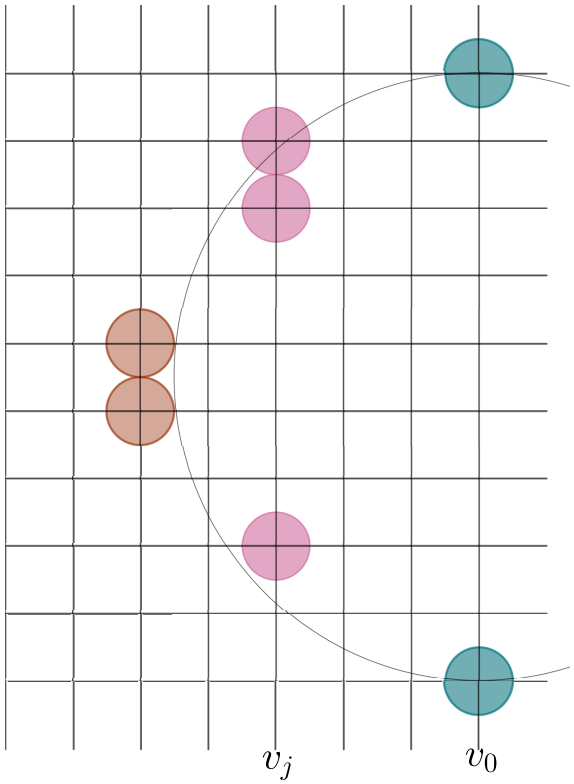


Figure 5.1: $j-$LSCC where on v_0 there are only two robots of color `diameter`, all robots of color `moving1` in the configuration are on v_j and all other robots are on left of v_j with color `done`.

Now a robot of color `moving1` can distinguish Phase 2 from Phase 1 from the fact that a robot of color `moving1` always sees at least one robot of color `diameter` not on its vertical line in Phase 2 (Lemma 5.4).

Let r be a robot of color `moving1` in Phase 2. If r on some v_j $(j > 1)$, sees both r_1 and r_2 with color `diameter` on $\mathcal{R}_I(r)$ and no robot of color `off` or `moving1` visible to r on $\mathcal{L}_I(r)$, also if r is nearest to c, then r moves left after changing its color to `done` only if r is strictly inside the circle $\mathcal{CIR}$. Otherwise, if r is on or strictly outside the circle then, it moves right (See Line 2-10 in Algorithm 14).

Now if $j = 1$ then r moves left after changing the color to `done` when r is strictly inside the circle, nearest to c and sees no robot of color `off` or `moving1` on $\mathcal{L}_I(r)$. Otherwise, if it is on the circle, it changes its color to `done` and terminates on

$\mathcal{L}_V(r) = v_1$ (See Line 11-18 in Algorithm 14). If r with color `moving1` sees another robot r' of color `moving1` on its right then it only moves right when it sees there is no other robot inside the rectangle bounded by the lines $\mathcal{L}_H(r)$, $\mathcal{L}_H(r')$, $\mathcal{L}_V(r)$ and $\mathcal{L}_V(r')$ (See Line 19-20 in Algorithm 14).

Algorithm 14: MOVING1FORMCIRCLE(r)

```
1  if r sees at least one robot of color diameter not on L_V(r) then
2      if R_I(r) (resp. L_I(r)) has exactly two robots of color diameter then
3          if R_I(r) (resp. L_I(r)) is more than one hop away from L_V(r) then
4              if r is nearest to c then
5                  if there is no robot of color off or moving1 on L_I(r) (resp. R_I(r))
                     then
6                      if r is strictly inside of CIR then
7                          r.color = done;
8                          move left (right resp.);
9                      else
10                         move right (left resp.);
11         else
12             if r is nearest to c then
13                 if there is no robot of color off or moving1 on L_I(r) (resp. R_I(r))
                     then
14                     if r is strictly inside of CIR then
15                         r.color = done;
16                         move left (right resp.);
17                     else
18                         r.color = done
19     else if R_I(r) (L_I(r) resp.) has a robot r' of color moving1 such that no robots on
         or strictly inside the rectangle bounded by L_H(r), L_H(r'), L_V(r) and L_V(r')
         except r and r' then
20         move right (left resp.);
```

We now have the following observations.

Observation 5.1. *In Phase 2, there can not be a configuration where two robots exist such that one is of color* `chord` *and another is color* `moving1`*.*

Observation 5.2. *In Phase 2, if a configuration has a robot r_c of color* `chord` *and a robot r_o of color* `off` *on the left (respectively right) of v_0 then the r_o must have to be on $H_L^C(r_c)$ (respectively $H_R^C(r_c)$).*

Observation 5.3. *In Phase 2, if a configuration has a robot r_m of color* `moving1`

and a robot r_o of color `off` *on the left (respectively right) of v_0 then the r_o must have to be on $H_L^C(r_m)$ (respectively $H_R^C(r_m)$).*

5.2.1.1 Correctness of Phase 2 (FormCircle)

To prove the correctness of Phase 2 we have to prove two things.

1. For a robot r on the left(resp. right) of the agreed diameter i.e., v_0, if r terminates, then it terminates on the grid point on the circumference of the agreed circle on left (resp. right) of v_0 which is on $\mathcal{L}_H(r)$ (Lemma 5.5)
2. Except r_1 and r_2 (the robots of color `diameter`), all other robots terminate in Phase 2. (Lemma 5.6).

For r_1 and r_2, they are already terminated on Phase 1 on the two endpoints of the agreed diameter, so they are also on the circle. Now to prove these two things we have some other lemmas that will be useful for the proof. In the following, we prove these lemmas along with the two above-mentioned results and summarize the main result in Theorem 5.1.

Lemma 5.1. *In Phase 2, a robot r with color* `chord` *always sees a robot of color* `diameter`*.*

Proof. Let r_1 and r_2 be the only two robots of color `diameter` on the vertical line v_0 at a time, say t in Phase 2. Let at time t, r be a robot of color `chord` on v_{i_1} on the left of v_0 that can not see both r_1 and r_2. Then there must exist two robots r_1' and r_2' strictly inside the rectangles bounded by v_{i_1}, $\mathcal{L}_H(r)$, $\mathcal{L}_H(r_1)$, v_0 and v_{i_1}, $\mathcal{L}_H(r)$, $\mathcal{L}_H(r_2)$, v_0 respectively (Figure 5.2). Note that r_1' and r_2' must have color `chord` (due to Observation 5.1 and Observation 5.2). Let $\mathcal{L}_V(r_1')$ is the vertical line v_{i_2} and $\mathcal{L}_V(r_2')$ is the vertical line v_{i_3} where $i_1 > i_2 \geq i_3 > 0$. We claim that $i_2 = i_3$. Otherwise, let at time t, $i_2 > i_3$. Now note that, r must have moved left from v_{i_2} before t as $i_1 > i_2$. Thus, there must exist a time $t_1 < t$ when r gets activated on v_{i_2} and sees it is terminal on v_{i_2} and sees r_1 and r_2 on $\mathcal{R}_I(r)$ and then moves left to v_{i_1}. Thus, at t_1, $\mathcal{R}_I(r) = v_0$. This implies r_2' must be on the

left or on v_{i_2} at time t_1. This is not possible as r'_2 is on v_{i_3} at time $t > t_1$ and a robot of color `chord` on the left of v_0 never moves right. Thus at time t, r'_1 and r'_2 must be on same vertical line say v_{i_2}. Now at t, r is on v_{i_1} which is on the left of v_{i_2}. This implies there exists a time before t when r moved left from v_{i_2}. Thus there exists a time $t_2 < t$ when r is activated on v_{i_2} and sees it is terminal on v_{i_2} and there is no robots between v_{i_2} and v_0. This implies at t_2, r'_1 and r'_2 must be on v_{i_2}. So at t_2, r can not be terminal on v_{i_2}. The contradiction arises because our assumption that r can not see both r_1 and r_2 is false. Hence, r must see at least one robot of color `diameter` in Phase 2.

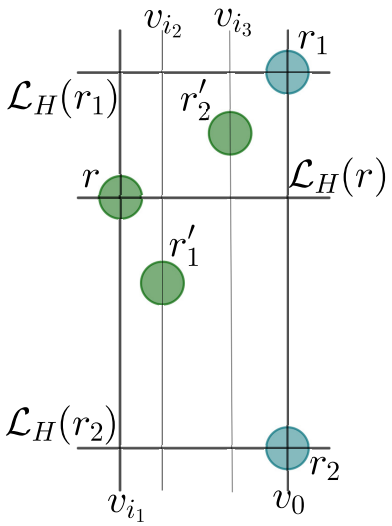


Figure 5.2: r_1 and r_2 is on v_0 with color `diameter`, r is on v_{i_1} with color `chord` which can not see r_1 and r_2 due to r'_1 on v_{i_2} and r'_2 on v_{i_3}.

□

Lemma 5.2. *In Phase 2, a robot of color* ***`off`*** *always sees a robot of color* ***`diameter`****.*

Proof. Let r_1 and r_2 be the only two robots of color `diameter` on the vertical line v_0 at a time, say t in Phase 2. Let r be a robot of color `off` in Phase 2 that does not see r_1 and r_2 at t. So at time t, r must be on $v_{\lceil \frac{d}{2} \rceil}$ and there must be two robots r'_1 and r'_2 strictly inside the rectangles bounded by $\mathcal{L}_H(r)$, $\mathcal{L}_H(r_1)$, $v_0, v_{\lceil \frac{d}{2} \rceil}$ and $\mathcal{L}_H(r)$, $\mathcal{L}_H(r_2)$, $v_0, v_{\lceil \frac{d}{2} \rceil}$ respectively (Figure 5.3). Now, r'_1

and r'_2 both can be of color either `chord` or of color `moving1` (by Observation 5.1 Observation 5.2, Observation 5.3). Now, if both r'_1 and r'_2 are of color `chord`, then we reach contradiction by arguing similarly as in Lemma 5.1. So, let us consider the case where both r'_1 and r'_2 are of color `moving1` at time t.

Let at t, r is on $v_{\lceil \frac{d}{2} \rceil}$. Also, r'_1 and r'_2 are on right of $v_{\lceil \frac{d}{2} \rceil}$. $\mathcal{L}_H(r)$ divides the grid in two halves. Let the half where r'_1 is located at time t be denoted as the upper half and the half where r'_2 is located at t be denoted as the lower half. Now there exists a time $t' < t$ such that all robots on the left of v_0 have color `off` and are on $v_{\lceil \frac{d}{2} \rceil}$. The first robot that moves right from $v_{\lceil \frac{d}{2} \rceil}$ after changing color to `moving1` must be nearest to c at time t'. Let r_{i_1} be such a robot. Note that r_{i_1} is not r. So without loss of generality let it be in the upper half at t'. Now there can be at most another robot, say r_{i_2}, which is also nearest to c at time t'. If this is the case then r_{i_2} must be in the upper half at time t' as otherwise, r becomes nearer to c than r_{i_1} and r_{i_2} contrary to the assumption. So, the first robot, say r_f, from lower half that moves right from $v_{\lceil \frac{d}{2} \rceil}$ must have moved after seeing a robot, say r'_f of color `moving1` on $\mathcal{R}_I(r_f)$ such that there is no other robots strictly between $\mathcal{L}_H(r_f)$ and $\mathcal{L}_H(r'_f)$ at some time t_1 where $t > t_1 > t'$. Note that r'_f must be on the upper half at t_1. So, at t_1, the region strictly between $\mathcal{L}_H(r_f)$ and $\mathcal{L}_H(r'_f)$ can not be empty as r is there. So, we arrive at a contradiction due to the wrong assumption that there exists a time t such that in the configuration at time t there exists a robot of color `off` that can not see both r_1 and r_2 of color `diameter`. Thus any robot of color `off` always sees at least one robot of color `diameter`.

□

Now, we have to prove that a robot of color `moving1` always sees at least one robot of color `diameter` in Phase 2 which is not on its own vertical line. This will ensure that a robot of color `moving1` distinguishes Phase 2 from Phase 1. This is because in Phase 1 even if a robot of color `moving1` sees a robot of color `diameter` it must be on its own vertical line. To prove this we have to prove the following lemma first.

Lemma 5.3. *Let at a time t, a configuration is a $j-$LSCC, where $j > 1$ and*

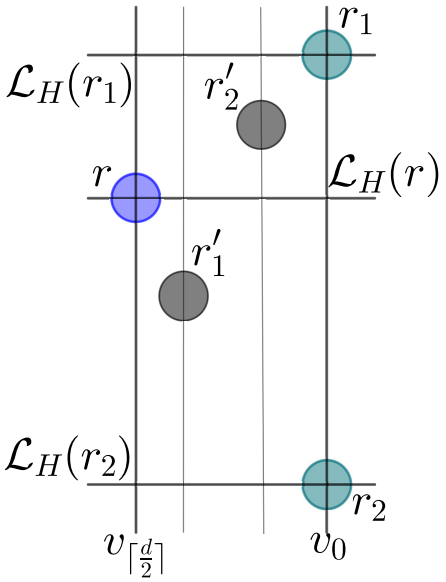


Figure 5.3: r_1 and r_2 is on v_0 with color `diameter`, r is on $v_{\lceil \frac{d}{2} \rceil}$ with color `off` which can not see r_1 and r_2 due to r'_1 and r'_2.

$\mathcal{OUT}_j \neq \phi$. Then,

1. *there exists a time $t_1 \geq t$ such that at time t_1, the configuration is again a $j-$LSCC where $\mathcal{IN}_j = \phi$ and $\mathcal{OUT}_j$ at time $t_1 = \mathcal{OUT}_j$ at time t.*

2. *Moreover, there is another time $t_2 > t_1$ such that at t_2, the configuration is a $(j-1)-$LSCC and $\mathcal{IN}_{j-1} \cup \mathcal{OUT}_{j-1}$ at time $t_2 = \mathcal{OUT}_j$ at time t_1.*

Proof. Let at time t the configuration is a $j-$LSCC where $j > 1$. In this case if $\mathcal{IN}_j = \phi$ then we have nothing to prove for the first part as $t_1 = t$. So, let at time t, $\mathcal{IN}_j \neq \phi$. Now to prove the first part we have to show that there exists a time $t' > t$ such that at time t' the configuration is again a $j-$LSCC configuration where $|\mathcal{IN}_j|$ at $t' < |\mathcal{IN}_j|$ at t and between $[t, t']$ no robot in $\mathcal{OUT}_j$ at time t moves even if it is activated. Let at t, $\mathcal{IN}_j = \{r_{i_1}, r_{i_2}, \ldots r_{i_p}\}$ where $p \geq 1$. Note that, distance between r_{i_k} and c is strictly less than distance between r and c for any $r \in \mathcal{OUT}_j$ and any $k \in \{1, 2 \ldots p\}$ at time t. So at t, a robot nearest to c must be from $\mathcal{IN}_j$. Let r_{i_n} be one such robot. Then upon activation, it moves

left after changing the color to `done`. Let t' be the first time instance such that for all $t_x \in [t, t'), r_{i_n}$ is on v_j in $\mathcal{C}(t_x)$ and in $\mathcal{C}(t')$, r_{i_n} is on v_{j+1} with color `done`. So, $|\mathcal{IN}_j|$ at $t' < |\mathcal{IN}_j|$ at t. Now we only have to show that any robot of color `moving1` in $\mathcal{OUT}_j$ at time t stays on v_j in the time interval $[t, t']$. If possible let some robots that were in $\mathcal{OUT}_j$ at time t move right, on or, before t'. Let r_o be the first such robot to move right. Then r_o must have been activated at some time $t'_1 < t'$ when it is nearest to c. But since $t'_1 < t'$, in $\mathcal{C}(t'_1), r_{i_n}$ was on v_j. Thus in $\mathcal{C}(t'_1)$, r_o can ot be nearest to c. So at t', the configuration is again a $j-$LSCC configuration where $|\mathcal{IN}_j|$ at $t' < |\mathcal{IN}_j|$ at t and between $[t, t']$ no robot in $\mathcal{OUT}_j$ at time t moves even if it is activated. So, $|\mathcal{OUT}_j|$ at time t' remains same to $|\mathcal{OUT}_j|$ at time t. This proves eventually there is a time t_1 when the configuration is a $j-$LSCC with $\mathcal{IN}_j = \phi$ and $\mathcal{OUT}_j$ at time $t_1 = \mathcal{OUT}_j$ at time t.

Now, for the second part, we have to prove that there exists a time $t_2 > t_1$ when all robots on v_j at t_1 are on v_{j-1} at time t_2. Thus we have to show that a robot after reaching v_{j-1} from v_j does not do anything until all robots of v_j at time t_1 reach v_{j-1}. If possible let a robot r after reaching v_{j-1} move again before all robots of v_j move. This implies there exists a time $t'_2 > t_1$ when there are robots on v_j with color `moving1` but r does not see any robot of color `moving1` on v_j from v_{j-1} at time t'_2. This implies there must exist at least one robot, say r', on v_j which does not have color `moving1` and which obstructs r from seeing any robot of color `moving1` on v_j at time t'_2 (Figure 5.4). Note that r' must be of color `done` and it must have moved from v_{j-1} to v_j after changing its color to `done` from `moving1`. This is because at t_1, $\mathcal{IN}_j = \phi$, so r' has not changed its color to `done` on v_j. So, at t'_2 there must be at least one robot of color `done` on v_j which has moved to v_j from v_{j-1} after t_1. Without loss of generality let r' be the first robot that moves to v_j from v_{j-1} after changing its color to `done` from `moving1`. Let r' is activated on v_{j-1} with color `moving1` at some time t'_4 where $t'_2 > t'_4 > t_1$. This implies r' also has not seen any robot on v_j of color `moving1` at time t'_4. Since $t'_2 > t'_4 > t_1$, there must exist a robot of color `moving1` on v_j at time t'_4. r' does not see that robot at t'_4 implies there must exists another robot, say r_s of color `done` on v_j at t'_4. Also, r_s must have moved to v_{j-1} to v_j for the

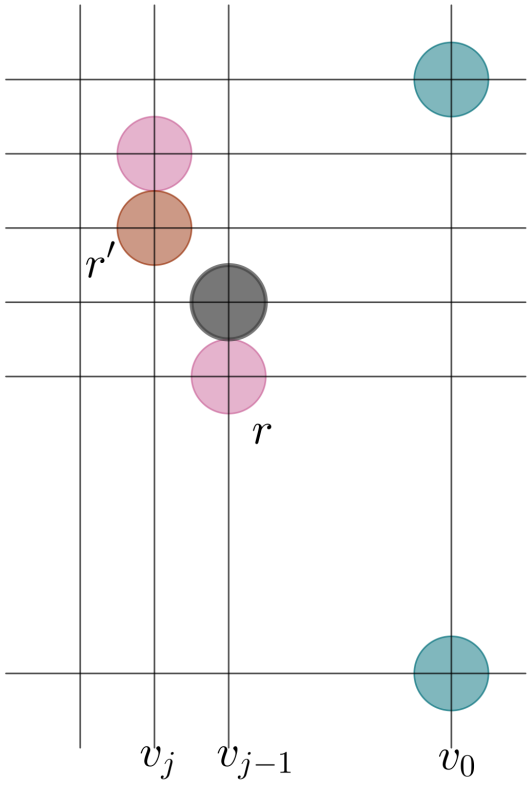


Figure 5.4: Configuration $\mathcal{C}(t_2')$. Here r on v_{j-1} has no visible robot of color `moving1` on v_j in spite of v_j having such a robot. r' actually obstructs the view of r.

same reason described above. So r' can not be the first robot that moved to v_j from v_{j-1} after changing its color to `done` from `moving1` after t_1. So no robot that reached v_{j-1} from v_j after t_1, moves until all robots of v_j at time t_1 moves to v_{j-1}. Let t_2 be the time when the last robot of v_j reaches v_{j-1} after t_1. Note that all robots of color `moving1` at time t_2 are on v_{j-1} and they are the only ones on v_{j-1}. Also, there are no robots between v_{j-1} and v_0 and all robots on the left of v_{j-1} have color `done`. So this configuration at time t_2 is a $(j-1)-$LSCC. Also, all robots that were on v_j at time t_1 are now on v_{j-1} at time t_2 and no other robots moved onto v_{j-1}. So, the set of all robots on v_{j-1} at time $t_2 = \mathcal{IN}_{j-1} \cup \mathcal{OUT}_{j-1}$ at time $t_2 = \mathcal{OUT}_j$ at time t_1.

□

We have seen in the description that, in Phase 2 the configuration becomes a $(\lceil \frac{d}{2} \rceil - 1)-$LSCC configuration. Now this lemma tells that for all $j > 1$ and $j \leq \lceil \frac{d}{2} \rceil - 1$ there is a time when the configuration becomes a $(j-1)-$ SLCC from $j-$SLCC. So starting from $(\lceil \frac{d}{2} \rceil - 1)-$LSCC we will eventually have $(\lceil \frac{d}{2} \rceil - 2)-$LSCC then $(\lceil \frac{d}{2} \rceil - 3)-$LSCC and so on until we have a 1-LSCC. Also note that from $i-$LSCC, $j-$LSCC can not be formed if $j > i$ as robots of color `moving1` that are on the left of v_0 only moves right in Phase 2. Moreover, we can have the following corollary.

Corollary 5.0.1. *After $(\lceil \frac{d}{2} \rceil - 1)$-LSCC is formed, for all $j \in \{1, 2, \ldots \lceil \frac{d}{2} \rceil - 2\}$ and for any $i < j$, a configuration can not have robots on v_i until j-LSCC is formed.*

Proof. Let us fix a $j \in \{1, 2, \ldots \lceil \frac{d}{2} \rceil - 2\}$. Let at time t which is after $(\lceil \frac{d}{2} \rceil - 1)$-LSCC is formed, the configuration is either a $p-$LSCC or, t is between t_1 and t_2 such that at t_1 the configuration is a $p-$LSCC and at t_2 it becomes $(p-1)-$LSCC where $j < p \leq \lceil \frac{d}{2} \rceil - 1$. Now by Lemma 5.3, at time t there can be no robots on v_s where $s < p - 1$. Now for any $i < j$, we have $i < j < p \implies i < j \leq p - 1$. Hence at t, there can be no robots on v_i where $i < j$. □

Now we prove the following lemma that ensures that a robot of color always sees at least one robot of color `diameter` not on its own vertical line in Phase 2.

Lemma 5.4. *In Phase 2, a robot r of color `moving1` can always see a robot of color `diameter` that is not on $\mathcal{L}_V(r)$.*

Proof. Let t be a time when a robot, say r, of color `moving1` on v_j can not see both of r_1 and r_2, the two robots of color `diameter` on v_0. Without loss of generality let r be on the left of v_0. This implies at time t, there must exist two robots r_1' and r_2' strictly inside the rectangles bounded by $\mathcal{L}_H(r), \mathcal{L}_H(r_1), v_0, v_j$ and $\mathcal{L}_H(r), \mathcal{L}_H(r_2), v_0, v_j$ respectively (Figure 5.5) i.e., r_1' and r_2' are on v_{i_1} and v_{i_2} where $j > i_1 \geq i_2$. Also, $j > 1$ and $j < \lceil \frac{d}{2} \rceil$ as even if a robot can have color `moving1` on $v_{\lceil \frac{d}{2} \rceil}$ it performs first look phase as a robot of color `moving1` after it moves right. Also from the description $(\lceil \frac{d}{2} \rceil - 1)-$LSCC is formed first

until then no robot moves to v_i where $i < \lceil \frac{d}{2} \rceil - 1$. So until $(\lceil \frac{d}{2} \rceil - 1)-$LSCC is formed all robots of color `moving1` must see both the robots r_1 and r_2 from $v_{\lceil \frac{d}{2} \rceil - 1}$. So t must be a time after $(\lceil \frac{d}{2} \rceil - 1)-$LSCC is formed. Observe that, r'_1 and r'_2 must be of color `moving1`. Now since at time t, there are robots on v_{i_1} and v_{i_2} where $i_1, i_2 < j$, there exists a time $t' < t$ when the configuration was a $j-$LSCC such that $\mathcal{IN}_j = \phi$ (Lemma 5.3 and Corollary 5.0.1) (Figure 5.5). Also in $\mathcal{C}(t)'$, r, r'_1, r'_2 were on v_j. Now similar to lemma 5.2, let us denote the half where r'_1 is located as the upper half and the other one as the lower half. Now the first robot that moves to v_{j-1} from v_j must be nearest to c at time t'. Without loss of generality let it be on the upper half at time t'. There can be at most another such robot which is also nearest to c at time t'. If another such robot exists then it also has to be on the upper half at time t' otherwise r becomes nearer to c at time t and moves to v_{j-1} before any other robots and thus at t r can not be at v_j as assumed. Also at time t, there is at least one robot of color `moving1` on the lower half. So let r_l be the first robot from the lower half at time t' that has moved to v_{j-1} from v_j. Then it must have moved from v_j after seeing a robot, say r'_l, of color `moving1` on $\mathcal{R}_I(r_l) = v_{j-1}$ such that there is no other robots except r_l and r'_l on or between the rectangle, say R, bounded by $v_j, v_{j-1}, \mathcal{L}_H(r_l)$ and $\mathcal{L}_H(r'_l)$ at a time $t'_1 > t'$ and $t > t'_1$. Now since r_l is the first robot from the lower half to move after t', r'_l must be from the upper half. Thus at time t'_1, R must contain r other than r_l and r'_l which is a contradiction. Hence, a robot of color `moving1` always sees a robot of color `diameter`. According to the algorithm, since r never reaches v_0 in Phase 2, it can not see r_1 and r_2 on its own vertical line. □

Let r be a robot in Phase 2 on the left of v_0. Note that $\mathcal{L}_H(r)$ intersects the circle $\mathcal{CIR}$ exactly once at a point C_r on the left of v_0. Thus, there exists exactly one grid point denoted as, $C_T(r)$ on $\mathcal{L}_H(r)$ such that either $dist(C_r, C_T(r)) = 0$ or $C_T(r)$ is on the left of C_r such that $dist(C_r, C_T(r)) < 1$. Then we define $C_T(r)$ to be the *Terminating Point* of r. We can now have the following observation

Observation 5.4. *In a 1-LSCC if a robot $r \in \mathcal{OUT}_1$, then r must be on $C_T(r)$.*

For r, $C_T(r) = \mathcal{L}_H(r) \cap v_j$ for some $j \geq 1$. Now, in a 1-LSCC if $r \in \mathcal{OUT}_1$, then

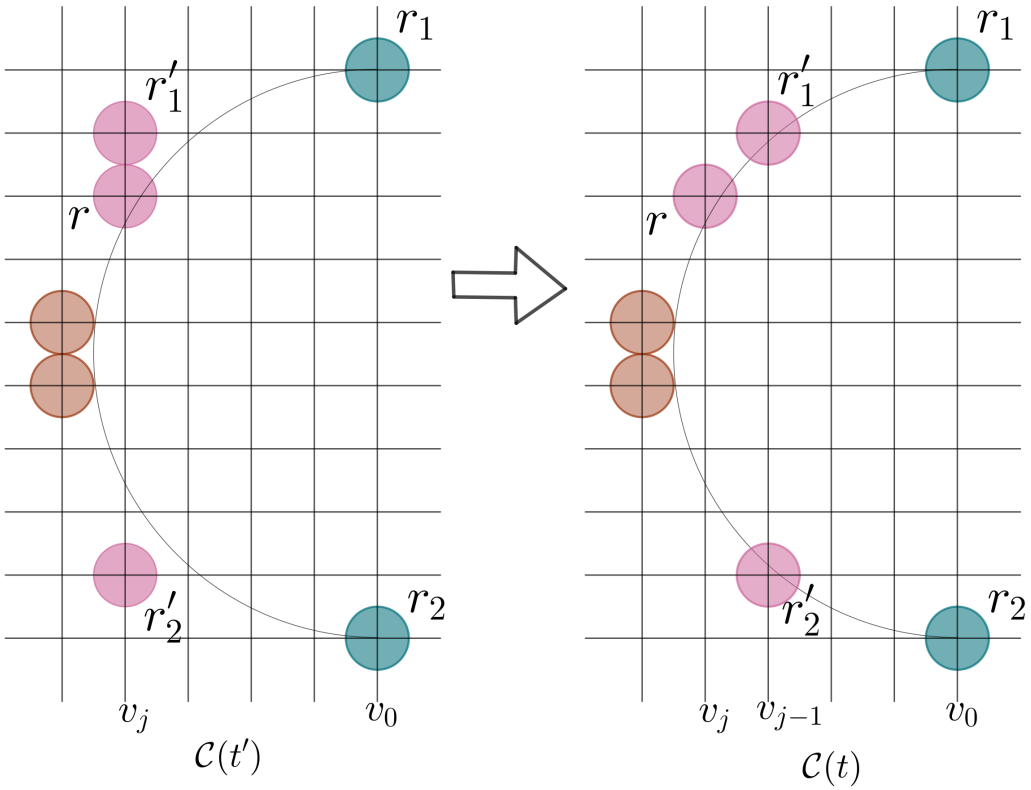


Figure 5.5: $\mathcal{C}(t')$ is a $j-LSCC$ where all robots on v_j are in $\mathcal{OUT}_j$. From $\mathcal{C}(t')$, $\mathcal{C}(t)$ is formed where r is still on v_j but r'_1 and r'_2 are on right of v_j obstructing r from seeing r_1 and r_2. Here r_1 and r_2 are the robots of color `diameter` on v_0.

$j \leq 1$. Hence, $j = 1$.

We now first ensure that if a robot r terminates, it does not terminate on a grid point that is not on $C_T(r)$.

Lemma 5.5. *A robot r can only terminate on $C_T(r)$.*

Proof. For this, first note that a robot can only terminate after $(\lceil \frac{d}{2} \rceil - 1)-$LSCC is formed. Now let for a robot r, $C_T(r)$ is the grid point $\mathcal{L}_H(r) \cap v_j$ for some $j \in [1, \lceil \frac{d}{2} \rceil] \cap \mathbb{N}$. If $j = \lceil \frac{d}{2} \rceil$ then in $(\lceil \frac{d}{2} \rceil - 1)-$LSCC, $r \in \mathcal{IN}_{\lceil \frac{d}{2} \rceil - 1}$. Then eventually r moves to $v_{\lceil \frac{d}{2} \rceil}$ after changing the color to `done`. When it is activated next it terminates on $C_T(r) = v_{\lceil \frac{d}{2} \rceil} \cap \mathcal{L}_H(r)$. So if $j = \lceil \frac{d}{2} \rceil$ then r can not terminate on any other position except $C_T(r)$.Now let us consider the case $j < \lceil \frac{d}{2} \rceil$. If

possible let r terminates on $\mathcal{L}_H(r) \cap v_i$ where $i > j \geq 1$. Then there must exist a time when r is on v_{i-1}. Then by Corollary 5.0.1 there exists a time when the configuration is $i-$LSCC. Here $r \in \mathcal{OUT}_i$ and it is strictly outside the circle $\mathcal{CIR}$ as $i > j$. So, again by lemma 5.3 there exists a time when the configuration becomes a $(i-1)-$LSCC and r is on v_{i-1}. Here $i-1 \geq j$ i.e $r \in \mathcal{OUT}_{i-1}$. Now if $i-1 = 1$ then r eventually changes the color to `done` and terminates on $v_j = v_1$ as $C_T(r) = \mathcal{L}_H(r) \cap v_1$ for this case (Observation 5.4). This is contrary to our assumption that r terminates on v_i where $i > j = 1$. So, let $i - 1 > 1$. Then eventually $(i-2)-$LSCC will be formed and r will be on v_{i-2}. Now note that according to the algorithm for Phase 2, after $(\lceil \frac{d}{2} \rceil - 1)-$LSCC is formed, only a robot of color `moving1` can move further from v_0 after changing the color to `done`. So even if a robot moves further from v_0 it can move in such a way only once as after that it terminates. So if r with color `moving1` is on v_{i-2} at some time in Phase 2, it can not move back to v_i and terminate. Thus we reach a contradiction assuming r terminates on v_i where $i > j$ where $C_T(r) = \mathcal{L}_H(r) \cap v_j$ for some $j \in [1, \lceil \frac{d}{2} \rceil] \cap \mathbb{N}$. Thus r must terminate either on $C_T(r) = \mathcal{L}_H(r) \cap v_j$ or on $\mathcal{L}_H(r) \cap v_i$ where $i < j$. If possible let r terminates on $\mathcal{L}_H(r) \cap v_i$ where $i < j$. This implies there exists a time t when the configuration is a $j-$LSCC. In this configuration r is on $C_T(r)$ thus $r \in \mathcal{OUT}_j$. Thus eventually $(j-1)-$LSCC will be formed where r is on v_{j-1}. Note that in this configuration $r \in \mathcal{IN}_{j-1}$. So, eventually, r will change its color to `done` and move to v_j. So r terminates on $\mathcal{L}_H(r) \cap v_j = C_T(r)$ contrary to our assumption. Hence if r terminates it must terminate at $C_T(r)$. □

Now we will prove that all robots that are not on v_0 terminate

Lemma 5.6. *All robots that are not on v_0 terminate eventually during Phase 2.*

Proof. If possible let r be a robot on the left of v_0 that never terminates in Phase 2. Let $C_T(r) = \mathcal{L}_H(r) \cap v_j$ for some $j \in \{1, 2, \ldots \lceil \frac{d}{2} \rceil\}$. For $j > 1$, r can terminate after it moves to $C_T(r)$ from v_{j-1} of a $(j-1)-$LSCC after changing the color to `done`. Now as it is assumed that r does not terminate, either $(j-1)-$LSCC is never formed or, even if it is formed r is not on v_{j-1} in $(j-1)-$LSCC. Now by

Lemma 5.3 if there is at least one robot that has not terminated $(j-1)-$LSCC will be formed eventually. Thus if r never terminates there can be only one possibility that when $(j-1)-$LSCC is formed r is not there on v_{j-1}. This implies r must be on some v_i where $i > j$ with color `done`. This implies r terminates at $v_i \cap \mathcal{L}_H(r)$ for some $i > j$. This is impossible due to Lemma 5.5. Hence r must terminate if $j > 1$. Now similarly, for $j = 1$, r never terminates implies, When $1-$LSCC is formed, r is not on v_1. Again this is impossible due to similar reasons as above. Thus all robots that are not on v_0 must terminate in Phase 2 □

Now using Lemma 5.5 and Lemma 5.6 and the fact that the robots of color `diameter` terminate after changing its color to `diameter` either from `chord` or from `moving1` we can state the following theorem.

Theorem 5.1. *From any initial configuration within finite time, all opaque fat robots with one axis agreement can terminate after forming a circle on an infinite grid under asynchronous scheduler by executing the algorithm $CIRCLE\_FG$.*

5.3 Concluding Remarks

Circle formation is a fundamental problem in swarm robotics that has been extensively studied in the Euclidean plane. On grid-based terrains, it has been explored under the obstructed visibility model, where robots are considered dimensionless. However, research on circle formation involving dimensioned agents has largely remained close to the continuous plane. This work bridges that gap by addressing the circle formation problem on an infinite rectangular grid, starting from any arbitrary configuration. Here, the solution is provided assuming the robots have one-axis agreement and a light with five colors, operating under an asynchronous scheduler.

As a direction for future research, this study needs to investigate the necessity of the one-axis agreement and explore the potential for reducing the number of colors required for the same problem if possible at all. Additionally, it would be really interesting to study the minimization of the circle's approximation errors in

this setting. Specifically, for any given number of robots, a key question remains whether they can form a perfect circle on the grid, with all robots terminating on distinct grid points that lie on a unique circle i.e., zero error circle formation on a grid.

Chapter 6

Conclusion

This chapter concludes the thesis by summarizing all the technical results presented throughout the work. Furthermore, several promising directions for future exploration, building upon the findings of this thesis, have been highlighted in this chapter.

In the first two chapters we have discussed about the point formation problem also known as the gathering problem. In third and fourth chapter we study the geometric pattern formation problem on discrete domain under fat, opaque robot mode.

In Chapter 2, we investigated the gathering problem on a discrete structure known as the "infinite triangular grid." The primary objective was to incorporate a realistic limited visibility model for the robot swarm. Unlike previous works on the triangular grid, the introduction of the limited visibility model enhances the practical applicability of this study. We have considered here the one hop vision model. An extra benefit of assuming the one hop vision model is that any algorithm that works in one hop vision model also works for obstructed visibility model in discrete domain. On this assumption, we have shown that the robots must have one axis agreement to solve the gathering problem on the infinite triangular grid. We have also proved the sufficiency of one axis agreement by providing an algorithm called 1-Hop 1-Axis Gather under a semi-synchronous scheduler in the same chapter. Additionally, we proved that the lower bound on

the time required to gather n robots is $O(n)$ and confirmed that the proposed algorithm meets this bound, making it time-optimal.

Open Problem 1. *In collision free gathering robots can only collide at the point they are gathering. So, Find out the minimum visibility range to design an collision free gathering algorithm on an infinite triangular grid under different scheduler.*

Open Problem 2. *Find out the class of configurations for which the gathering can be solved even without axis agreement.*

Open Problem 3. *Study the gathering with myopic robot problem on different regular tessellations, It would be interesting to study this problem under an infinite hexagonal grid.*

In Chapter 3, we studied the Gathering problem on a finite grid. We assumed here the gathering vertex is always visible to the robots but it can change its position to any adjacent node in a round. We call this gathering vertex a "resource". It is assumed that there is a door at a corner of the grid and the robots can enter the grid one by one. We also assumed that the resource doesn't change its position if it coincides with the location of at least one robot. This problem is similar to a cops and robbers problem on grid but with a subtle difference that here in a round both robber (in our case the resource) and the cops (in our case the robots) can all move simultaneously in one single round.

In this chapter, we are interested on finding out the minimum number of robots necessary to gather at the resource. The robots are oblivious and silent, they do not have any axis agreement, but can see the whole grid. The robots can't distinguish the door vertex unless they are on it. The resource can't stay at a single vertex alone for more than T_f number of activations.

Under these assumptions, we have shown that for a fully synchronous scheduler two robots are necessary and sufficient to solve this problem and for any other scheduler at least 3 robots are necessary.

Open Problem 4. *For a grid of size $m \times n$, with one corner having a door vertex and a resource position chosen arbitrarily except the door vertex, find the min-*

imum numbers of robots necessary and sufficient to solve the gathering problem on the resource under semi-synchronous and asynchronous scheduler.

In Chapter 4, we have discussed about the line formation problem and in the Chapter 5, we have discussed the circle formation problem on infinite grid. We have shown in Chapter 4 that assuming one axis agreement, from any arbitrary configuration luminous, opaque fat robots having 3 colors can solve the line formation problem on an infinite grid . Further using an algorithm from Chapter 4 as a subroutine, in Chapter 5 we solved the circle formation problem on infinite rectangular grid for luminous, opaque, fat robots having 5 colors with one axis agreement and for any arbitrary initial configuration.

Open Problem 5. *Is 3 colors and one axis agreement necessary for the line formation algorithm for opaque and fat robots on an infinite rectangular grid from any arbitrary initial configuration?*

Open Problem 6. *Is 5 colors and one axis agreement necessary for the line formation algorithm for opaque and fat robots on an infinite rectangular grid from any arbitrary initial configuration?*

Open Problem 7. *Is there an algorithm that solves the circle formation problem where the formed circle is an exact circle and all robots are on the circle and as well as on the grid points?*

Bibliography

[1] Ranendu Adhikary, Kaustav Bose, Manash Kumar Kundu, and Buddhadeb Sau. Mutual visibility by asynchronous robots on infinite grid. In Seth Gilbert, Danny Hughes, and Bhaskar Krishnamachari, editors, *Algorithms for Sensor Systems - 14th International Symposium on Algorithms and Experiments for Wireless Sensor Networks, ALGOSENSORS 2018, Helsinki, Finland, August 23-24, 2018, Revised Selected Papers*, volume 11410 of *Lecture Notes in Computer Science*, pages 83–101. Springer, 2018. doi:10.1007/978-3-030-14094-6_6.

[2] Ranendu Adhikary, Manash Kumar Kundu, and Buddhadeb Sau. Circle formation by asynchronous opaque robots on infinite grid. *Computer Science*, 22(1), Feb. 2021. URL: https://journals.agh.edu.pl/csci/article/view/3840, doi:10.7494/csci.2021.22.1.3840.

[3] Chrysovalandis Agathangelou, Chryssis Georgiou, and Marios Mavronicolas. A distributed algorithm for gathering many fat mobile robots in the plane. In Panagiota Fatourou and Gadi Taubenfeld, editors, *ACM Symposium on Principles of Distributed Computing, PODC '13, Montreal, QC, Canada, July 22-24, 2013*, pages 250–259. ACM, 2013. doi:10.1145/2484239.2484266.

[4] Noa Agmon, Sarit Kraus, and Gal A. Kaminka. Multi-robot perimeter patrol in adversarial settings. In *2008 IEEE International Conference on Robotics and Automation, ICRA 2008, May 19-23, 2008, Pasadena, California, USA*, pages 2339–2345. IEEE, 2008. doi:10.1109/ROBOT.2008.4543563.

[5] Noa Agmon and David Peleg. Fault-tolerant gathering algorithms for autonomous mobile robots. *SIAM J. Comput.*, 36(1):56–82, 2006. doi:10.1137/050645221.

[6] Aisha Aljohani and Gokarna Sharma. Complete visibility for mobile robots with lights tolerating faults. *Int. J. Netw. Comput.*, 8(1):32–52, 2018. URL: http://www.ijnc.org/index.php/ijnc/article/view/166.

[7] Ricardo A. Baeza-Yates and René Schott. Parallel searching in the plane. *Comput. Geom.*, 5:143–154, 1995. doi:10.1016/0925-7721(95)00003-R.

[8] Ralph Beckers, Owen E Holland, and Jean-Louis Deneubourg. Fom local actions to global tasks: Stigmergy and collective robotics. In *Prerational Intelligence: Adaptive Behavior and Intelligent Systems Without Symbols and Logic, Volume 1, Volume 2 Prerational Intelligence: Interdisciplinary Perspectives on the Behavior of Natural and Artificial Systems, Volume 3*, pages 1008–1022. Springer, 2000.

[9] Gerardo Beni. Coherent swarm motion under distributed control. In *Proc. Int. Symp. on Distributed Autonomous Robotic System, Wako,*, pages 39–52, 1992.

[10] Subhash Bhagat, Sruti Gan Chaudhuri, and Krishnendu Mukhopadhyaya. Fault-tolerant gathering of asynchronous oblivious mobile robots under one-axis agreement. *J. Discrete Algorithms*, 36:50–62, 2016. URL: https://doi.org/10.1016/j.jda.2015.10.005, doi:10.1016/J.JDA.2015.10.005.

[11] Subhash Bhagat and Krishnendu Mukhopadhyaya. Optimum algorithm for mutual visibility among asynchronous robots with lights. In Paul G. Spirakis and Philippas Tsigas, editors, *Stabilization, Safety, and Security of Distributed Systems - 19th International Symposium, SSS 2017, Boston, MA, USA, November 5-8, 2017, Proceedings*, volume 10616 of *Lecture Notes in Computer Science*, pages 341–355. Springer, 2017. doi:10.1007/978-3-319-69084-1_24.

[12] Kaustav Bose, Ranendu Adhikary, Manash Kumar Kundu, and Buddhadeb Sau. Arbitrary pattern formation by opaque fat robots with lights. In Manoj Changat and Sandip Das, editors, *Algorithms and Discrete Applied Mathematics - 6th International Conference, CALDAM 2020, Hyderabad, India, February 13-15, 2020, Proceedings*, volume 12016 of *Lecture Notes in Computer Science*, pages 347–359. Springer, 2020. doi:10.1007/978-3-030-39219-2_28.

[13] Kaustav Bose, Ranendu Adhikary, Manash Kumar Kundu, and Buddhadeb Sau. Arbitrary pattern formation on infinite grid by asynchronous oblivious robots. *Theor. Comput. Sci.*, 815:213–227, 2020. URL: https://doi.org/10.1016/j.tcs.2020.02.016, doi:10.1016/J.TCS.2020.02.016.

[14] Kaustav Bose, Manash Kumar Kundu, Ranendu Adhikary, and Buddhadeb Sau. Arbitrary pattern formation by asynchronous opaque robots with lights. In Keren Censor-Hillel and Michele Flammini, editors, *Structural Information and Communication Complexity - 26th International Colloquium, SIROCCO 2019, L'Aquila, Italy, July 1-4, 2019, Proceedings*, volume 11639 of *Lecture Notes in Computer Science*, pages 109–123. Springer, 2019. doi:10.1007/978-3-030-24922-9_8.

[15] Zohir Bouzid, Shantanu Das, and Sébastien Tixeuil. Gathering of mobile robots tolerating multiple crash faults. In *IEEE 33rd International Conference on Distributed Computing Systems, ICDCS 2013, 8-11 July, 2013, Philadelphia, Pennsylvania, USA*, pages 337–346. IEEE Computer Society, 2013. doi:10.1109/ICDCS.2013.27.

[16] Zohir Bouzid, Maria Gradinariu Potop-Butucaru, and Sébastien Tixeuil. Optimal byzantine-resilient convergence in uni-dimensional robot networks. *Theor. Comput. Sci.*, 411(34-36):3154–3168, 2010. URL: https://doi.org/10.1016/j.tcs.2010.05.006, doi:10.1016/J.TCS.2010.05.006.

[17] Quentin Bramas and Sébastien Tixeuil. Brief announcement: Probabilistic asynchronous arbitrary pattern formation. In George Giakkoupis, editor,

Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, Chicago, IL, USA, July 25-28, 2016, pages 443–445. ACM, 2016. doi:10.1145/2933057.2933074.

[18] Sebastian Brandt, Felix Laufenberg, Yuezhou Lv, David Stolz, and Roger Wattenhofer. Collaboration without communication: Evacuating two robots from a disk. In Dimitris Fotakis, Aris Pagourtzis, and Vangelis Th. Paschos, editors, *Algorithms and Complexity - 10th International Conference, CIAC 2017, Athens, Greece, May 24-26, 2017, Proceedings*, volume 10236 of *Lecture Notes in Computer Science*, pages 104–115, 2017. doi:10.1007/978-3-319-57586-5_10.

[19] Davide Canepa, Xavier Défago, Taisuke Izumi, and Maria Potop-Butucaru. Flocking with oblivious robots. In Borzoo Bonakdarpour and Franck Petit, editors, *Stabilization, Safety, and Security of Distributed Systems - 18th International Symposium, SSS 2016, Lyon, France, November 7-10, 2016, Proceedings*, volume 10083 of *Lecture Notes in Computer Science*, pages 94–108, 2016. doi:10.1007/978-3-319-49259-9_8.

[20] Davide Canepa and Maria Gradinariu Potop-Butucaru. Stabilizing flocking via leader election in robot networks. In Toshimitsu Masuzawa and Sébastien Tixeuil, editors, *Stabilization, Safety, and Security of Distributed Systems, 9th International Symposium, SSS 2007, Paris, France, November 14-16, 2007, Proceedings*, volume 4838 of *Lecture Notes in Computer Science*, pages 52–66. Springer, 2007. doi:10.1007/978-3-540-76627-8_7.

[21] Jannik Castenow, Matthias Fischer, Jonas Harbig, Daniel Jung, and Friedhelm Meyer auf der Heide. Gathering anonymous, oblivious robots on a grid. *Theor. Comput. Sci.*, 815:289–309, 2020. URL: https://doi.org/10.1016/j.tcs.2020.02.018, doi:10.1016/J.TCS.2020.02.018.

[22] Sruti Gan Chaudhuri and Krishnendu Mukhopadhyaya. Leader election and gathering for asynchronous fat robots without common chirality. *J. Discrete Algorithms*, 33:171–192, 2015. URL: https://doi.org/10.1016/j.jda.2015.04.001, doi:10.1016/J.JDA.2015.04.001.

[23] Huda Chuangpishit, Jurek Czyzowicz, Leszek Gasieniec, Konstantinos Georgiou, Tomasz Jurdzinski, and Evangelos Kranakis. Patrolling a path connecting a set of points with unbalanced frequencies of visits. In A Min Tjoa, Ladjel Bellatreche, Stefan Biffl, Jan van Leeuwen, and Jirí Wiedermann, editors, *SOFSEM 2018: Theory and Practice of Computer Science - 44th International Conference on Current Trends in Theory and Practice of Computer Science, Krems, Austria, January 29 - February 2, 2018, Proceedings*, volume 10706 of *Lecture Notes in Computer Science*, pages 367–380. Springer, 2018. doi:10.1007/978-3-319-73117-9_26.

[24] Serafino Cicerone, Alessia Di Fonso, Gabriele Di Stefano, and Alfredo Navarra. Arbitrary pattern formation on infinite regular tessellation graphs. *Theor. Comput. Sci.*, 942:1–20, 2023. URL: https://doi.org/10.1016/j.tcs.2022.11.021, doi:10.1016/J.TCS.2022.11.021.

[25] Serafino Cicerone, Gabriele Di Stefano, and Alfredo Navarra. Asynchronous arbitrary pattern formation: the effects of a rigorous approach. *Distributed Comput.*, 32(2):91–132, 2019. URL: https://doi.org/10.1007/s00446-018-0325-7, doi:10.1007/S00446-018-0325-7.

[26] Mark Cieliebak. Gathering non-oblivious mobile robots. In Martin Farach-Colton, editor, *LATIN 2004: Theoretical Informatics, 6th Latin American Symposium, Buenos Aires, Argentina, April 5-8, 2004, Proceedings*, volume 2976 of *Lecture Notes in Computer Science*, pages 577–588. Springer, 2004. doi:10.1007/978-3-540-24698-5_60.

[27] Mark Cieliebak, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Distributed computing by mobile robots: Gathering. *SIAM J. Comput.*, 41(4):829–879, 2012. doi:10.1137/100796534.

[28] Reuven Cohen and David Peleg. Convergence properties of the gravitational algorithm in asynchronous robot systems. *SIAM J. Comput.*, 34(6):1516–1528, 2005. doi:10.1137/S0097539704446475.

[29] Reuven Cohen and David Peleg. Convergence of autonomous mobile robots with inaccurate sensors and movements. *SIAM J. Comput.*, 38(1):276–302, 2008. doi:10.1137/060665257.

[30] Andrew Collins, Jurek Czyzowicz, Leszek Gasieniec, Adrian Kosowski, Evangelos Kranakis, Danny Krizanc, Russell Martin, and Oscar Morales-Ponce. Optimal patrolling of fragmented boundaries. In Guy E. Blelloch and Berthold Vöcking, editors, *25th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '13, Montreal, QC, Canada - July 23 - 25, 2013*, pages 241–250. ACM, 2013. doi:10.1145/2486159.2486176.

[31] Jurek Czyzowicz, Leszek Gasieniec, Thomas Gorry, Evangelos Kranakis, Russell Martin, and Dominik Pajak. Evacuating robots via unknown exit in a disk. In Fabian Kuhn, editor, *Distributed Computing - 28th International Symposium, DISC 2014, Austin, TX, USA, October 12-15, 2014. Proceedings*, volume 8784 of *Lecture Notes in Computer Science*, pages 122–136. Springer, 2014. doi:10.1007/978-3-662-45174-8_9.

[32] Jurek Czyzowicz, Leszek Gasieniec, Adrian Kosowski, and Evangelos Kranakis. Boundary patrolling by mobile agents with distinct maximal speeds. In Camil Demetrescu and Magnús M. Halldórsson, editors, *Algorithms - ESA 2011 - 19th Annual European Symposium, Saarbrücken, Germany, September 5-9, 2011. Proceedings*, volume 6942 of *Lecture Notes in Computer Science*, pages 701–712. Springer, 2011. doi:10.1007/978-3-642-23719-5_59.

[33] Jurek Czyzowicz, Leszek Gasieniec, and Andrzej Pelc. Gathering few fat mobile robots in the plane. *Theor. Comput. Sci.*, 410(6-7):481–499, 2009. URL: https://doi.org/10.1016/j.tcs.2008.10.005, doi:10.1016/J.TCS.2008.10.005.

[34] Jurek Czyzowicz, Konstantinos Georgiou, Ryan Killick, Evangelos Kranakis, Danny Krizanc, Lata Narayanan, Jaroslav Opatrny, and Sunil M. Shende. Priority evacuation from a disk using mobile robots - (extended

abstract). In Zvi Lotker and Boaz Patt-Shamir, editors, *Structural Information and Communication Complexity - 25th International Colloquium, SIROCCO 2018, Ma'ale HaHamisha, Israel, June 18-21, 2018, Revised Selected Papers*, volume 11085 of *Lecture Notes in Computer Science*, pages 392–407. Springer, 2018. doi:10.1007/978-3-030-01325-7_32.

[35] Jurek Czyzowicz, Konstantinos Georgiou, Evangelos Kranakis, Danny Krizanc, Lata Narayanan, Jaroslav Opatrny, and Sunil M. Shende. Search on a line by byzantine robots. *Int. J. Found. Comput. Sci.*, 32(4):369–387, 2021. doi:10.1142/S0129054121500209.

[36] Jurek Czyzowicz, Konstantinos Georgiou, Evangelos Kranakis, Lata Narayanan, Jaroslav Opatrny, and Birgit Vogtenhuber. Evacuating robots from a disk using face-to-face communication (extended abstract). In Vangelis Th. Paschos and Peter Widmayer, editors, *Algorithms and Complexity - 9th International Conference, CIAC 2015, Paris, France, May 20-22, 2015. Proceedings*, volume 9079 of *Lecture Notes in Computer Science*, pages 140–152. Springer, 2015. doi:10.1007/978-3-319-18173-8_10.

[37] Jurek Czyzowicz, Evangelos Kranakis, Danny Krizanc, Lata Narayanan, and Jaroslav Opatrny. Search on a line with faulty robots. *Distributed Comput.*, 32(6):493–504, 2019. URL: https://doi.org/10.1007/s00446-017-0296-0, doi:10.1007/S00446-017-0296-0.

[38] Gianlorenzo D'Angelo, Alfredo Navarra, and Nicolas Nisse. A unified approach for gathering and exclusive searching on rings under weak assumptions. *Distributed Comput.*, 30(1):17–48, 2017. URL: https://doi.org/10.1007/s00446-016-0274-y, doi:10.1007/S00446-016-0274-Y.

[39] Gianlorenzo D'Angelo, Gabriele Di Stefano, and Alfredo Navarra. *Gathering Asynchronous and Oblivious Robots on Basic Graph Topologies Under the Look-Compute-Move Model*, pages 197–222. Springer New York, New York, NY, 2013. doi:10.1007/978-1-4614-6825-7_13.

[40] Gianlorenzo D'Angelo, Gabriele Di Stefano, and Alfredo Navarra. Gathering on rings under the look-compute-move model. *Distributed Comput.*, 27(4):255–285, 2014. URL: https://doi.org/10.1007/s00446-014-0212-9, doi:10.1007/S00446-014-0212-9.

[41] Shantanu Das, Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Masafumi Yamashita. Autonomous mobile robots with lights. *Theor. Comput. Sci.*, 609:171–184, 2016. URL: https://doi.org/10.1016/j.tcs.2015.09.018, doi:10.1016/J.TCS.2015.09.018.

[42] Shantanu Das, Paola Flocchini, Nicola Santoro, and Masafumi Yamashita. Forming sequences of geometric patterns with oblivious mobile robots. *Distributed Comput.*, 28(2):131–145, 2015. URL: https://doi.org/10.1007/s00446-014-0220-9, doi:10.1007/S00446-014-0220-9.

[43] Xavier Défago and Akihiko Konagaya. Circle formation for oblivious anonymous mobile robots with no common sense of orientation. In *Proceedings of the Second ACM International Workshop on Principles of Mobile Computing*, POMC '02, page 97–104, New York, NY, USA, 2002. Association for Computing Machinery. doi:10.1145/584490.584509.

[44] Xavier Défago, Maria Potop-Butucaru, and Philippe Raipin Parvédy. Self-stabilizing gathering of mobile robots under crash or byzantine faults. *Distributed Comput.*, 33(5):393–421, 2020. URL: https://doi.org/10.1007/s00446-019-00359-x, doi:10.1007/S00446-019-00359-X.

[45] Xavier Défago and Samia Souissi. Non-uniform circle formation algorithm for oblivious mobile robots with convergence toward uniformity. *Theor. Comput. Sci.*, 396(1-3):97–112, 2008. doi:10.1016/j.tcs.2008.01.050.

[46] Mattia D'Emidio, Daniele Frigioni, and Alfredo Navarra. Characterizing the computational power of anonymous mobile robots. In *36th IEEE International Conference on Distributed Computing Systems, ICDCS 2016, Nara, Japan, June 27-30, 2016*, pages 293–302. IEEE Computer Society, 2016. doi:10.1109/ICDCS.2016.58.

[47] Yoann Dieudonné, Franck Petit, and Vincent Villain. Leader election problem versus pattern formation problem. In Nancy A. Lynch and Alexander A. Shvartsman, editors, *Distributed Computing, 24th International Symposium, DISC 2010, Cambridge, MA, USA, September 13-15, 2010. Proceedings*, volume 6343 of *Lecture Notes in Computer Science*, pages 267–281. Springer, 2010. doi:10.1007/978-3-642-15763-9_26.

[48] Ayan Dutta, Sruti Gan Chaudhuri, Suparno Datta, and Krishnendu Mukhopadhyaya. Circle formation by asynchronous fat robots with limited visibility. In Ramaswamy Ramanujam and Srini Ramaswamy, editors, *Distributed Computing and Internet Technology - 8th International Conference, ICDCIT 2012, Bhubaneswar, India, February 2-4, 2012. Proceedings*, volume 7154 of *Lecture Notes in Computer Science*, pages 83–93. Springer, 2012. doi:10.1007/978-3-642-28073-3_8.

[49] Caterina Feletti, Carlo Mereghetti, and Beatrice Palano. Uniform circle formation for swarms of opaque robots with lights. In Taisuke Izumi and Petr Kuznetsov, editors, *Stabilization, Safety, and Security of Distributed Systems - 20th International Symposium, SSS 2018, Tokyo, Japan, November 4-7, 2018, Proceedings*, volume 11201 of *Lecture Notes in Computer Science*, pages 317–332. Springer, 2018. doi:10.1007/978-3-030-03232-6_21.

[50] Caterina Feletti, Carlo Mereghetti, and Beatrice Palano. *Uniform Circle Formation for Swarms of Opaque Robots with Lights: 20th International Symposium, SSS 2018, Tokyo, Japan, November 4–7, 2018, Proceedings*, pages 317–332. 01 2018. doi:10.1007/978-3-030-03232-6_21.

[51] Caterina Feletti, Carlo Mereghetti, and Beatrice Palano. Uniform circle formation for fully, semi-, and asynchronous opaque robots with lights. *Applied Sciences*, 13(13), 2023. URL: https://www.mdpi.com/2076-3417/13/13/7991, doi:10.3390/app13137991.

[52] Caterina Feletti, Carlo Mereghetti, Beatrice Palano, and Priscilla Raucci. Uniform circle formation for fully semi-, and asynchronous opaque robots

with lights. In Ugo Dal Lago and Daniele Gorla, editors, *Proceedings of the 23rd Italian Conference on Theoretical Computer Science, ICTCS 2022, Rome, Italy, September 7-9, 2022*, volume 3284 of *CEUR Workshop Proceedings*, pages 207–221. CEUR-WS.org, 2022. URL: https://ceur-ws.org/Vol-3284/8511.pdf.

[53] Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro, editors. *Distributed Computing by Mobile Entities, Current Research in Moving and Computing*, volume 11340 of *Lecture Notes in Computer Science*. Springer, 2019. doi:10.1007/978-3-030-11072-7.

[54] Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Giovanni Viglietta. Distributed computing by mobile robots: uniform circle formation. *Distributed Comput.*, 30(6):413–457, 2017. doi:10.1007/s00446-016-0291-x.

[55] Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Peter Widmayer. Gathering of asynchronous robots with limited visibility. *Theor. Comput. Sci.*, 337(1-3):147–168, 2005. URL: https://doi.org/10.1016/j.tcs.2005.01.001, doi:10.1016/J.TCS.2005.01.001.

[56] Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Peter Widmayer. Arbitrary pattern formation by asynchronous, anonymous, oblivious robots. *Theor. Comput. Sci.*, 407(1-3):412–447, 2008. URL: https://doi.org/10.1016/j.tcs.2008.07.026, doi:10.1016/J.TCS.2008.07.026.

[57] Paola Flocchini, Nicola Santoro, Giovanni Viglietta, and Masafumi Yamashita. Rendezvous with constant memory. *Theor. Comput. Sci.*, 621:57–72, 2016. URL: https://doi.org/10.1016/j.tcs.2016.01.025, doi:10.1016/J.TCS.2016.01.025.

[58] Nao Fujinaga, Yukiko Yamauchi, Hirotaka Ono, Shuji Kijima, and Masafumi Yamashita. Pattern formation by oblivious asynchronous mobile robots. *SIAM J. Comput.*, 44(3):740–785, 2015. doi:10.1137/140958682.

[59] Vincenzo Gervasi and Giuseppe Prencipe. Coordination without communication: the case of the flocking problem. *Discret. Appl. Math.*, 144(3):324–344, 2004. URL: https://doi.org/10.1016/j.dam.2003.11.010, doi:10.1016/J.DAM.2003.11.010.

[60] Satakshi Ghosh, Pritam Goswami, Avisek Sharma, and Buddhadeb Sau. Move optimal and time optimal arbitrary pattern formations by asynchronous robots on infinite grid. *Int. J. Parallel Emergent Distributed Syst.*, 38(1):35–57, 2023. doi:10.1080/17445760.2022.2124411.

[61] Rory Hector, Ramachandran Vaidyanathan, Gokarna Sharma, and Jerry L. Trahan. Optimal convex hull formation on a grid by asynchronous robots with lights. *IEEE Trans. Parallel Distributed Syst.*, 33(12):3532–3545, 2022. doi:10.1109/TPDS.2022.3158202.

[62] Yoshiaki Ito, Yonghwan Kim, and Yoshiaki Katayama. Brief announcement: Mutually-visible uniform circle formation by asynchronous mobile robots on grid plane. In *International Symposium on Stabilizing, Safety, and Security of Distributed Systems*, pages 352–357. Springer, 2022.

[63] 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 J. Comput.*, 41(1):26–46, 2012. doi:10.1137/100797916.

[64] Tomoko Izumi, Taisuke Izumi, Sayaka Kamei, and Fukuhito Ooshita. Mobile robots gathering algorithm with local weak multiplicity in rings. In Boaz Patt-Shamir and Tínaz Ekim, editors, *Structural Information and Communication Complexity, 17th International Colloquium, SIROCCO 2010, Sirince, Turkey, June 7-11, 2010. Proceedings*, volume 6058 of *Lecture Notes in Computer Science*, pages 101–113. Springer, 2010. doi:10.1007/978-3-642-13284-1_9.

[65] Mohammad R Jahanshahi, Wei-Men Shen, Tarutal Ghosh Mondal, Mohamed Abdelbarr, Sami F Masri, and Uvais A Qidwai. Reconfigurable

swarm robots for structural health monitoring: a brief review. *International Journal of Intelligent Robotics and Applications*, 1(3):287–305, 2017.

[66] Sayaka Kamei, Anissa Lamani, Fukuhito Ooshita, and Sébastien Tixeuil. Asynchronous mobile robot gathering from symmetric configurations without global multiplicity detection. In Adrian Kosowski and Masafumi Yamashita, editors, *Structural Information and Communication Complexity - 18th International Colloquium, SIROCCO 2011, Gdansk, Poland, June 26-29, 2011. Proceedings*, volume 6796 of *Lecture Notes in Computer Science*, pages 150–161. Springer, 2011. doi:10.1007/978-3-642-22212-2_14.

[67] Akitoshi Kawamura and Yusuke Kobayashi. Fence patrolling by mobile agents with distinct speeds. *Distributed Comput.*, 28(2):147–154, 2015. URL: https://doi.org/10.1007/s00446-014-0226-3, doi:10.1007/S00446-014-0226-3.

[68] Ralf Klasing, Euripides Markou, and Andrzej Pelc. Gathering asynchronous oblivious mobile robots in a ring. *Theor. Comput. Sci.*, 390(1):27–39, 2008. URL: https://doi.org/10.1016/j.tcs.2007.09.032, doi:10.1016/J.TCS.2007.09.032.

[69] Manash Kumar Kundu, Pritam Goswami, Satakshi Ghosh, and Buddhadeb Sau. Arbitrary pattern formation by opaque fat robots on infinite grid. *Int. J. Parallel Emergent Distributed Syst.*, 37(5):542–570, 2022. doi:10.1080/17445760.2022.2088750.

[70] Andrey Kupavskii and Emo Welzl. Lower bounds for searching robots, some faulty. *Distributed Comput.*, 34(4):229–237, 2021. URL: https://doi.org/10.1007/s00446-019-00358-y, doi:10.1007/S00446-019-00358-Y.

[71] Giuseppe Antonio Di Luna, Paola Flocchini, Sruti Gan Chaudhuri, Federico Poloni, Nicola Santoro, and Giovanni Viglietta. Mutual visibility by luminous robots without collisions. *Inf. Comput.*, 254:392–418, 2017. URL: https://doi.org/10.1016/j.ic.2016.09.005, doi:10.1016/J.IC.2016.09.005.

[72] Giuseppe Antonio Di Luna, Paola Flocchini, Federico Poloni, Nicola Santoro, and Giovanni Viglietta. The mutual visibility problem for oblivious robots. In *Proceedings of the 26th Canadian Conference on Computational Geometry, CCCG 2014, Halifax, Nova Scotia, Canada, 2014*. Carleton University, Ottawa, Canada, 2014. URL: http://www.cccg.ca/proceedings/2014/papers/paper51.pdf.

[73] Maja Mataric. *Interaction and Intelligent Behavior*. PhD thesis, PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 1994.

[74] Moumita Mondal and Sruti Gan Chaudhuri. Uniform circle formation by fat robots under non-uniform visibility ranges. In Nandini Mukherjee and Sriram V. Pemmaraju, editors, *ICDCN 2020: 21st International Conference on Distributed Computing and Networking, Kolkata, India, January 4-7, 2020*, pages 58:1–58:5. ACM, 2020. doi:10.1145/3369740.3372779.

[75] 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, Hyderabad, India, January 5-7, 2017*, page 7. ACM, 2017. URL: http://dl.acm.org/citation.cfm?id=3007786.

[76] Debasish Pattanayak, H. Ramesh, and Partha Sarathi Mandal. Collaborative evacuation of mobile robots. In Nandini Mukherjee and Sriram V. Pemmaraju, editors, *ICDCN 2020: 21st International Conference on Distributed Computing and Networking, Kolkata, India, January 4-7, 2020*, page 64:1. ACM, 2020. doi:10.1145/3369740.3373806.

[77] Debasish Pattanayak and Gokarna Sharma. Time-color tradeoff on uniform circle formation by asynchronous robots. In *IEEE International Parallel and Distributed Processing Symposium, IPDPS 2024, San Francisco, CA, USA, May 27-31, 2024*, pages 987–997. IEEE, 2024. doi:10.1109/IPDPS57955.2024.00092.

[78] Pavan Poudel and Gokarna Sharma. Time-optimal gathering under limited visibility with one-axis agreement. *Inf.*, 12(11):448, 2021. URL: https://doi.org/10.3390/info12110448, doi:10.3390/INFO12110448.

[79] Pavan Poudel, Gokarna Sharma, and Aisha Aljohani. Sublinear-time mutual visibility for fat oblivious robots. In R. C. Hansdah, Dilip Krishnaswamy, and Nitin H. Vaidya, editors, *Proceedings of the 20th International Conference on Distributed Computing and Networking, ICDCN 2019, Bangalore, India, January 04-07, 2019*, pages 238–247. ACM, 2019. doi:10.1145/3288599.3288602.

[80] Paul Scharre. Robotics on the battlefield part II: The coming swarm. *Center for New American Security*, 2014.

[81] Gokarna Sharma, Rusul Alsaedi, Costas Busch, and Supratik Mukhopadhyay. The complete visibility problem for fat robots with lights. In Paolo Bellavista and Vijay K. Garg, editors, *Proceedings of the 19th International Conference on Distributed Computing and Networking, ICDCN 2018, Varanasi, India, January 4-7, 2018*, pages 21:1–21:4. ACM, 2018. doi:10.1145/3154273.3154319.

[82] Gokarna Sharma, Costas Busch, and Supratik Mukhopadhyay. Bounds on mutual visibility algorithms. In *Proceedings of the 27th Canadian Conference on Computational Geometry, CCCG 2015, Kingston, Ontario, Canada, August 10-12, 2015*. Queen's University, Ontario, Canada, 2015. URL: http://research.cs.queensu.ca/cccg2015/CCCG15-papers/43.pdf.

[83] Gokarna Sharma, Costas Busch, and Supratik Mukhopadhyay. Mutual visibility with an optimal number of colors. In Prosenjit Bose, Leszek Antoni Gasieniec, Kay Römer, and Roger Wattenhofer, editors, *Algorithms for Sensor Systems - 11th International Symposium on Algorithms and Experiments for Wireless Sensor Networks, ALGOSENSORS 2015, Patras, Greece, September 17-18, 2015, Revised Selected Papers*, volume 9536

of *Lecture Notes in Computer Science*, pages 196–210. Springer, 2015. doi:10.1007/978-3-319-28472-9_15.

[84] Gokarna Sharma, Ramachandran Vaidyanathan, Jerry L. Trahan, Costas Busch, and Suresh Rai. Complete visibility for robots with lights in O(1) time. In Borzoo Bonakdarpour and Franck Petit, editors, *Stabilization, Safety, and Security of Distributed Systems - 18th International Symposium, SSS 2016, Lyon, France, November 7-10, 2016, Proceedings*, volume 10083 of *Lecture Notes in Computer Science*, pages 327–345, 2016. doi:10.1007/978-3-319-49259-9_26.

[85] Gokarna Sharma, Ramachandran Vaidyanathan, Jerry L. Trahan, Costas Busch, and Suresh Rai. O(log n)-time complete visibility for asynchronous robots with lights. In *2017 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2017, Orlando, FL, USA, May 29 - June 2, 2017*, pages 513–522. IEEE Computer Society, 2017. doi:10.1109/IPDPS.2017.51.

[86] Arijit Sil and Sruti Gan Chaudhuri. Formation of straight line by swarm robots. In Jyotsna Kumar Mandal, Imon Mukherjee, Sambit Bakshi, Sanjay Chatterji, and Pankaj K. Sa, editors, *Computational Intelligence and Machine Learning*, pages 99–111, Singapore, 2021. Springer Singapore.

[87] Samia Souissi, Taisuke Izumi, and Koichi Wada. Oracle-based flocking of mobile robots in crash-recovery model. *Theor. Comput. Sci.*, 412(33):4350–4360, 2011. URL: https://doi.org/10.1016/j.tcs.2010.11.011, doi:10.1016/J.TCS.2010.11.011.

[88] Gabriele Di Stefano, Pietro Montanari, and Alfredo Navarra. About ungatherability of oblivious and asynchronous robots on anonymous rings. In Zsuzsanna Lipták and William F. Smyth, editors, *Combinatorial Algorithms - 26th International Workshop, IWOCA 2015, Verona, Italy, October 5-7, 2015, Revised Selected Papers*, volume 9538 of *Lecture Notes in Computer Science*, pages 136–147. Springer, 2015. doi:10.1007/978-3-319-29516-9_12.

[89] Gabriele Di Stefano and Alfredo Navarra. Gathering of oblivious robots on infinite grids with minimum traveled distance. *Inf. Comput.*, 254:377–391, 2017. URL: https://doi.org/10.1016/j.ic.2016.09.004, doi:10.1016/J.IC.2016.09.004.

[90] Daniel P Stormont. Autonomous rescue robot swarms for first responders. In *CIHSPS 2005. Proceedings of the 2005 IEEE International Conference on Computational Intelligence for Homeland Security and Personal Safety, 2005.*, pages 151–157. IEEE, 2005. doi:10.1109/CIHSPS.2005.1500631.

[91] 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.

[92] Kokichi Sugihara and Ichiro Suzuki. Distributed motion coordination of multiple mobile robots. *Proceedings. 5th IEEE International Symposium on Intelligent Control 1990*, pages 138–143 vol.1, 1990. URL: https://api.semanticscholar.org/CorpusID:61599595.

[93] Ichiro Suzuki and Masafumi Yamashita. Distributed anonymous mobile robots. In Nicola Santoro and Paul G. Spirakis, editors, *SIROCCO'96, The 3rd International Colloquium on Structural Information & Communication Complexity, Siena, Italy, June 6-8, 1996*, pages 313–330. Carleton Scientific, 1996.

[94] Ichiro Suzuki and Masafumi Yamashita. Distributed anonymous mobile robots. In Nicola Santoro and Paul G. Spirakis, editors, *SIROCCO'96, The 3rd International Colloquium on Structural Information & Communication Complexity, Siena, Italy, June 6-8, 1996*, pages 313–330. Carleton Scientific, 1996.

[95] Ichiro Suzuki and Masafumi Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM J. Comput.*, 28(4):1347–1363, 1999. doi:10.1137/S009753979628292X.

[96] O Tanaka. Forming a circle by distributed anonymous mobile robots. *Bachelor thesis, Department of Electrical Engineering, Hiroshima University, Hiroshima, Japan*, 1992.

[97] Yusaku Tomita, Yukiko Yamauchi, Shuji Kijima, and Masafumi Yamashita. Plane formation by synchronous mobile robots without chirality. In James Aspnes, Alysson Bessani, Pascal Felber, and João Leitão, editors, *21st International Conference on Principles of Distributed Systems, OPODIS 2017, Lisbon, Portugal, December 18-20, 2017*, volume 95 of *LIPIcs*, pages 13:1–13:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. URL: https://doi.org/10.4230/LIPIcs.OPODIS.2017.13, doi:10.4230/LIPICS.OPODIS.2017.13.

[98] Taichi Uehara, Yukiko Yamauchi, Shuji Kijima, and Masafumi Yamashita. Plane formation by semi-synchronous robots in the three dimensional euclidean space. In Borzoo Bonakdarpour and Franck Petit, editors, *Stabilization, Safety, and Security of Distributed Systems - 18th International Symposium, SSS 2016, Lyon, France, November 7-10, 2016, Proceedings*, volume 10083 of *Lecture Notes in Computer Science*, pages 383–398, 2016. doi:10.1007/978-3-319-49259-9_30.

[99] Giovanni Viglietta. Rendezvous of two robots with visible bits. In Paola Flocchini, Jie Gao, Evangelos Kranakis, and Friedhelm Meyer auf der Heide, editors, *Algorithms for Sensor Systems - 9th International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics, ALGOSENSORS 2013, Sophia Antipolis, France, September 5-6, 2013, Revised Selected Papers*, volume 8243 of *Lecture Notes in Computer Science*, pages 291–306. Springer, 2013. doi:10.1007/978-3-642-45346-5_21.

[100] Masafumi Yamashita and Ichiro Suzuki. Characterizing geometric patterns formable by oblivious anonymous mobile robots. *Theor. Comput. Sci.*, 411(26-28):2433–2453, 2010. URL: https://doi.org/10.1016/j.tcs.2010.01.037, doi:10.1016/J.TCS.2010.01.037.

[101] Yukiko Yamauchi, Taichi Uehara, Shuji Kijima, and Masafumi Yamashita. Plane formation by synchronous mobile robots in the three-dimensional euclidean space. *J. ACM*, 64(3):16:1–16:43, 2017. doi:10.1145/3060272.

[102] Yukiko Yamauchi and Masafumi Yamashita. Pattern formation by mobile robots with limited visibility. In Thomas Moscibroda and Adele A. Rescigno, editors, *Structural Information and Communication Complexity - 20th International Colloquium, SIROCCO 2013, Ischia, Italy, July 1-3, 2013, Revised Selected Papers*, volume 8179 of *Lecture Notes in Computer Science*, pages 201–212. Springer, 2013. doi:10.1007/978-3-319-03578-9_17.

[103] Yukiko Yamauchi and Masafumi Yamashita. Randomized pattern formation algorithm for asynchronous oblivious mobile robots. In Fabian Kuhn, editor, *Distributed Computing - 28th International Symposium, DISC 2014, Austin, TX, USA, October 12-15, 2014. Proceedings*, volume 8784 of *Lecture Notes in Computer Science*, pages 137–151. Springer, 2014. doi:10.1007/978-3-662-45174-8_10.

[104] Yan Yang, Samia Souissi, Xavier Défago, and Makoto Takizawa. Fault-tolerant flocking for a group of autonomous mobile robots. *J. Syst. Softw.*, 84(1):29–36, 2011. URL: https://doi.org/10.1016/j.jss.2010.08.026, doi:10.1016/J.JSS.2010.08.026.

[105] Honghai Zhang and Jennifer C. Hou. Maintaining sensing coverage and connectivity in large sensor networks. *Ad Hoc Sens. Wirel. Networks*, 1(1-2):89–124, 2005. URL: http://www.oldcitypublishing.com/journals/ahswn-home/ahswn-issue-contents/ahswn-volume-1-number-1-2-2005/ahswn-1-1-2-p-89-124/.

List of Publications

Based on this thesis, the following papers have been published:

- Pritam Goswami, Avisek Sharma, Satakshi Ghosh, Buddhadeb Sau: **Time Optimal Gathering of Myopic Robots on an Infinite Triangular Grid**. Theoretical Computer Science, 1023:114930, 2025

  https://www.sciencedirect.com/science/article/pii/S0304397524005474

- Pritam Goswami, Avisek Sharma, Satakshi Ghosh, Buddhadeb Sau: **Brief Announcement: Rendezvous on a Known Dynamic Point in a Finite Unoriented Grid.** SSS 2023: 374-379.

  https://doi.org/10.1007/978-3-031-44274-2_27

- Pritam Goswami, Manash Kumar Kundu, Satakshi Ghosh, Buddhadeb Sau: **Circle Formation by Asynchronous Opaque Fat Robots on an Infinite Grid.** Int. J. Parallel Emergent Distributed Syst., 39(2):214–247, 2024

  https://doi.org/10.1080/17445760.2024.2316017

Index

$\mathcal{APF}$, 16
$\mathcal{ASYNC}$, 14
$\mathcal{FCOM}$ robots, 14
$\mathcal{FSTA}$ robots, 13
$\mathcal{FSYNC}$, 14
$\mathcal{LUMI}$ robots, 13
$\mathcal{OBLOT}$ robots, 13
$\mathcal{SSYNC}$, 14
Arbitrary Pattern Formation, 16
Formation problem, 16
Gathering, 18

activation, 8
anonymous, 6
asynchronous, 14
autonomous, 6

epoch, 26

fat robot, 6
fully-synchronous, 14

homogeneous, 6

identical, 6
infinite triangular grid, 26

limited vision, 11

local coordinate system, 7

Multiplicity, 10
myopic robot, 11

non-restricted visibility, 11
Non-rigid, 16

oblivious, 13
obstructed visibility, 11
opaque robots, 12

point robot, 6

Rendezvous, 18
restricted visibility, 11
rigid, 16
robot swarm, 6

scheduler, 14
Semi-Synchronous, 14
silent, 13
stigmergy, 4

two axis agreement, 7

visibility range, 11
visibility region, 11