Gathering and Arbitrary Pattern Formation by Asynchronous Robot Swarm

Doctoral dissertation submitted by

Satakshi Ghosh

for the award of the PhD degree of

JADAVPUR UNIVERSITY

Kolkata, India

Supervisor:

Prof. Buddhadeb Sau

Department of Mathematics

Jadavpur University

Kolkata, India

NOVEMBER, 2023

Certificate from the Supervisor

This is to certify that this thesis entitled "*Gathering and Arbitrary Pattern Formation by Asynchronous Robot Swarm*" submitted by **Ms. Satakshi Ghosh** who got her name registered on **5th September, 2018** for the award of Ph.D (Science) degree of Jadavpur University, is absolutely based upon her 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.

....

(Signature of the Supervisor with date and official seal)

Dedicated To

MY DIDA

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 September, 2018 and November, 2023.

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:

- Satakshi Ghosh, Pritam Goswami, Avisek Sharma, 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). https://doi.org/10.1080/17445760.2022.2124411

- Satakshi Ghosh, Avisek Sharma, Pritam Goswami, Buddhadeb Sau: **Oblivious Robots Performing Different Tasks on Grid Without Knowing Their Team Members.** ICARA 2023: 180-186. https://doi.org/10.1109/ICARA56516.2023.10125816

- Satakshi Ghosh, Avisek Sharma, Pritam Goswami, Buddhadeb Sau: **Brief Announcement: Asynchronous Gathering of Finite Memory Robots on a Circle Under Limited Visibility.** SSS 2023: 430-434. https://doi.org/10.1007/978-3-031-44274-2_32

.....

Date: Satakshi Ghosh

Acknowledgement

In my academic odyssey, some extraordinary individuals have played pivotal roles, guiding and supporting me through myriad challenges, ultimately leading to the rewarding culmination of my Ph.D. journey. Their influence and contributions have left indelible imprints on both my academic and personal growth.

Foremost, I must express my deepest gratitude to my dedicated supervisor, Professor Buddhadeb Sau. His expert guidance and genuine enthusiasm for my research have not only been invaluable but transformative. Beyond the academic realm, his mentorship and wisdom have deeply influenced me, shaping not only the scholar I've become but also the person I aspire to be.

My heartfelt gratitude also goes to the illustrious faculty of the Department of Mathematics at Jadavpur University. Their dedication to teaching and research has been an unending source of inspiration, guiding my scholarly pursuits and fostering a love for learning.

I'd like to extend my gratefulness to Dr. Rajib Kumar Das, whose presence on my Research Advisory Committee provided me with precious moments of knowledge-sharing.

A special note of appreciation goes out to all the members and office staff of Jadavpur University. Their tireless efforts behind the scenes, from administrative tasks to maintaining the academic infrastructure, have significantly eased my journey.

I would like to acknowledge the financial support provided by Jadavpur University, Kolkata, through the RUSA scheme and the West Bengal state government fellowship scheme. These initiatives were instrumental in facilitating my academic career.

My research collaborators, Pritam Goswami, Avisek Sharma, Archak Das, and Dr. Manash Kumar Kundu deserve the most special mention. Our collective efforts and shared ideas have not only led to the publication of all the research

papers but also have nurtured an environment of intellectual synergy and growth. I will always be grateful to Pritam Goswami and Avisek Sharma, for without them it would not have been possible for me to achieve this dream. Furthermore, I'm profoundly thankful to Dr. Kaustav Bose. His motivation and belief in our endeavors have been the driving force behind my work. Within the academic community, the companionship of my labmates Brati Mondal, Raja Das, Dr. Ranendu Adhikary, Debajyoti Biswas, Dr. Sangita Patra, Dr. Ananya Saha, Dr. Arpita Dey, and Dr. Suvankar Barai has been a priceless gift. Our discussions, camaraderie, and collaborative spirit have been instrumental in shaping my academic journey.

To my parents, I owe a debt of gratitude that words can scarcely capture. Their love and belief in me, the sacrifices they've made for my education, and the values they've instilled in me have formed the bedrock of my every achievement.

The two pillars of my strength are my sister Prithwa Ghosh and my best friend Trisha Nandi. These two are my lifeline and they make every challenge seem easy. All my family members, Kritartha Ghosh, Krishna Ghosh, Bedatrayee Goswami, Parananda Mandal, Krittika Ghosh, Manipuspak Ghosh, Achyut Goswami, Pranay Senapati, Pulak Ghosh have been my constant source of strength. I especially thank my father and mother-in-law for their understanding and support. To all my friends, Antara Sensarma, Subhechha Das, Suman Chakraborty, Trina Bhowmick, Rituparna Saha, Raka Som, Kushal Guha Bakshi, Rajkumar Nayak, Rakibul Haque, Deep Roy, Aritra Dutta, the laughter we shared did make the most challenging times bearable. I thank Sangita Sarkar and Torsha Nandi for never stopping believing in me.

Lastly, my husband, Pallab Ghosh, has been my unwavering anchor. His faith in my abilities, his patience during the demanding phases of my research, and his unending love for me through thick and thin have been my greatest motivation and I couldn't have asked for a more supportive partner.

I feel immensely blessed to have such an incredible support system, and I recognize that this achievement wouldn't have been possible without each and every one of them. From the depths of my heart, thank you.

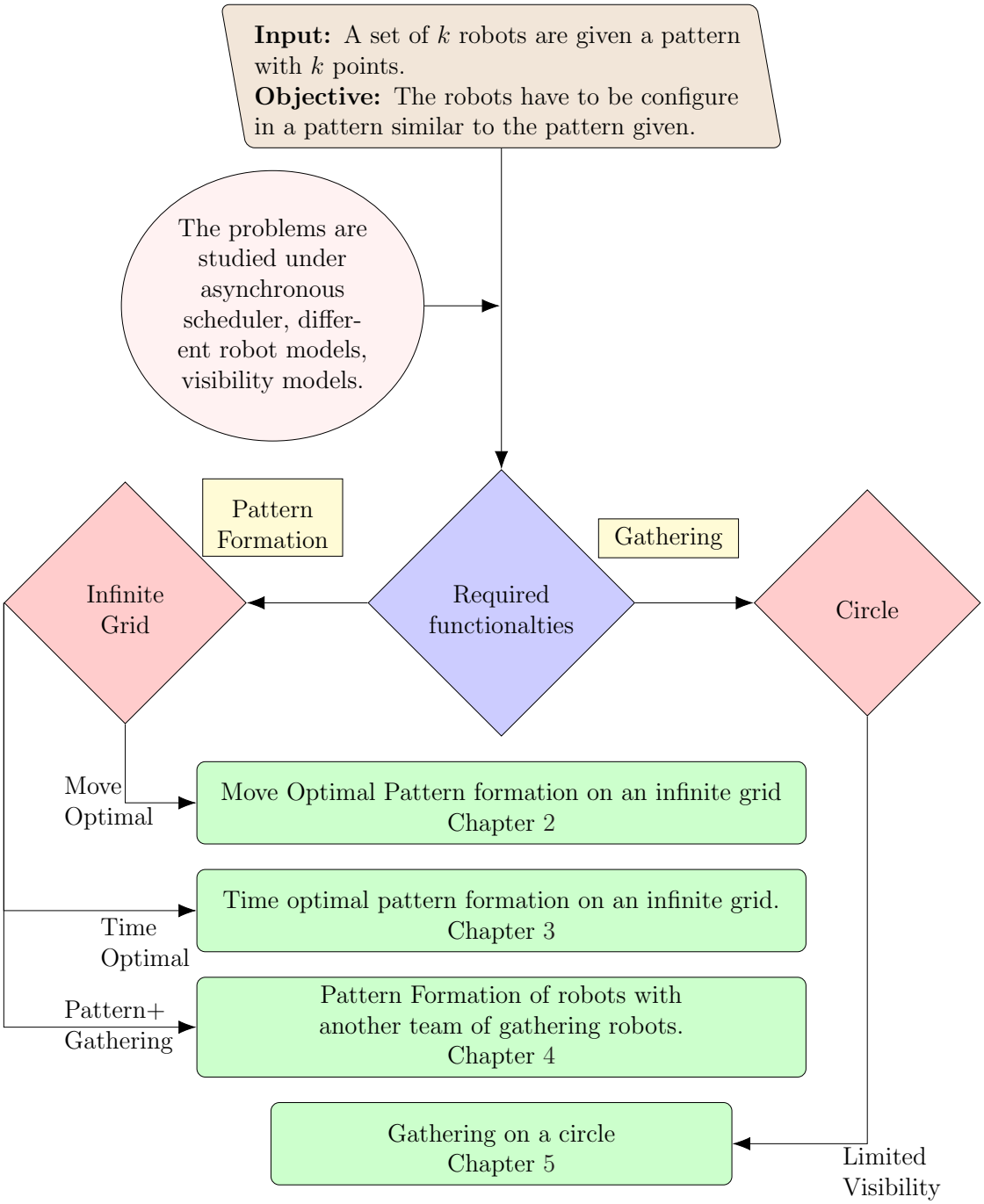

Figure 1: Overview of the contribution of the thesis

Contents

Chapter 1

Introduction

Swarm robotics is a niche within the field of robotics that places a strong emphasis on accomplishing tasks using robots that possess only the most fundamental capabilities. This approach has garnered significant attention and research focus over the last two decades. The central challenge in swarm robotics revolves around coordinating multiple robots effectively to work together in harmony. Unlike traditional robotics, where individual robots often have extensive capabilities and functionality, swarm robots are intentionally designed to be simplistic. They are equipped with only basic features and functionalities to reduce cost, complexity, and development efforts.

The reasons behind this simplicity is rooted in practicality. Building and deploying robots with advanced capabilities can be prohibitively expensive and technically intricate. In contrast, designing and deploying a swarm of robots with minimal capabilities is a far more economical and straightforward endeavor.

The real-world applications of swarm robotics are diverse and encompass areas such as search and rescue missions, environmental monitoring, precision agriculture, and industrial automation. In these scenarios, the collective power of numerous simple robots working together can offer distinct advantages. These advantages include enhanced scalability, fault tolerance, and cost-effectiveness, making swarm robotics an attractive solution for many challenging tasks. In summary, swarm robotics is characterized by its pursuit of task completion through

the collaboration of robots with minimal capabilities. This approach has gained traction due to its potential to address coordination challenges while providing a cost-effective and practical alternative to complex, high-capability robotic solutions.

This concept draws inspiration from the collective behavior of social animals, such as ants and bees, where individual organisms with limited capabilities collaborate to achieve complex objectives. Robot swarms have gained significant attention in robotics research and various application domains due to their potential advantages in scalability, fault tolerance, and cost-effectiveness. Robot swarms have a wide range of applications across various fields due to their ability to work collectively and efficiently. Here are some notable applications of robot swarms:

Search and Rescue: Robot swarms can be deployed in disaster-stricken areas to search for survivors and assess damage. They can cover large areas quickly, locate people in need of help, and transmit critical information back to rescue teams.

Precision Agriculture: Swarms of agricultural robots can plant seeds, monitor crop health, and apply fertilizers or pesticides more precisely. They can work continuously, reduce the need for manual labor, and optimize resource usage.

Environmental Monitoring: Robot swarms can be used to monitor environmental conditions in oceans, forests, or urban areas. They can collect data on temperature, humidity, pollution levels, and wildlife behavior.

Manufacturing and Logistics: In manufacturing, swarms of robots can collaborate to assemble products efficiently. In logistics, they can work together to optimize warehouse operations, package sorting, and inventory management.

Aerial Surveillance: Drones in swarms can provide aerial surveillance for security, crowd monitoring, or wildlife tracking. They can cover large areas

simultaneously and capture data from multiple angles.

Medical Applications: Robot swarms can assist in minimally invasive surgeries by providing precise control and visualization. They can also transport medical supplies in hospital environments.

Construction and Infrastructure: Swarms of construction robots can work together to build structures, lay bricks, or perform maintenance tasks. They can improve efficiency and safety on construction sites.

Exploration and Mapping: Swarms of autonomous robots can explore unknown or hazardous environments, such as caves, mines, or other planets. They can create detailed maps and collect data without putting humans at risk.

Underwater Exploration: In marine research, robot swarms can study the ocean, including coral reefs, shipwrecks, and marine life. They can be deployed for extended periods and reach depths that are difficult for humans.

Military and Defense: Swarms of drones or ground robots can be used for reconnaissance, surveillance, and communication relay in military operations. They can provide a tactical advantage by enhancing situational awareness.

Entertainment and Art: Robot swarms can create visually stunning light shows or choreographed performances for entertainment purposes. They offer unique possibilities for artistic expression and audience engagement.

These are just a few examples of the diverse applications of robot swarms. As technology continues to advance and researchers develop new algorithms and capabilities, the potential for robot swarms in various industries is expected to expand even further.

1.1 Theoretical Framework

This section provides an overview of the theoretical framework under which the computational and complexity issues related to distributed computing by robot

swarms are studied. The framework covers a large spectrum of settings. The different sets of assumptions regarding the capabilities of the robots and the environment in which they operate give rise to several variations of the framework.

1.1.1 The Robots

A *robot swarm* is a collection of computational entities known as "robots" each possessing the ability to perform local computations and move independently. These robots exhibit several key characteristics: they are *autonomous* (lacking centralized control), *anonymous* (lacking unique identifiers), *identical* (physically indistinguishable), and *homogeneous* (executing the same algorithm).

In terms of their physical attributes, most literature in this field assumes the "*point robot*" model, which simplifies robots to dimensionless points in a two-dimensional plane. However, this simplistic representation ignores the practical reality that even small robots have physical dimensions. To address this limitation, the "*fat robot*" model was introduced in the literature. In the fat robot model, robots are depicted as identical disks, acknowledging that they occupy physical space, making it a more realistic representation for scenarios involving physical interactions and constraints.

1.1.2 Visibility

Robots have capability of observing its surroundings. There are mainly three types of models regarding the visibility of the robots.

Full visibility: *Full visibility* is the common visibility model. In this model, each robot is able to observe all the other robots in the domain.

Limited visibility: In the *limited visibility* model, the visibility range of each robot is limited i.e. a robot can not observe the entire domain of robots. There is a finite distance $D > 0$ such that two robots can see each other if and only if the distance between them is $\leq D$.

Obstructed visibility: In the *obstructed visibility* or *opaque robot* model, the visibility range of each robot is unlimited, but its visibility can be obstructed by the presence of other robots. Formally, two robots p and q are able to see each other if and only if no other robot lies on the line segment joining p and q.

1.1.3 Local Coordinate Systems

Each robot maintains its own local coordinate system. For every robot, its current location serves as the origin in this coordinate system, and this origin moves with the robot as it changes position. Importantly, there is no overarching global coordinate system accessible to any of the robots. Moreover, each robot employs its own unique unit of length for measurements. However, it's possible for some level of consistency to exist among the local coordinate systems employed by the robots, leading to the categorization of four primary models based on the extent of this consistency.

Two axis agreement: The robots agree on the direction and orientation of both axes.

One axis agreement: The robots agree on the direction and orientation of only one axis.

Chirality: The robots agree on cyclic orientation i.e., clockwise and anticlockwise.

No agreement: There is no assumption on consistency among the local coordinate systems.

1.1.4 Look-Compute-Move Cycle

The activation of robots operate according to the *Look-Compute-Move (LCM)* cycle. In each cycle, a previously idle or inactive robot wakes up and executes the following steps.

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

Compute: Based on the perceived configuration, the robot executes computations according to an algorithm whether to stay put or to move to the destination point. The deterministic algorithm is same for all robots.

Move: The robot either remain stationary or makes a move to a destination point.

After executing a Look-Compute-Move cycle, the robot becomes inactive. Then after some finite time, it wakes up again to perform another Look-Compute-Move cycle.

1.1.5 Memory and Communication

Based on the memory and communication capabilities, there are four models: $\mathcal{OBLOT}$ (oblivious), $\mathcal{FSTA}$ (finite state), $\mathcal{FCOM}$ (finite communication) and $\mathcal{LUMI}$ (luminous).

$\mathcal{OBLOT}$: The most general and well studied model is $\mathcal{OBLOT}$. Here robots are assumed to be *silent* and *oblivious*. The robots are silent as they have no explicit means of communication. The robots are oblivious in the sense that they have no memory of past observations, computations and actions, i.e., at the end of each Look-Compute-Move cycle, all obtained information (observations, computations, and actions) are erased. Therefore, the computation in each cycle is based solely on what is observed in the current cycle.

$\mathcal{FSTA}$: In $\mathcal{FSTA}$, the robots are silent, but have finite memory. As in $\mathcal{LUMI}$, each robot is equipped with a light. However, the light of a robot is only visible itself. In Look, a robot observes the positions of robots visible to it, but not their lights. However, it can see the colour of its own light. Then based on this, the robot performs computations according to an algorithm

to decide a destination point and colour. Then the robot sets its light to that colour and moves to the computed destination. Note that $\mathcal{FSTA}$ with one colour is also the same as $\mathcal{OBLOT}$.

$\mathcal{FCOM}$: In $\mathcal{FCOM}$, the robots are oblivious, but have finite communication bits. Here, each robot is equipped with a light. However, the light of a robot is only visible to others, but not to itself. In LOOK, a robot observes the positions of robots visible to it and also their lights, but it cannot see the colour of its own light. Then based on this, the robot performs computations according to an algorithm to decide a destination point and colour. Then the robot sets its light to that colour and moves to the computed destination. Again, $\mathcal{FCOM}$ with one colour is the same as $\mathcal{OBLOT}$.

$\mathcal{LUMI}$: In this model, each robot is equipped with a visible light which can assume a finite number of predefined colours. The light of a robot is visible to itself and also by other robots. The lights serve both as a weak explicit communication mechanism and a form of memory. In Look, a robot observes the positions of robots visible to it, along with their lights and also its own light. Then based on this, the robot performs computations according to an algorithm to decide a destination point and colour. Then the robot sets its light to that colour and moves to the computed destination. The light is persistent in the sense that the colour of the light is not erased at the end of a Look-Compute-Move cycle and is retained in the Look phase of the next Look-Compute-Move cycle. Notice that a robot having light with only one possible colour has the same capability as the one with no light. Therefore, $\mathcal{LUMI}$ generalizes the $\mathcal{OBLOT}$ model.

1.1.6 Activation and Synchronization

There are three models: fully synchronous (FSync), semi-synchronous (SSync) and asynchronous (ASync) based on the activation and timing of action of the robots (See also Fig 1.1).

Fully synchronous: Time can be logically divided into global rounds in FSync. All of the robots are activated for each round. They execute their actions simultaneously after taking the snapshots at the same moment. As a result, the robots' Look, Compute, and Move phases are synchronized.

Semi-synchronous: The sole difference between the FSync and SSync models is that not every robot is always activated in every round in the SSync model. Every robot, though, is activated infinitely often.

Asynchronous: ASync is the most general model. The robots are activated independently and each robot initiates its cycle individually. The amount of time spent in Look, Compute, Move and inactive states are finite but unbounded, unpredictable and not same for different robots. The robots do not have a common understanding of time as a result. Additionally, a moving robot can be observed, allowing calculations to be relied on outdated positional data. Additionally, before a robot moves, the configuration that it perceives during the Look phase may alter dramatically.

1.1.7 Movement of the Robots

In the study of robot movements, there are typically two main models considered: "Rigid" and "Non-Rigid." These models pertain to how robots move, whether they follow straight lines or can take curved trajectories. Additionally, the speed of movement plays a role in determining the duration of robot movements in different scenarios:

Rigid Movement Model: In the Rigid movement model, each robot is capable of reaching its intended destination without any interruptions in its path. This means that the robot's movement is continuous, and there are no pauses or halts along the way. The robot smoothly moves from its starting point to its final destination.

Non-Rigid Movement Model: In the Non-Rigid movement model, a robot's movement may pause or stop before it reaches its intended destination.

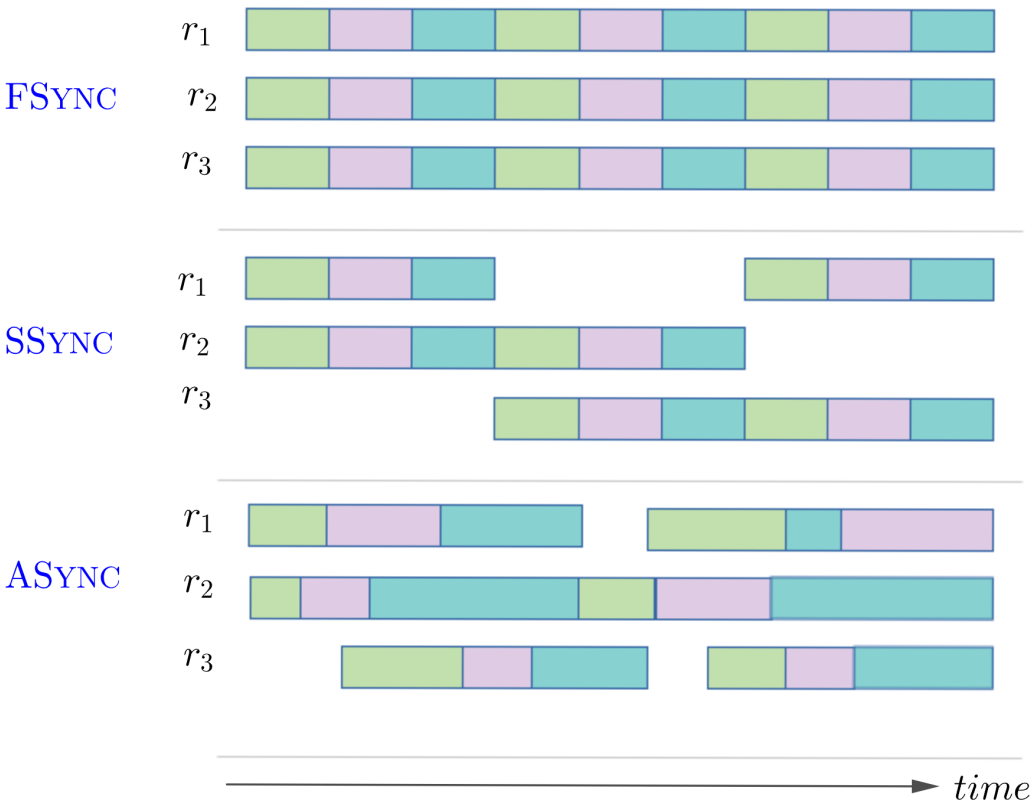

Figure 1.1: The three main models concerning the synchronization of the robots. Here r_1, r_2 and r_3 indicate three robots and green, pink and blue colour respectively indicate the LOOK, COMPUTE and MOVE state of robots r_1, r_2, r_3

However, there is a critical distance parameter denoted as $\delta > 0$. This parameter is significant because it ensures that: if the robot's destination is within a distance of no more than δ units away, then the robot is guaranteed to reach its destination without interruptions. If the destination is farther away, at a distance greater than δ units, then the robot is guaranteed to traverse at least a distance of δ units toward its destination before any potential interruptions occur.

These movement models are crucial in the analysis of robot behavior and coordination, especially in scenarios where robots need to reach specific locations or perform tasks while accounting for potential interruptions or variations in their

movement capabilities.

1.1.8 Topologies under discrete and continuous domain

In this case, robots operate in both continuous and discrete domains, such as a plane or a continuous circle, and the network is modeled as a graph. Examples of some topologies under discrete and continuous domains are given below:

Ring: A ring network is a network topology in which each node connects to exactly two other nodes, forming a single continuous pathway for signals through each node.

Tree: A tree is an undirected graph in which any two nodes are connected by exactly one path, equivalently a connected acyclic undirected graph.

Infinite Grid: An infinite path is the graph $P = (Z, E)$, where $E = \{(i, i+1)|i \in Z\}$. An infinite grid can be defined as the Cartesian product $G = P \times P$. Assume that the infinite grid G is embedded in the Cartesian plane R^2.

Continuous Circle: In a continuous domain, a circle is a geometric shape that consists of all points that are at a fixed distance (known as the radius) from a central point (known as the center). The circle has no discrete points, and it is a smooth and continuous curve with an infinite number of points along its circumference.

1.1.9 Fault tolerance

In the realm of robotics, it is common to assume that robots always operate flawlessly without any errors. However, in reality, robots can experience failures. These failures are typically categorized into two main types: Crash faults and Byzantine faults.

Crash Faults: A crash fault occurs when a robot encounters a problem that causes it to stop functioning permanently. These problems can be caused by

hardware malfunctions, software glitches, or other issues that render the robot inoperative. Crash faults, while significant, are generally considered simpler to address compared to Byzantine faults.

Byzantine Faults: Byzantine faults pose a more complex challenge. In this scenario, a faulty robot may exhibit unpredictable and arbitrary behavior, which can include providing incorrect information to other robots or engaging in malicious actions. Byzantine faults encompass crash faults as a subset. Dealing with Byzantine faults demands the development of robust fault-tolerant algorithms and mechanisms to ensure the correct operation of a robotic system in the presence of such unpredictable faults.

1.1.10 Orientation of the Robots

In a multi-robot system, each individual robot ('r') has its own distinct unit of measurement for length and establishes its local cartesian coordinate system. This local system defines the directions and orientations of coordinate axes, all centered around the robot's position. When there is no presumption of uniformity or consistency among these local coordinate systems, we refer to it as "disorientation." In this context, it means that the coordinate systems of different robots can be significantly dissimilar. To understand the level of agreement or consistency among these local coordinate systems, we can identify various forms:

Global Consistency: This represents the strongest form of agreement. It occurs when all robots within the system unanimously concur on the directions and orientations of all coordinate axes, effectively making their coordinate systems identical.

k-Axes Agreement: This is an another form of consensus. It means that all robots agree on at least 'k' axes (where 'k' is a positive integer, less than the total number of axes, denoted as 'd') in terms of their directions and orientations.

Additionally, there is a concept called "Chirality", which refers to the robots' alignment on a cyclic orientation, such as choosing to follow either a clockwise or counterclockwise direction in their coordinate systems. It's worth noting that even when global geometric agreement exists, there may still be differences among the robots in terms of the unit of length they use to measure distances. This means that they might agree on directions and orientations, but they might employ different units for measuring lengths.

However, in scenarios where all robots have the same visibility radius, as seen in the unit-disc model of limited visibility, they do manage to reach an agreement on the unit of length. This demonstrates that under specific conditions, a consensus on the unit of length can be achieved, despite potential variations in other aspects.

1.1.11 Multiplicity Detection

In the context of robots residing on the same point or vertex, the concept of multiplicity arises. There are four distinct types of multiplicity detection, each designed to limit the capabilities of the robots as much as possible. Here are definitions for these four types:

Global Weak Multiplicity Detection Ability: A robot possesses global weak multiplicity detection ability when it can determine whether a multiplicity (i.e., multiple robots occupying the same point) exists at any given point or vertex in the system. This ability allows the robot to identify the presence of multiplicity globally across the entire system.

Global Strong Multiplicity Detection Ability: A robot is said to have global strong multiplicity detection ability when it can accurately compute the precise number of robots that make up a multiplicity at any given point or vertex in the system. This means it can determine the exact count of robots in a multiplicity globally across the system.

Local-Weak Multiplicity Detection Ability: A robot exhibits local-weak multiplicity detection ability when it can detect the presence of a multiplicity at

a specific point or vertex only if it itself is part of that multiplicity. In other words, it can identify multiplicity locally but only if it is directly involved in it.

Local-Strong Multiplicity Detection Ability: A robot possesses local-strong multiplicity detection ability when it can accurately calculate the precise number of robots forming a multiplicity at a particular point or vertex only if it is part of that multiplicity. It can determine the exact count of robots in a multiplicity but only within its local context.

These definitions describe various levels of multiplicity detection capabilities, ranging from the ability to detect multiplicity globally across the entire system to the more limited abilities restricted to local observations and participation in the multiplicity.

1.2 Related Literature

In this section, we present a brief survey of theoretical studies on the computational and complexity issues related to distributed computing by robot swarms. The Arbitrary pattern formation (Apf) is one of the fundamental problem in distributed computing. This problem has been studied in various settings. In the Euclidean plane, this problem was first studied by Suzuki and Yamashita [51]. They provided a complete characterization of the class of pattern formable in FSync and SSync for anonymous robots with unbounded memory. Then Flochhini [35] studied the cases of solvability of this problem under various assumptions. They showed that without a common coordinate system Apf problem is not solvable, but when there are both axes-agreement, the problem can be solved. Furthermore, with one axis agreement, any odd number of robots can form an arbitrary pattern, but an even number of robots cannot in the worst case. Later in [53] they characterized the families of pattern formable by oblivious robots in FSync and SSync. In [29] authors have established a relationship between Leader Election and Arbitrary Pattern Formation of robots

under asynchronous scheduler. Later they also showed that the arbitrary pattern formation is possible to solve when $n \geq 4$ with chirality (resp. $n \geq 5$ without chirality) if and only if leader election is solvable. In [15] authors consider the arbitrary pattern formation problem with four robots in the asynchronous model with or without chirality. In [11] authors have solved the APF problem with inaccurate movement where the formed pattern is very similar to the target pattern, but not exactly the same. A randomized pattern formation problem was studied in [14]. In [19] authors have shown some configurations where embedded pattern formation is solvable without chirality and some configuration where embedded pattern formation are deterministically unsolvable.

For grid network the arbitrary pattern formation was first studied by [10] in $\mathcal{OBLOT}$ model with full visibility. Later in [17] authors studied the APF on a regular tessellation graph. In [39] they provide randomized algorithm with full visibility and oblivious robots and with obstructed visibility and $\mathcal{LUMI}$ robots, both APF algorithms are time and move optimal.

Another interesting direction of solving this problem is when visibility is limited. Yamauchi in their paper [54] first showed that oblivious robots under FSYNC model with limited visibility can not solve APF. Therefore, they considered non-oblivious robots with unlimited memory. For these robots, they presented algorithms that work in FSYNC with non-rigid movements and in SSYNC with rigid movements. After that in [43], the authors have solved this problem in an infinite grid under 2 hop visibility. The problem was studied in a synchronous setting for robots with constant-size memory, where the robots agree on a common coordinate system.

A special case of formation problem is mutual visibility problem [1] and gathering problem [49]. In mutual visibility, robots are opaque so the main task is to form a configuration where no three robots are co-linear. Recently APF problem was solved in euclidean plane in [12] with opaque robots and in [9] with fat robots considering the luminous model. In an infinite grid, the arbitrary pattern formation problem was studied in [41] with opaque robots and in [42] with fat robots. The work in [52] solves APF in the obstructed visibility model without any

agreement in the coordinate system, where they showed that the run time to solve APF is bounded above by the time required to elect a leader. Another special case of formation problem, Uniform circle formation is investigated in [2, 31]. Das *et al.* solved the problem of forming a sequence of patterns in a given order ([25]). Further, they extended the sequence of pattern formation problems for luminous robots in [24]. There are many works ([18, 36]) where the pattern can be formed by robots with multiplicities. Pattern formation in the presence of faulty robots is an important topic of research.

In distributed system there are huge research studied with faulty robots. In [3] authors studied the gathering of robots with the presence of crash and byzantine faulty robots. After that gathering of robots with the presence of fault are studied in different domains also. In [26] authors provided the gathering of robots with fault in a ring. In [8] they introduced the gathering on a grid with faulty robots. Pattern formation in the presence of faulty robots is first introduced by [47]. In [47] they studied the formation of patterns allowing crash fault robots on plane.

But in all these works they only consider that all robots perform one individual task. But in [6] authors first showed that two specific but different tasks can be done by robots simultaneously on a plane. They showed that gathering and circle formation can be done by oblivious robots in a plane simultaneously with two different teams of robots using one-axis agreement. In [17], authors considered multiplicity points in the target pattern that needs to be formed. So their work is also capable of forming an arbitrary pattern along with an additional multiplicity point. But by following their algorithm a robot on team gathering might end up forming a pattern and also a robot on team arbitrary pattern formation might end up on the multiplicity point.

In distributed computing, the GATHERING of mobile robots has been the focus of intensive investigation under various computational power and communication varieties. This problem has been thoroughly studied both in continuous and discrete domains. There are few surveys on this problem [7,20,33]. In the continuous case, both the gathering and the rendezvous problems have been investigated in

the context of swarm of autonomous mobile robots operating in one and two-dimensional spaces. In these works, robots either gather or converge to a single point. Here we mainly focus on the gathering of robots when a robot has limited visibility. In [4] authors showed that without chirality and any agreement on the local coordinate system, robots can gather (resp. converge) under FSYNC (SSYNC) scheduler. Flocchini *et al.*, in their paper [34], provided an algorithm solving the gathering problem by robots in a plane under limited visibility and ASYNC scheduler. They have assumed that all robots agree on the orientation of both coordinate axes. After that in [28] authors provided an algorithm solving the gathering under limited visibility and a tight run-time analysis of the algorithm. In [48], authors provided a time optimal algorithm for the gathering of robots on a plane under limited visibility under asynchronous scheduler.

In the discrete case, the robots are dispersed in a network modeled as a graph and are required to gather at a single node and terminate. The main difficulty in solving the problem is symmetry. In the paper [38], authors showed that, the gathering of asynchronous oblivious robots in a bipartite graph with a local vision (i.e a robot can see its immediate neighbor only). Also, in [21], authors have provided an algorithm of *rendezvous* in an arbitrary graph. Another interesting way of research in swarm robotics is to investigate if a problem is robust against faulty robots. In [13], authors studied the gathering of robots in a network with byzantine robots. Also in [3, 27], authors provided gathering algorithms considering the crash and byzantine faults. In recent days, researchers started to investigate dynamic graphs [16] also. In [44], Di Luna *et al.* started the investigation of gathering in dynamic rings. In [22, 45] the gathering of robots in polygon terrain has been studied. In [37] an optimal gathering algorithm for robots on a triangular grid with 1-hop visibility under semi synchronous scheduler has been provided. There is huge research work on the gathering of robots on grid networks [5, 23].

There are some recent works where robots are on a continuous circle. In [30, 40], authors showed an algorithm solving the *rendezvous* of mobile agents with different speeds in a cycle. In [32], authors start the investigation of solving

GATHER and ELECT by the set of robots R deployed in a continuous cycle $\mathcal{C}$, they primarily focus on ELECT. But in their model, robots have no visibility i.e they can only see if they are at the same location. After that Di Luna *et al.* [46], prove that with limited visibility π, gathering of anonymous and oblivious robots is possible in SSYNC scheduler but it is impossible to gather all robots when visibility is less than or equals to $\frac{\pi}{2}$ on continuous circle $\mathcal{C}$.

1.3 Overview of the Thesis

The central focus of this thesis revolves around addressing the challenges posed by two critical problems in the realm of distributed computing by robot swarms: the "ARBITRARY PATTERN FORMATION" problem and the "GATHERING" problem. These problems stand at the core of research within this field, with numerous existing solutions, each built upon varying assumptions. However, the primary objective of this thesis is to tackle these distributed problems while making minimal assumptions.

Specifically, our aim is to develop algorithms that can effectively operate in asynchronous settings, which is a notable departure from many existing approaches. Furthermore, we prioritize the minimization of both the number of robot movements and the time required to achieve the desired outcomes. Another critical aspect of our research involves considering the notion of visibility. In a specific study, we delve into the challenges posed by limited visibility and examine its significant impact on the feasibility of solving various tasks using robots. This investigation underscores the inherent difficulties associated with limited visibility when attempting to accomplish tasks within the context of robot swarm coordination.

In Chapter 2 and Chapter 3, we study the move optimal and time optimal ARBITRARY PATTERN FORMATION on a two dimensional infinite rectangular grid. In the standard continuous setting, the robots can move in any direction and by any amount. On an infinite grid, the movements of the robots are restricted only along grid lines. The infinite grid setting, however, allows the robots to have a

partial agreement on coordinate system as they can align the axes of their local coordinate systems along the grid lines. In Chapter 4, we consider the Gathering and Arbitrary Pattern Formation of oblivious robots on an infinite rectangular grid by two teams of robots. Here a robot does not know the task of another robots. Then in Chapter 5, we consider the gathering of robots on a continuous circle. Here we consider robots have limited visibility. Finally in Chapter 6, we discuss some directions for future research.

Chapter 2

Move Optimal Arbitrary Pattern Formations on Infinite Grid by Oblivious robots

In distributed systems robot swarm coordination problems are being studied for the last two decades. Arbitrary Pattern Formation (Apf) is a fundamental coordination problem for autonomous robot swarms. The goal of this problem is to design a distributed algorithm that guides the robots to form any specific but arbitrary pattern given to the robots as an input. In this context, the main research difficulties are which patterns can be formed and how they can be formed. In the Euclidean plane, robots can move in any direction for a very small amount of distance, but it is not always possible for robots with weak capabilities to move accurately. So it is interesting to consider this type of problem in grid terrain, where robot movement is restricted in-between grid points. In practical applications, the interest has shifted to using a large number of simple robots which are easy to design and deploy, and have minimal capabilities to make the system cost-effective.

Leader election is an important task for the pattern formation problem, where a unique robot is elected as a leader (See section 2.4.1). In [10] an arbitrary pattern formation algorithm is given in an infinite rectangular grid under asynchronous scheduler considering $\mathcal{OBLOT}$ model. We aim to solve the arbitrary pattern

formation problem in an infinite rectangular grid by a swarm of robots with the optimal number of moves under a fully asynchronous scheduler. This work proposes an algorithm that solves the APF problem in an infinite rectangular grid with the optimal number of robot moves in $\mathcal{OBLOT}$ model. Our work shows that the algorithm proposed in [10] is not move optimal asymptotically.

2.1 Robot Model

Classical oblivious Robots: In the first problem the $\mathcal{OBLOT}$ model is considered for the robots. In this model, robots are anonymous, identical, and oblivious, i.e. they have no memory of their past rounds. They can not communicate with each other. All robots are initially in distinct positions on the grid. The robots can see the entire grid and all other robots' positions which means they have global visibility. This implies the robots are transparent and hence visibility of a robot can not be obstructed by another robots (this is an assumption for simplifying the problem). Robots have no access to any common global coordinate system. They have no common notion of chirality or direction. A robot has its local view and it can calculate the positions of other robots with respect to its local coordinate system with the origin at its own position. There is no agreement on the grid about which line is x or y-axis and also about the positive or negative direction of the axes. As the robots can see the entire grid, they will set the axes of their local coordinate systems along the grid lines.

Look-Compute-Move cycles: An active robot operates according to the Look-Compute-Move cycle. In each cycle a robot takes a snapshot of the positions of the other robots according to its own local coordinate system (Look); based on this snapshot, it executes a deterministic algorithm to determine whether to stay put or to move to an adjacent grid point (Compute); and based on the algorithm the robot either remain stationary or makes a move to an adjacent grid point (Move). When the robots are oblivious they have no memory of past configurations and previous actions. After completing each Look-Compute-Move cycle,

the contents in each robot's local memory are deleted.

Scheduler: We assume that robots are controlled by a fully asynchronous adversarial scheduler (ASync). The robots are activated independently and each robot executes its cycles independently. This implies the amount of time spent in Look, Compute, Move and inactive states is finite but unbounded, unpredictable and not same for different robots. The robots have no common notion of time.

Movement: In discrete domains, the movements of robots are assumed to be instantaneous. This implies that the robots are always seen on grid points, not on edges. However, in our work, we do not need this assumption. In the first proposed move optimal algorithm, we assume the movements are to be instantaneous for simplicity. However, this algorithm also works without this assumption. In that case, the robots are asked to wait and do nothing if they see a robot on a grid edge. In the second proposed time-optimal algorithm no such assumption is required. That is, if a robot sees any robot on an edge, it still does its job as directed by the algorithm. The movement of the robots is restricted from one grid point to one of its four neighboring grid points.

Measuring Run-time: Generally, time is measured in rounds in fully synchronous settings. But as robots can stay inactive for an indeterminate time in semi-synchronous and asynchronous models, epochs are considered instead of rounds. During an epoch, it is assumed that all robots are activated at least once. Here in the second algorithm, we calculate the run-time with respect to epochs.

Next, we describe the Arbitrary Pattern Formation problem in an infinite rectangular grid in the next section.

2.2 Problem description

Problem Statement

Suppose a swarm of robots is placed in an infinite rectangular grid such that

no two robots are on the same grid node and the configuration formed by the robots is asymmetric. The Arbitrary Pattern Formation (APF) problem asks to design a distributed algorithm following which the robots autonomously can form any arbitrary but specific pattern, which is provided to the robots as an input, without scaling it or colliding with another robot.

Let's discuss some facts about the problem. The input target pattern can be any arbitrary pattern. The required distributed algorithm has to be independent of the input target pattern, that is, the same algorithm should work for any target input pattern. The whole target pattern is given to all the robots but no robot knows its target position beforehand. The target pattern can be provided to the robots with respect to some coordinate system but the robots initially do not agree on any coordinate system. Robots are not allowed to scale the pattern configuration but are allowed to transform or rotate it. From the motivation from [50] we assume that the initial configuration of robots is asymmetric. But the target pattern can be any pattern (can possibly be symmetric). The algorithm solving APF must take care that no two robots collide.

In the next two section, we provide an algorithm that solves the Arbitrary Pattern Formation problem in $\mathcal{OBLOT}$ model.

2.3 Our Contribution:

In this work, our goal is to solve APF problem under the full visibility model optimally in terms of energy. Energy is measured by the number of moves made by the robots required to solve the problem. First, we define the input size of the problem. Let k be the number of robots in the input configuration. Suppose the dimensions of the smallest enclosing rectangles of the initial configuration of the robots and the pattern to be formed are respectively $m \times n$ and $M \times N$. Let $\mathcal{D} = \max\{m, n, M, N\}$ and $\mathcal{D}' = \max\{\mathcal{D}, k\}$.

In [10] authors provided an algorithm solving APF problem in $\mathcal{OBLOT}$ model. But the solution is not optimal in terms of energy because the algorithm proposed in [10] needs all total $O(k\mathcal{D}^2) = O(\mathcal{D}'^3)$ moves. Also in [10] authors showed that

any algorithm solving APF requires $\Omega(\mathcal{D}'^2)$ total moves. In this work, we propose an algorithm that solves the APF problem in $\mathcal{OBLOT}$ model which requires $O(\mathcal{D}'^2)$ total robot moves, which is asymptotically optimal.

2.4 Optimal move APF Algorithm (APFOpt-Move)

2.4.1 Agreement on global co-ordinate system

Here we consider an infinite rectangular grid G as a cartesian product $P \times P$, where P is an infinite (from both sides) path graph. The infinite grid G is embedded in the Cartesian Plane R^2. We know that the solvability of Arbitrary pattern formation depends on the symmetries of the initial configuration of the robots. Here we are assuming that the initial configuration is asymmetric. The robots do not have an access to any global coordinate system even though each robot can form a local coordinate system aligning the axes along the grid lines. To form the target pattern the robots need to reach an agreement on a global coordinate system. This subsection provides details of the procedure that allows the robots to reach an agreement on a global coordinate system.

For a given configuration ($\mathcal{C}$) formed by the robots, let smallest enclosing rectangle, denoted by $s.rect(\mathcal{C})$, is the smallest grid aligned rectangle which contains all the robots. Suppose the $s.rect$ of the initial configuration $\mathcal{C}_I$ is a rectangle $\mathcal{R} = ABCD$ of size $m \times n$, such that $m > n > 1$. Let $|AB| = n$. Then consider the binary string $\{s_i\}$ associated with a corner A, λ_{AB} as follows. Scan the grid from A along the side AB to B and sequentially all grid lines of $s.rect(\mathcal{C}_\mathcal{I})$ parallel to AB in the same direction. And $s_i = 0$, if the position is unoccupied and $s_i = 1$ otherwise (fig 2.1). Similarly construct the other binary strings λ_{BA}, λ_{CD} and λ_{DC} (here we consider only the smaller sides of the $s.rect(\mathcal{C})$). Since the initial configuration is asymmetric we can find a unique lexicographically largest string. If λ_{AB} is the lexicographically largest string, then A is called as the leading corner of $\mathcal{R}$.

Next, suppose $\mathcal{R}$ is an $m \times m$ square, then consider the eight binary strings λ_{AB}, λ_{BA}, λ_{CD}, λ_{DC}, λ_{BC}, λ_{CB}, λ_{AD}, λ_{DA}. Again since initial configuration is asymmetric, we can find a unique lexicographically largest string among them. Hence we can find a leading corner here as well.

Next, let $\mathcal{C}_I$ be a line AB, we will have two strings λ_{AB} and λ_{BA}. Since $\mathcal{C}_I$ is asymmetric then λ_{AB} and λ_{BA} must be distinct. If λ_{AB} is lexicographically larger than λ_{BA}, then we choose A as the leading corner.

Now for either case, if λ_{AB} is the lexicographically largest string then the leading corner A is considered as the origin, and the $x-$ axis is taken along the AB line. If $\mathcal{C}_I$ is not a line then the $y-$ axis is taken along the AD line. If λ_{AB} is a line then the $y-$ coordinate of all the positions of robots is going to be zero and in this case, the $y-$ axis will be determined later.

For any given asymmetric configuration $\mathcal{C}$ if λ_{AB} is the largest associated binary string to $\mathcal{C}$ then the robot causing first non zero entry in λ_{AB} is called *head* let $\mathcal{H}$ and the robot causing last non zero entry in λ_{AB} is called as *tail* let $\mathcal{T}$. We denote the i^{th} robot of the λ_{AB} string as r_{i-1}. A robot other than head and tail is called ***Inner robot***. Further we denote $\mathcal{C}' = \mathcal{C} \setminus \{tail\}$ and $\mathcal{C}'' = \mathcal{C} \setminus \{tail, head\}$ and $\mathcal{C}''' = \mathcal{C} \setminus \{Head\}$, Let $\mathcal{C}_T$ be the target configuration and $s.rect(\mathcal{C}_T) = \mathcal{R}_T$. Let $\mathcal{R}_T$ is a rectangle of size $M \times N$ with $M \geq N$. We can calculate the binary strings associated with corners in the same manner as previous. $\mathcal{C}_T$ is expressed in the coordinate system with respect to the origin, where origin will be the leading corner. Let the $A'B'C'D'$ be the smallest rectangle enclosing the target pattern with $A'B' \leq B'C'$. Let $\lambda_{A'B'}$ be the largest (may not be unique) among all other strings. Then the target pattern is to be formed such that $A = A'$, $A'B'$ direction is along the positive x axis and $A'D'$ direction is along the positive y axis. If target pattern have symmetry then we have to choose any one among the larger string and fixed the coordinate system. So as previously said $head_{target}$ will be the first one and $tail_{target}$ will be the last one in the $s.rect$ of $\mathcal{C}_\mathcal{T}$. Also we define $\mathcal{C}_\mathcal{T}' = \mathcal{C}_\mathcal{T} \setminus \{tail_{target}\}$, $\mathcal{C}_\mathcal{T}'' = \mathcal{C}_\mathcal{T} \setminus \{head_{target}, tail_{target}\}$, $\mathcal{C}_\mathcal{T}''' = \mathcal{C}_\mathcal{T} \setminus \{head_{target}\}$. We denote the $head_{target}$ position as t_0 and $tail_{target}$ position as t_{k-1}. Let H_i be the horizontal line having the height i from x-axis. Let for each i there are $p(i)$

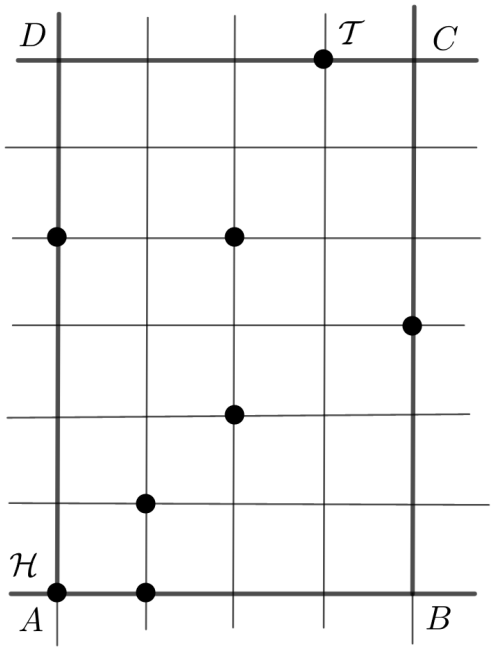

Figure 2.1: Here the binary string of AB is 11000010000010000001101000000000010. Similarly by calculating the associated strings of BA, CD and DC (only smaller sides) and comparing in lexicographically order, AB is the largest string. $\mathcal{H}$ and $\mathcal{T}$ are head and tail respectively.(as described in section 2.4.1)

target positions on H_i. We denote the target positions of H_0 as t_0, t_1.....$t_{p(0)-1}$ from left to right. For H_1 we denote the target positions as $t_{p(0)}$ to $t_{p(0)+p(1)-1}$ from right to left. For H_2 we denote the target positions as $t_{p(0)+p(1)}$ to $t_{p(0)+p(1)+p(2)-1}$ from right to left. Similarly we can denote all other target positions on H_i, $i > 0$ except $tail_{target}$. Next we give some relevant definitions.

Definition 2.1 (Inner robot)**.** *A robot that is not head or tail in a configuration is called an inner robot.*

Definition 2.2 (Compact Line)**.** *A line is called compact if there is no empty grid point between two robots.*

2.4.2 A brief outline of APFOptMove algorithm

In this algorithm, our goal is to make Apf with move optimal. For this, robots initially form a line and then form the arbitrary pattern that is given as input. We can divide our algorithm into eight phases. Since the initial configuration is asymmetric, the robot can agree on a global coordinate system. So robots can recognize where the pattern can be formed. Here robots have to maintain the coordinate system during their movements. When a robot wakes up, it can be in any phase among the eight phases. So, as the robots are oblivious they can check by their condition in which phase it is currently in. The conditions are expressed in the Boolean function listed in Table 2.1. As the movement of a robot is restricted in the discrete domain, here we have to maintain the movement without collision throughout the algorithm. As maintaining the asymmetry is another main challenge of our algorithm, in the first three phases the head will be put at the origin and the tail will expand to the smallest enclosing rectangle for another robot can move but the head and tail remain unchanged and asymmetry also maintained. In phase four robots form a line on the x-axis without the tail and one inner robot of $\mathcal{C}'$. In phase five all robots move to the fixed target position sequentially without a head and tail. In the last three phases, the head and tail will reach their fixed target positions. During these phases movement of robots are difficult as here the coordinate system may change. But here we showed that asymmetry and global coordinate will be maintained in all phases. In this way, arbitrary pattern formation can be done.

2.4.3 Detailed description of the eight phases

Phase 1: A robot is in phase 1 then tail will move upwards and all other robots will remain static. When in phase 1 $\neg\{S_1 \wedge S_2\} \wedge \neg\{S_3 \wedge S_4\}$ is true.

Theorem 2.1. *If we have an asymmetric configuration $\mathcal{C}$ at some time t, then by phase 1*

S_0	$\mathcal{C} = \mathcal{C}_T$
S_1	$\mathcal{C}' = \mathcal{C}'_T$
S_2	x-coordinate of the tail in $\mathcal{C} = x$ Coordinate of t_{target} in $\mathcal{C}_T$
S_3	$m \geq \max\{N, n\} + 2$
S_4	$m \geq 2.\max\{M, V\}$ where v is the length of the vertical side of the smallest enclosing rectangle of $\mathcal{C}'$
S_5	The head in $\mathcal{C}$ is at the origin.
S_6	$n \geq \max\{N + 1, H + 1, k\}$ where H is the length of the horizontal side of the smallest enclosing rectangle of $\mathcal{C}'$
S_7	$\mathcal{C}'' = \mathcal{C}''_T$
S_8	$\mathcal{C}'$ has a non-trivial reflectional symmetry with respect to a vertical line.
S_9	$\mathcal{C}''' = \mathcal{C}'''_T$.
S_{10}	Line formation on x-axis without tail and one inner robot.

Table 2.1: The Boolean variable on the left is true if and only if the condition on the right is satisfied.

- *After one move upward, the new configuration is still asymmetric and the coordinate system remains unchanged.*

- *after one move by the tail upwards* $\neg\{S_1 \wedge S_2\} = true$

Proof. Let $ABCD$ be the initial smallest enclosing rectangle at any time t, and let the binary string associated with AB be the largest with $|AB| = n$ and $|AD| = m$, $m \geq n$. So initially the tail is on the side CD. But after the tail moves upward the smallest enclosing rectangle changes. Let the new rectangle is $ABC'D'$. Here tail $\mathcal{T}$ is now on the side $C'D'$. Now let $\mathcal{T}$ is the only robot initially on side CD , then it is obvious that $\lambda_{AB}^{new} > \lambda_{BA}^{new}$. But if there are multiple robots on CD then let t is the p^{th} and q^{th} term of λ_{AB}^{old} and λ_{BA}^{old}. Then,

Case-1: When $p = q$ then t is the middle robot of CD, here p^{th} term is the last 1 occurs in λ_{AB}^{old} so if we calculate the binary string of first $(p-1)$ term of AB and BA in the new $s.rect$ then also we get $\lambda_{AB}^{new} > \lambda_{BA}^{new}$.

Case-2: When $p > q$ then if we calculate the binary strings of λ_{AB}^{old} and λ_{BA}^{old} then $\mathcal{T}$ will be appear earlier in BA, than AB. Now if we calculate the binary strings of first q term of AB and BA then $\lambda_{BA}^{old}|_q > \lambda_{BA}^{new}|_q$. Also $\lambda_{AB}^{old}|_q > \lambda_{AB}^{new}|_q$. But $\lambda_{AB}^{old} > \lambda_{BA}^{old}$. So we have $\lambda_{AB}^{old}|_q > \lambda_{BA}^{old}|_q$. Finally we can say $\lambda_{AB}^{new}|_q > \lambda_{BA}^{new}|_q$, so $\lambda_{AB}^{new} > \lambda_{BA}^{new}$.

Case-3: When $p < q$ in that case when $\mathcal{T}$ robot move upward then in the new $s.rect$ we can calculate that in this case also $\lambda_{AB}^{new} > \lambda_{BA}^{new}$.

So in all the cases $\lambda_{AB}^{new} > \lambda_{BA}^{new}$. Now we show that the binary string of AB is larger than $C'D'$. As $\mathcal{T}$ is the tail so we know that the binary string of AB is larger than CD, but when the tail robot moves one step upward in that case as there is no robot other than the tail in $C'D'$ so if we calculate binary string it will be $\lambda_{AB}^{new} > \lambda_{C'D'}^{new}$.
In this case, $ABC'D'$ is a non-square grid, so four binary strings to consider here. By calculating we can say that AB is the largest binary string in this new smallest enclosing rectangle. So the new configuration is still asymmetric. So the coordinate system is unchanged. As the tail moves upward so the x-coordinate remains unchanged, so $\neg\{S_1 \wedge S_2\}$ remains the same after one move by the tail robot. □

Phase 2: In this phase head $\mathcal{H}$ will move left towards the origin. When the algorithm is in phase 2 then either $S_3 \wedge S_4 \wedge \neg S_5 \wedge \neg S_7$ or $\neg S_2 \wedge S_3 \wedge S_4 \wedge \neg S_5 \wedge S_7$ is true.

Theorem 2.2. *If we have an asymmetric configuration $\mathcal{C}$ at time t in phase 2 then*

1. *after one move by the head robot to the left, the new configuration is still asymmetric and the coordination system is not changed.*

2. *after a finite number of moves by the head to the left S_5 is true and phase 2 terminates.*

Proof. As $ABCD$ is the smallest enclosing rectangle of all robots, let λ_{AB} be the lexicographically largest string, after one move of the head to the left, now also λ_{AB} is the largest string. Let at the i^{th} term the first 1 occurs in λ_{AB}, then in the other strings all the $(i-1)$ terms are 0. But when the head moves one distance to the left then the new string form is now the largest. So the new configuration is asymmetric and the global coordinate will not change. So (1) is true, similarly by the finite number of moves head move to the origin then S_5 true. □

Phase 3: The goal of this phase is to make S_6 true. In this phase the robots will check either S_8 is true or false. When algorithm is in phase 3 then $S_3 \wedge S_4 \wedge S_5 \wedge \neg S_6 \wedge \neg S_7 =$ true. When S_8 is false, then the tail will move rightwards and the rest will remain static. But when S_8 is true, the $\mathcal{C}'$ has a nontrivial reflectional symmetry with respect to a vertical line V.

Let the smallest enclosing rectangle is $\mathcal{R} = ABCD$ where $|AB| = n$ and $|AD| = m$, $m > n$. Let λ_{AB} be the lexicographically largest string. In this case tail will move right and after finite number of move we have $S_3 \wedge S_4 \wedge S_5 \wedge S_6 \wedge \neg S_7 =$ true.

Theorem 2.3. *If we have an asymmetric configuration $\mathcal{C}$ at some time t then in phase 3*

1. *after one move rightward the new configuration is still asymmetric and the global coordinate system remain unchanged.*
2. *after one move $S_4 \wedge S_5 \wedge \neg S_7$ = true.*
3. *after finite number of moves by the tail to the rightwards $S_3 \wedge S_4 \wedge S_5 \wedge S_6 \wedge \neg S_7$ = true.*

Proof. Let the smallest enclosing circle at time t is $ABCD$, where λ_{AB} is the largest string. After one move by the tail rightward there may arise two cases.

Case-1: Suppose now tail robot is at C, then by one move of tail the new $s.rect$ is $APQD$, where $|AP| = (n+1)$. Now it is easy to check that as $m \geq n+2$ so

$m > n + 1$. So we get that $AD > AP$. This implies that the new configuration is still not square, so we have to consider here only four binary strings, and as earlier λ_{AP} will be a larger string. So we can conclude that the configuration is still asymmetric and the coordinate system is not changed by one move of the tail. It is easy to check that S_4 and S_5 are true here but not S_7. After the movement of the tail, S_3 may become false, so we are in phase 1 then. Then the tail moves upwards and one upwards move still has $S_3 \wedge S_4 \wedge S_5 \wedge S_6 \wedge \neg S_7$ is true.

Case-2: Let after one move by the tail the smallest enclosing circle remain unchanged. As in this phase, the head is in origin and the tail has moved until S_4 true, in the binary string of CD or DC is smaller than AB. Also in this phase, S_8 is not true. So we must have AB larger string than BA, so finally we get $\lambda_{AB}^{new} > \lambda_{BA}^{new}$. Note that S_8 is either true or false in phase 3 by a finite number of moves of the tail the configuration remains asymmetric.

If S_8 is true then there may happen two cases:

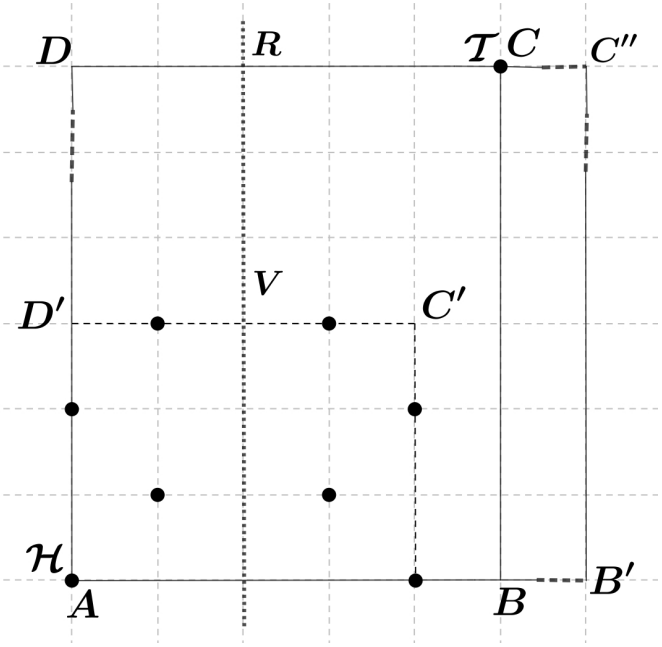

Figure 2.2: Case-1: $\mathcal{C}'$ has a vertical symmetry and tail will move rightwards

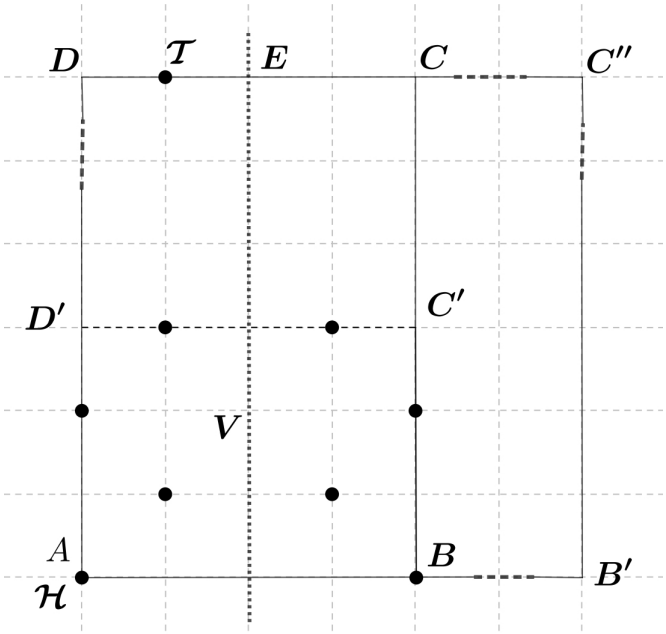

Figure 2.3: Case-2: $\mathcal{C}'$ has a vertical symmetry and tail will move leftwards

Case-1: There is a vertical symmetry in $\mathcal{C}'$ but if we consider $ABCD$ then the tail is rightward of vertical symmetry line V. Then tail will move rightwards and after a finite number of moves, S_6 is true (fig 2.2).

Case-2: When the tail is in the left portion with respect to the vertical symmetric line V. Then tail will not move rightwards (fig 2.3). It will move to the left one more step than D, then the coordinate system will be changed. B will be then origin and x-axis = BA and y-axis = BC. Then the case will be the same as case-1.

□

Phase 4: In this phase the configuration satisfies $S_3 \wedge S_4 \wedge S_5 \wedge S_6 \wedge \neg S_7 \wedge \neg S_{10}$ = true. Other than the tail and one inner robot, all other inner robots form a line on the x-axis. In phase four, the head is in origin, and all the robots on the

x-axis first make the line compact, i.e. there is no empty grid point between two robots. After the x-axis becomes compact, when a robot r_i is on H_i and there are no robots in between H_i and the x-axis and the right part of r_i is empty in its horizontal line, then the robot moves to the x-axis. This procedure is done one by one by robots. In between this movement, no collision will occur. Finally when a robot sees that except for itself and tail all other inner robots are on the x-axis then it will not move to the x-axis (fig 2.4). So all the inner robots other than the tail and one inner robot form a line on the x-axis.

Theorem 2.4. *If we have an asymmetric configuration $\mathcal{C}$ such that $S_3 \wedge S_4 \wedge S_5 \wedge S_6 \wedge \neg S_7 \neg S_{10}$ = true then in phase 4 after finite number of moves by the robots S_{10} becomes true and phase 4 terminates.*

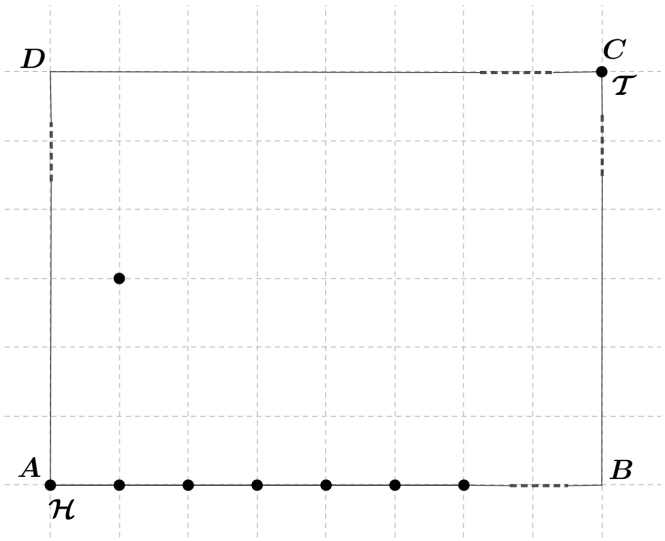

Figure 2.4: Line formation of robots on x axis without tail and last inner robot.

Proof. In previous phases when the tail robot expands the smallest enclosing rectangle and the head robot moves to the origin, then in this phase robots that are on the x-axis make the line compact. In this move, a robot will move to its left grid point if it is empty, so the robot's movement is in the left direction. So collision will not occur. A robot r_i which is on a horizontal line (let H_k) first

checks that all the down lines robots are on the x-axis or not, if yes then when a robot sees that the right side of its horizontal line has also no robots, it will move to the x-axis. In this way, robots move down to the x-axis one by one. So the movement of the robot is sequential here. A robot will not move until it sees that it is the rightmost robot in its horizontal line and there is no more robot in the down horizontal lines other than the x-axis. As the $s.rect$ is expanded by the tail robot in the previous phases, a robot always gets a path in the grid and moves to the x-axis. So collision will not occur in this movement. Finally without the tail and one inner robot, after a time all other robots will form a line on the x-axis. Hence S_{10} is true. □

Phase 5: In this phase, inner robots will move to the fixed target one by one. When one inner robot sees that all other robots without itself and the tail are on the x-axis then it moves to t_{k-2}. We call this inner robot as *Last inner robot*. When a robot on the x-axis sees that the *Last inner robot* is at its target position then i^{th} robot from the left on the x-axis will reach to t_i target position when it sees that from t_{i+1} to t_{k-2} positions are occupied (fig 2.5).

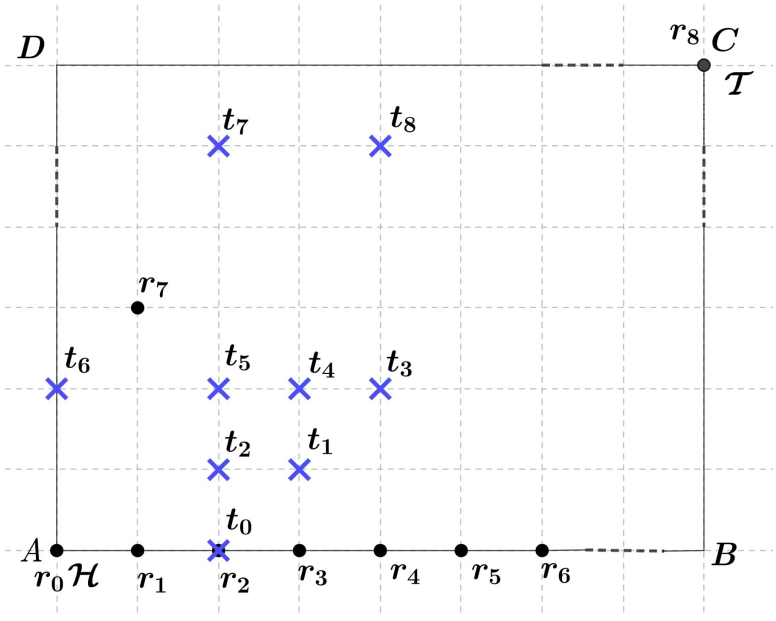

Figure 2.5: Target formation without head and tail.

Theorem 2.5. *If we have an asymmetric configuration initially at some time t then in phase 5*

1. *By movement of inner robots the new configuration is still asymmetric and the coordinate system remains the same.*
2. *After any move of the inner robots in this phase $\mathcal{C}'' = \mathcal{C}''_T$ and phase 5 terminate and S_7 becomes true.*

Proof. As the head is in origin and the tail robot expands the *s.rect* of the initial configuration, so by the movement of inner robots, the coordinate system will not change and the configuration remains asymmetric. Here our main concern is collision. In this phase an inner robot r_i moves to target position t_i when t_{i+1} to t_{k-2} positions are occupied by robots. As of last inner robot moves to t_{k-2} at the start of this phase. Let us denote the robots on the x-axis from left to right as $r_0, r_1, \ldots, r_{k-3}$. Then firstly r_{k-3} reaches t_{k-3}, then r_{k-4} reaches t_{k-4} and so on. Finally, r_1 moves to t_1. As the movement in this phase is sequential that is no other robot moves until one moving robot reaches its target position. Also since the target positions are ordered in such a way that every inner robot will find a unblock path to reach its target position. So here no collision will occur. Eventually, each inner robot reaches its target position. So finally S_7 is true.

□

Phase 6: The tail will move to the left until the x-coordinate matches with the $tail_{target}$. In this phase, the tail will move to make S_2 true.

Theorem 2.6. *If we have an asymmetric configuration $\mathcal{C}$ at some time t then after a finite number of moves of robots, phase 6 terminates and S_2 becomes true.*

Proof. In phase 6 if we have an asymmetric configuration, then depending on whether S_8 is true or not there are two cases. Let the smallest enclosing rectangle of $\mathcal{C}$ be $ABCD$ where A is the origin and $AB = x$-axis and $AD = y$-axis. Let

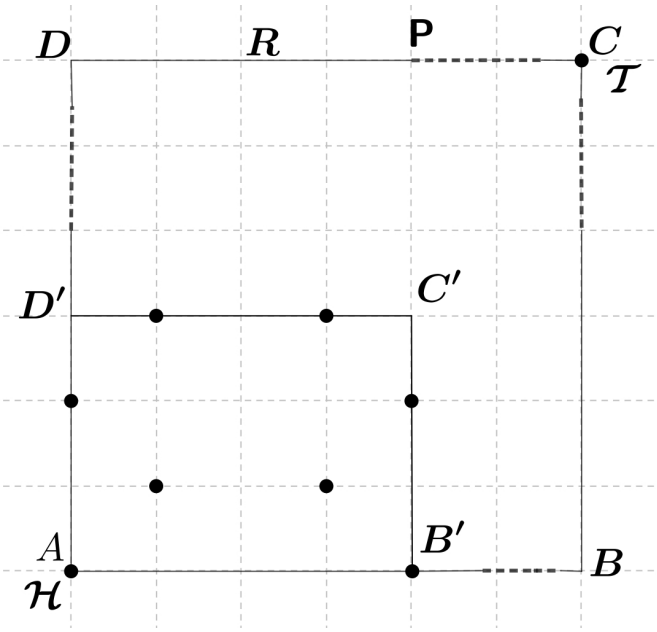

Figure 2.6: When S_8 is true in phase 6

without tail the smallest enclosing rectangle is $AB'C'D'$. In this phase without tail, all the robots are now on $AB'C'D'$.

Case-1: Let S_8 is not true i.e. there is no symmetry in $\mathcal{C}'$. Then the tail robot will move on the CD side, no matter where $tail_{target}$ is AB will be the lexicographically largest string. Finally tail will move up to S_2 will true.

Case-2: Let S_8 is true (fig 2.6). Since there is symmetry in $\mathcal{C}'$, here the head robot is in A. Let P be the point of intersection between $B'C'$ and CD. In this case when the tail robot moves left in the line CD, then it will move up to that point whose x-coordinate is the same as $tail_{target}$. In this move by the tail when it will reach a point let R which is in between P and D then a vertical symmetry will be created. The tail robot's destination can not be [P,R] because then B' will be the head. So when the tail crosses R there will be a vertical symmetry. In this case also S_1 hold. When the tail robot moves left in CD then in other cases coordinate system remains invariant and S_2 holds.

So in both cases when phase 6 terminates then $S_2 \wedge S_3 \wedge S_4 \wedge S_5 \wedge S_7$ true. □

Phase 7: The aim of this phase is that Head will moves to $head_{target}$ (fig 2.7). Consider a configuration that is asymmetric and in phase 7, then with respect to the global coordinate system as fixed, let $ABCD$ is the smallest enclosing rectangle and λ_{AB} be the lexicographically largest string. Clearly head is on the side AB and the tail is on CD. If we mark all the target points on the grid then let the smallest rectangle be $AB'C'D$. Let $head_{target}$ be the final position of head, then by finite move head will move to $head_{target}$.

Theorem 2.7. *Let $\mathcal{C}$ be the asymmetric configuration at time t in phase 7 then by a finite number of moves by the head to the right, phase 7 terminate when $\neg S_0 \wedge S_1 \wedge S_2 \wedge S_9$ = true.*

Proof. Let $ABCD$ be the smallest enclosing circle of an asymmetric configuration $\mathcal{C}$ of phase 7, Here AB is the lexicographically largest string. Now we have to plot the smallest enclosing rectangle of the target pattern with respect to our current coordinate system. This phase aims to move the head robot from the origin A to its fixed target position, which will be in the right direction of A on AB. As there may be vertical symmetry in target configuration when the head moves to its target then also vertical symmetry will happen. So in all the cases, AB will be a lexicographically larger string when the head robot moves to its target position. So when the head reaches its final position phase 7 terminates and $\neg S_0 \wedge S_1 \wedge S_2 \wedge S_9$ is true. □

Phase 8: Tail moves downwards up to $tail_{target}$. In this phase the position of tail will be upward of $tail_{target}$. So when in phase 8 tail will move downwards to the point $tail_{target}$ (fig 2.7). So when in phase 8 then $\neg S_0 \wedge S_1 \wedge S_2$ = true, but after tail move then S_0 is true.

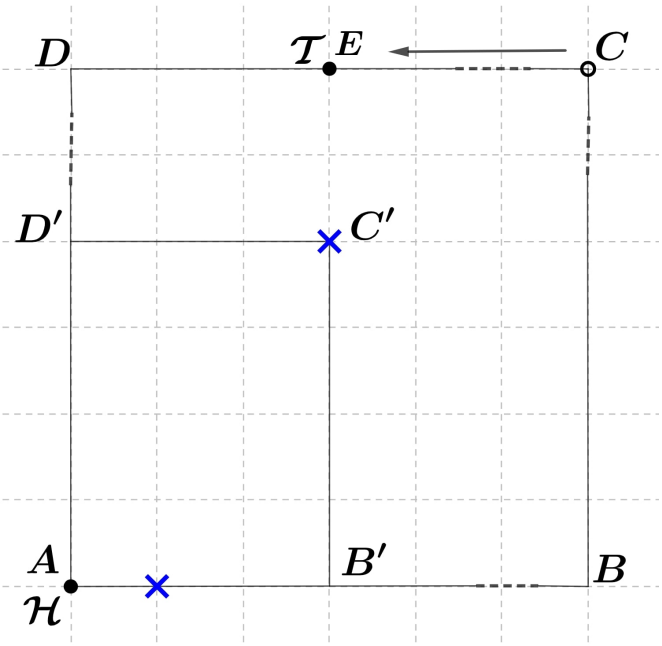

Figure 2.7: In phase 7 head move to it's fixed target and in phase 8 tail move to its target point.

Theorem 2.8. *If we have a configuration $\mathcal{C}$ at some time t then in phase 8 after a finite number of moves by the tail, S_0 becomes true.*

Proof. In this phase by a finite number of moves by the tail robot, the arbitrary pattern given as input will form. After phase 6 may be the configuration has symmetry or not.

1. Let the configuration is asymmetric. Then the tail robot is now on the CD side, where $ABCD$ is the smallest enclosing rectangle. Let $AB'C'D'$ be the $s.rect$ of the target configuration. Then $tail_{target}$ will be in the downwards vertical line of the tail. So when the tail robot moves down to its target position AB will be always a lexicographically larger string.

2. Let the configuration is asymmetric but the position of $tail_{target}$ is on the upper side or in its horizontal line or downside. This is only possible when the initial configuration is the same as the target without the tail's position.

In this case, the tail will move to its position. No symmetry will occur during this move of the tail.

3. Let the configuration is symmetric. As the initial configuration is asymmetric the symmetry may arise in phase 7, so when there is a vertical symmetry then let $ABCD$ be the $s.rect$ and AB and BA be the larger string, as S_4 is true here so the target position for tail will be the downside of its recent position and after it moves and reaches to its $tail_{target}$ then S_0 is true.

□

In figure 2.8 we presented the algorithm APFOPTMOVE by a flow chart. Starting from any asymmetric configuration where S_0 is not true, we can observe that the path ends, where S_0 is true, passing through a finite number of phases. So we can conclude that.

Theorem 2.9. *The* APFOPTMOVE *solves the Arbitrary pattern formation problem within finite time in* $\mathcal{OBLOT}$ *model.*

2.4.4 Move Complexity of the algorithm

Suppose the dimensions of the smallest enclosing rectangles of the initial configuration of the robots and the pattern to be formed are respectively $m \times n$ and $M \times N$. In [10] author proved that any algorithm solving the arbitrary pattern formation problem in an infinite grid requires $\Omega(k\mathcal{D})$ moves, where $\mathcal{D} = \max\{m, n, M, N\}$ that is, $\mathcal{D} = \max\{AB, CD, A'B', C'D'\}$ and, k is the number of robots. Let $\mathcal{D}' = \max\{k, \mathcal{D}\}$. Then the mentioned result in [10] becomes that any algorithm solving the arbitrary pattern formation problem in an infinite grid requires $\Omega(\mathcal{D}'^2)$ moves. We show that our algorithm requires $O(\mathcal{D}')$ moves for each robot. Which gives the total number of required movements is $O(k\mathcal{D}') = O(\mathcal{D}'^2)$, which is asymptotically optimal. In Phase 1, Phase 2, and Phase 3 of our algorithm, only one robot moves. So a robot participating in these phases uses $O(\mathcal{D}')$ move. Then in phase 4 a robot comes down on the x-axis to

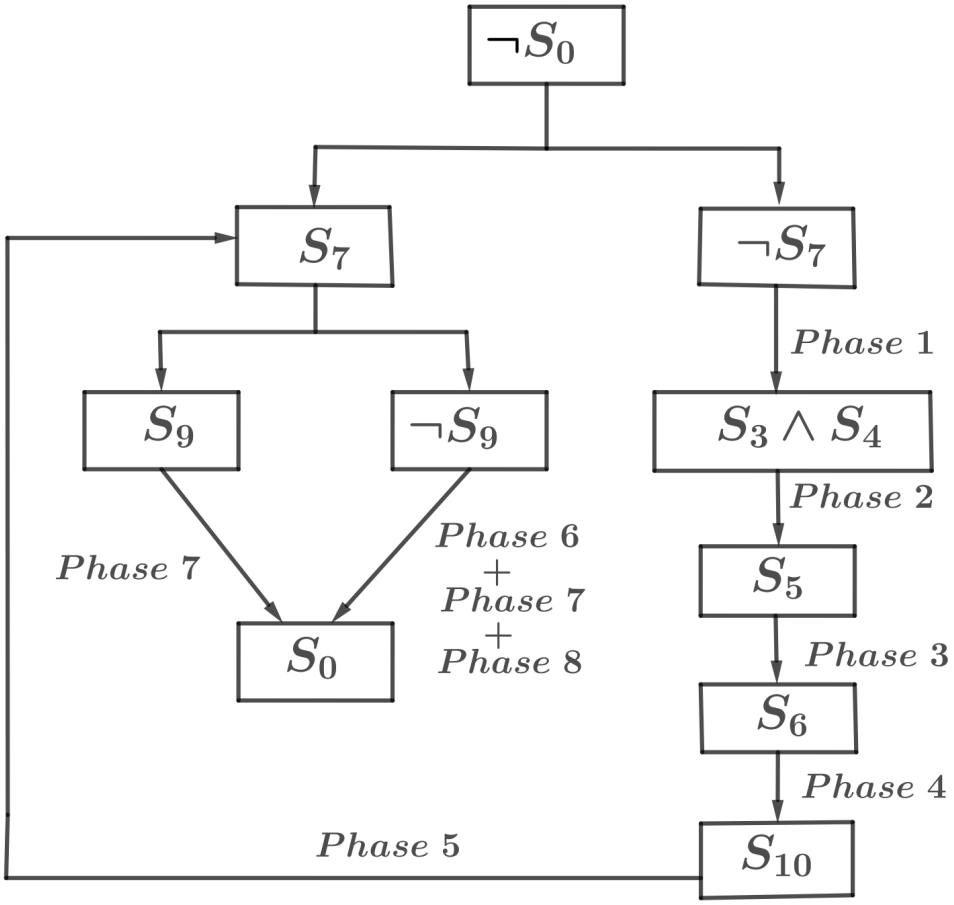

Figure 2.8: Flow Chart of the Algorithm APFOptMove

form a compact line and in phase 5 a robot reaches its target position from the x-axis, so here also a robot in total has to move at most $4\mathcal{D}$ steps. In phase 6, phase 7, and phase 8 only one robot moves, and that robot needs to make $O(\mathcal{D}')$ moves. So we can conclude that any phase of our algorithm requires $O(\mathcal{D}')$ moves for a robot, which is asymptotically optimal. Since starting from any asymmetric configuration, our algorithm terminates via a finite number of phases among 8 phases, so we can finally conclude the following theorem.

Theorem 2.10. *Arbitrary pattern formation is solvable by optimal $O({\mathcal{D}'}^2)$ moves in an asynchronous scheduler by oblivious robots from any asymmetric configuration.*

Now we observe that the time complexity of the APFOptMove is $O(\mathcal{D}'^2)$. We observed in the proof of Theorem 2.10 that each robot makes $O(\mathcal{D}')$ moves. Hence in worst case for each robot $O(\mathcal{D}')$ epochs are sufficient. Since the movements of robots are sequential in this algorithm, the algorithm must terminate within $O(k\mathcal{D}') = O(\mathcal{D}'^2)$ epochs. We record this result in following Theorem.

Theorem 2.11. *The* APFOptMove *algorithm solves the* Apf *problem within* $O(\mathcal{D}'^2)$ *epoch time under asynchronous scheduler.*

2.5 Concluding Remarks

This chapter studies algorithm for Arbitrary Pattern Formation (Apf) problem on an infinite rectangular grid by a robot swarm starting from any asymmetric initial configuration in classical $\mathcal{OBLOT}$ robotic model. This work gives an algorithm for the Apf problem in $\mathcal{OBLOT}$ model which is asymptotically move optimal. Suppose the dimensions of the smallest enclosing rectangles of the initial configuration of the robots and the pattern to be formed are respectively $m \times n$ and $M \times N$. If $\mathcal{D}'$ is the maximum of the number of robots and $\mathcal{D}$ where $\mathcal{D} = \max\{m, n, M, N\}$, then the algorithm uses total $O(\mathcal{D}'^2)$ movements to solve the Apf problem. Unfortunately the move optimal algorithm in $\mathcal{OBLOT}$ model is not time optimal, so studying for an algorithm in $\mathcal{OBLOT}$, which is both move optimal and time optimal, could be a possible direction from this work. If there are more than one robot in a node of a network then that point known as Multiplicity point. One can consider another version of Apf problem where the initial or the target configuration of robots can have more than one robot in one position.

Chapter 3

Time and Move Optimal Arbitrary Pattern Formations on Infinite Grid by Luminous Robots

We aim to solve the arbitrary pattern formation problem in an infinite rectangular grid by a swarm of robots with the optimal number of moves and within optimal time under a fully asynchronous scheduler. First, in chapter 2 proposes an algorithm that solves the APF problem in an infinite rectangular grid with the optimal number of robot moves in $\mathcal{OBLOT}$ model but not time optimal. Furthermore, we propose another algorithm for solving APF problem on an infinite rectangular grid considering in $\mathcal{LUMI}$ model for robots which is both move and time optimal. This work also shows that APF problem can be solved faster than the algorithm proposed in [10] by introducing communicable memory.

3.1 Robot Model

Robots with lights: In this problem the $\mathcal{LUMI}$ model has been considered. In this model, the robots are anonymous and identical and they have constant memory (finite number of lights). Each robot has a light that can assume one color at a time from a constant number of different colors. All the other assumptions

are the same as the classical oblivious robots model.

Look-Compute-Move cycles: An active robot operates according to the Look-Compute-Move cycle. In each cycle a robot takes a snapshot of the positions of the other robots according to its own local coordinate system (Look); based on this snapshot, it executes a deterministic algorithm to determine whether to stay put or to move to an adjacent grid point (Compute); and based on the algorithm the robot either remain stationary or makes a move to an adjacent grid point (Move). When each robot is equipped with an externally visible light, which can assume a $O(1)$ number of predefined colors, the robots communicate with each other using these lights. The lights are not deleted at the end of a cycle. In this algorithm we use one light which takes `off`, `head` and `line` colours.

Scheduler: We assume that robots are controlled by a fully asynchronous adversarial scheduler (ASync). The robots are activated independently and each robot executes its cycles independently. This implies the amount of time spent in Look, Compute, Move and inactive states is finite but unbounded, unpredictable and not same for different robots. The robots have no common notion of time.

Movement: In discrete domains, the movements of robots are assumed to be instantaneous. This implies that the robots are always seen on grid points, not on edges. However, in our work, we do not need this assumption. In the time-optimal algorithm, robots movement may be instantaneous or not. No such assumption is required. That is, if a robot sees any robot on an edge, it still does its job as directed by the algorithm. The movement of the robots is restricted from one grid point to one of its four neighboring grid points.

Measuring Run-time: Generally, time is measured in rounds in fully synchronous settings. But as robots can stay inactive for an indeterminate time in semi-synchronous and asynchronous models, epochs are considered instead of rounds. During an epoch, it is assumed that all robots are activated at least once.

Here in the algorithm, we calculate the run-time with respect to epochs.

Paper	**Robot Model**	**Visibility Model**	**No. of Colours**	**Move Complexity**	**Time Complexity**
[10]	$\mathcal{OBLOT}$	Unobstructed	-	$O(\mathcal{D}'^3)$	$\Omega(\mathcal{D}'^2)$[Res.3.1]
[42]	$\mathcal{LUMI}$	Obstructed	9	-	-
APFOptMove	$\mathcal{OBLOT}$	Unobstructed	-	$O(\mathcal{D}'^2)$[Th.2.10] (**optimal**)	$O(\mathcal{D}'^2)$[Th.2.11]
FastAPF	$\mathcal{LUMI}$	Unobstructed	3	$O(\mathcal{D}'^2)$[Th.3.10] (**optimal**)	$O(\mathcal{D}')$[Th.3.9] (**optimal**)

Table 3.1: Comparison Table.

3.2 Our Contribution:

In this work, our goal is to solve Apf problem under the full visibility model optimally in terms of time. Time can be measured by the total number of epochs required to solve the problem. First, we define the input size of the problem. Let k be the number of robots in the input configuration and suppose the dimensions of the smallest enclosing rectangles of the initial configuration of the robots and the pattern to be formed are respectively $m \times n$ and $M \times N$. Let $\mathcal{D} = \max\{m, n, M, N\}$ and $\mathcal{D}' = \max\{\mathcal{D}, k\}$.

In this work, we propose algorithm that solves Apf in $\mathcal{LUMI}$ model. This algorithm requires $O(\mathcal{D}')$ epoch time which is faster than the algorithm proposed in [10]. This also establishes that our algorithm is asymptotically time optimal in $\mathcal{LUMI}$ model. Although we cannot tell that the algorithm proposed in [10] is not time optimal because it is done in $\mathcal{OBLOT}$ which is a weaker model than $\mathcal{LUMI}$. Further, we also show that the algorithm is move optimal as well.

Although all above-mentioned works in this subsection are done under the full visibility model, in [42] authors solve Apf under obstructed view considering fat robots. However, authors did not do any complexity analysis. Authors in [42] used 9 colors to solve the problem where our algorithm uses only 3 colors. The

above discussion is briefly presented in Table 3.2.

Paper	**Robot Model**	**Visibility Model**	**No. of Colours**	**Move Complexity**	**Time Complexity**
[10]	$\mathcal{OBLOT}$	Unobstructed	-	$O(\mathcal{D}'^3)$	$\Omega(\mathcal{D}'^2)$[Res.3.1]
[42]	$\mathcal{LUMI}$	Obstructed	9	-	-
APFOptMove	$\mathcal{OBLOT}$	Unobstructed	-	$O(\mathcal{D}'^2)$[Th.2.10] (**optimal**)	$O(\mathcal{D}'^2)$[Th.2.11]
FastAPF	$\mathcal{LUMI}$	Unobstructed	3	$O(\mathcal{D}'^2)$[Th.3.10] (**optimal**)	$O(\mathcal{D}')$[Th.3.9] (**optimal**)

Table 3.2: Comparison Table.

3.3 Optimal time APF Algorithm (FastAPF)

This section proposes an algorithm, named FastAPF for Apf which is time optimal and move optimal as well. Before going to the algorithm, for convenience let go through some definitions, notions supporting the algorithm. Let $ABCD$ be the unique smallest rectangle enclosing a given initial configuration, where $AB \geq BC$. If $ABCD$ is not square then consider the set of strings S to be $\{\lambda_{AB}, \lambda_{BA}, \lambda_{CD}, \lambda_{DC}\}$. If $ABCD$ is square then consider the set of strings S to be $\{\lambda_{AB}, \lambda_{BA}, \lambda_{CD}, \lambda_{DC}, \lambda_{BC}, \lambda_{CB}, \lambda_{AD}, \lambda_{DA}\}$.

3.3.1 Coordinate System setup

If the given configuration is asymmetric then S contains a lexicographically strictly largest string. Let λ_{AB} be the largest among other strings in S. Then robots consider A as the origin and AB direction as the positive x axis and AD direction as the positive y axis. In such a case, the first robot in λ_{AB} string is said to be *head*.

Each robot has a light. This light can take two colors, namely, `head` and `line` which are readable as well as communicable. The light can indicate another state when the light is off. We denote this one as `off` color.

Next suppose in a given configuration there is a robot, call it *head*, with `head` color on the boundary of $ABCD$. There are two cases, firstly if the head is not situated in any corner and another when the head is at a corner of the $ABCD$, say A. For the first case we assume that the head is on the side AB, such that $AB \geq CD$. For such a case consider A as origin, AB direction as positive x axis, and AD direction as positive y axis. For the second case, consider the head robot is situated at a corner, say A. In such a case consider A as the origin. If $AB > BC$ then consider AB direction as positive x axis and AD direction as positive y axis. If $AB = BC$, then we assume the configuration is asymmetric and hence there is a lexicographically strictly largest string, say λ_{AB} is S. In such a case consider the AB direction as a positive x axis and AD direction as a positive y axis.

Definition 3.1 (Tail robot)**.**
Case-I: (When there exist no robot with **`head`** *color) In this case we assume that the configuration is asymmetric. The last robot in the lexicographically strictly largest string in S is said to be Tail.*
Case-II: (When there exist a robot with **`head`** *color) In this case the tail is the rightmost robot of the topmost horizontal line.*

Let the $A'B'C'D'$ be the smallest rectangle enclosing the target pattern with $A'B' \geq B'C'$. Let $\lambda_{A'B'}$ be the largest (may not be unique) among all other strings in S for the target pattern. We denote the i^{th} target position in $\lambda_{A'B'}$ string as t_i. Then the target pattern is to be formed such that $A = A'$, $A'B'$ direction is along the positive x axis and $A'D'$ direction is along the positive y axis. $head_{target}$ will be the first one and $tail_{target}$ will be the last one in the $A'B'C'D'$.

Let $s = \max\{AD, A'D'\}$. Let name the lines parallel and above to x axis by $H_1, H_1, \ldots, H_s$. Name the vertical lines from left to right as $V_1, V_2, \ldots,$ where V_1 is the y-axis. Let at any time t the configuration be $C(t)$. Let in the target pattern the number of robots in H_i line be n_i. Let in $C(t)$ the total number of robots below the line H_i be b_i. Let in $C(t)$ the total number of robots above the line H_i be a_i. Let $b'_i = \sum_{j<i} n_i$ and $a'_i = \sum_{j>i} n_i$. A horizontal line H_i is said to

be *saturated* if $a_i = a'_i$ and $b_i = b'_i$ In Figure 3.1 H_5 is a saturated line.

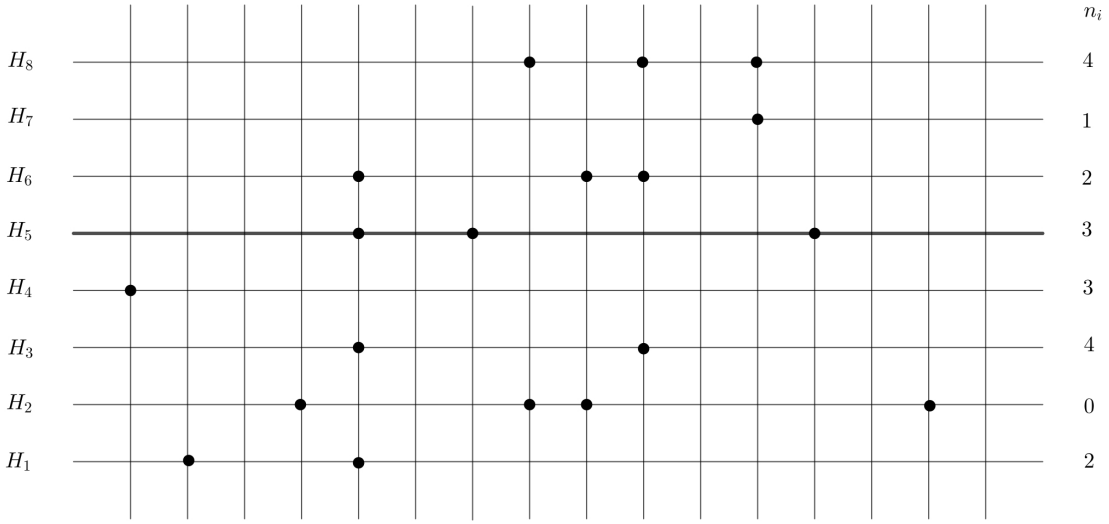

Figure 3.1: In this example the configuration has one Saturated line. In the target pattern the number of robots in H_i line be n_i. Here observe that 3 robots are initially in H_5 and also $n_5 = 3$, so H_5 is a saturated line

The next subsection describes all the intermediate procedures of FastAPF.

3.3.2 Elements of the algorithm

In order to optimize the time, our main motive is to make the algorithm so that it allows parallel movement of robot as much as possible avoiding collision. The algorithm is divided into six procedures. Initially, since the configuration is asymmetric, a robot that activates at first can find out the things listed in Table 3.3.

1	Smallest enclosing rectangle $ABCD$ with $AB \geq BC$
2	If the configuration is asymmetric, lexicographically largest string λ_{AB}
3	Head robot
4	Tail robot

Table 3.3

Throughout the algorithm, the head robot remains head using `head` color as the flag. And further, the coordinate system remains unaltered throughout the algorithm that has been taken care of. Once there exist a robot with `head` color and hence the head robot is fixed. Any robot other than the head or tail is said to be *inner* robot. Next, we define terminologies for different configurations.

1. $\mathcal{C}_{init}$ = Initial configuration
2. $\mathcal{C}_{target}$ = Target configuration
3. $\mathcal{C}$ = A possible configuration from initial configuration
4. $\mathcal{C}' = \mathcal{C} \setminus \{Head\}$
5. $\mathcal{C}'' = \mathcal{C} \setminus \{Tail\}$
6. $\mathcal{C}''' = \mathcal{C} \setminus \{Head, Tail\}$
7. $\mathcal{C}'_{target} = \mathcal{C}_{target} \setminus \{Head\}$
8. $\mathcal{C}''_{target} = \mathcal{C}_{target} \setminus \{Tail\}$
9. $\mathcal{C}'''_{target} = \mathcal{C}_{target} \setminus \{Head, Tail\}$

Next we define different conditions on configuration.

C_0	$\mathcal{C} = \mathcal{C}_{target}$
C_1	$\mathcal{C}' = \mathcal{C}'_{target}$
C_2	$\mathcal{C}'' = \mathcal{C}''_{target}$
C_3	$\mathcal{C}''' = \mathcal{C}'''_{target}$
C_4	a robot with `head` color.
C_5	Head color is at corner.
C_6	All inner robots are with `line` color except those who are on a saturated line.
C_7	Tail is at a point with x-coordinate $\max\{AB + 1, A'B' + 1, k\}$ and y-coordinate $\max\{BC, B'C'\}$.

Table 3.4: Boolean functions

3.3.3 Detailed description and Correctness

The algorithm is described here by six procedures and we will discuss these procedures in details.

1. **Procedure-I:** *Input*: $\neg C_3 \vee (C_3 \wedge \neg C_1 \wedge \neg C_2)$

   In the first procedure, Head identifies itself and turns its `head` colour on. If the smallest enclosing rectangle of the initial configuration is a non square rectangle, then head robot goes to the origin if it is not. But if the initial configuration is square, then after the head robot turns on its `head` colour, the tail will move rightwards and make the configuration rectangle. After that, the head will move to the origin.

   Output: $C_4 \wedge C_5$ is true.

   Discussion This procedure gets executed when either there is an inner robot not at its target position or all inner robots are at their target but neither head nor tail is at its target position. In such a case, the head robot first turns on its `head` colour and then moves leftwards until it reaches its origin if the smallest enclosing rectangle of the initial configuration is a non square rectangle. But when the initial configuration is square, then head robot turns on `head` colour and then first tail robot moves right up to the configuration becoming a rectangle. After that the head robot moves left towards origin. Note that if the input configuration is asymmetric, then the configuration throughout the procedure remains asymmetric. And also, the things listed in Table 3.3 remain unchanged.

   Theorem 3.1. *If we have an asymmetric configuration $\mathcal{C}$ at some time t,*

   - *after one move by head robot leftward, the new configuration is still asymmetric and the coordinate system remains unchanged.*
   - *after finite move of robots by procedure-I, $C_4 \wedge C_5$ will be true.*

   Proof. When $\neg C_3 \vee (C_3 \wedge \neg C_1 \wedge \neg C_2)$ is true, then first the head robot will be selected. The head robot turns on its `head` color and moves left

towards the origin if the smallest enclosing rectangle of the configuration is a non square rectangle. Note that if the initial configuration is square, then after the head turns on its color, the tail will move rightwards and make the configuration rectangle. After that, the head will move to origin if it is not. There will be no symmetry as the head robot is with `head` colour on and the configuration becomes a rectangle after the tail robot moves rightwards. As the head robot turns on its `head` color and moves left, then the global coordinate will be maintained throughout the movement of head robot. Because λ_{AB} is the lexicographically largest string and head robot is the first robot of this string. So when the head robot moves left the new string will be larger than the previous one. So the global coordinate will not be changed and when it reaches the origin by finite number of moves, then $C_4 \wedge C_5$ will be true. □

2. **Procedure-II:** *Input*: $C_1 \wedge \neg C_2$

   In this procedure, the head moves to its target position and turns off its `head` color if it is on.

   Output: C_0 is true.

   Discussion This procedure gets executed when every robot except the head is at their respective target position. In this procedure, the head occupies its target position and turns off its `head` color if it is on. Now the head target must be on the x-axis, so either head needs to move right or left to reach its destination in this procedure. If the head needs to move left then clearly the things listed in Table 3.3 remain unchanged. Also in the other case, since the head target is the first target position of the lexicographically largest string of the target pattern, the listed things in Table 3.3 remain unchanged.

   Theorem 3.2. *If we have an asymmetric configuration $\mathcal{C}$ at some time t, then by finite move of head robot of procedure-II, C_0 is true.*

Proof. When all the robots are in their target positions without the head robot and the colour of the head robot is not on, then it will move to its target position as it will be on the x-axis and C_0 will be true. But if the `head` colour is on but head robot is not in its target position and all the other robot is in target position. Then the head robot will be at origin and its target position will be either at the origin or in the rightward direction of head robot. If the target of head robot is in origin then it will turn off its head colour and C_0 will be true. If the target is rightwards then head robot moves to its target and turns off its colour and C_0 will be true. In this procedure, only head robot moves and within the move, the robot is with `head` colour. So no collision and symmetry will occur within the move of head robot. So after finite move of head, the C_0 will be true. □

3. **Procedure-III:** *Input*: $(C_2 \wedge \neg C_1) \vee (C_4 \wedge C_5 \wedge C_3)$

   Case-I: If the y-coordinate of the tail target is the same as the y-coordinate of the tail, then the tail reaches at target by horizontal movements.

   Case-II: If the y-coordinate of the tail target is not the same as the y-coordinate of the tail, then we consider the following cases.

   Case-IIA: If the y-coordinate of the tail target is greater than the y-coordinate of the tail, then move upwards until Case-I is achieved.

   Case-IIB: If the y-coordinate of the tail target is less than the y-coordinate of the tail, move horizontally so that the x-coordinate of the tail target becomes the same as the x-coordinate of the tail. Then the tail moves downwards until condition C_0 is true.

   Output: $C_0 \vee C_1$ is true.

   Discussion This procedure gets executed when either the pattern except the tail is formed $(C_2 \wedge \neg C_1)$ or all inner robots are at their target position and the head is at the origin with head color $(C_4 \wedge C_5 \wedge C_3)$. For both cases, careful movement is assigned to the tail robot. For all the cases it can be checked that the things listed in Table 3.3 remain unchanged. If the input

configuration was $C_2 \wedge \neg C_1$, after execution of this procedure the target pattern is formed (C_0). And if the input pattern was $C_4 \wedge C_5 \wedge C_3$ then after execution of this procedure the resulting configuration is such that the target pattern is formed except head robot.

Theorem 3.3. *If we have an asymmetric configuration $\mathcal{C}$ at time t, then*

(a) by the movement of tail robot of procedure-III, the global coordinate will be maintained.

(b) there will be no collision in procedure-III and $C_0 \vee C_1$ will be true.

Proof. In this procedure, the tail robot will move to its target position. In case-I, the tail robot will move in its horizontal line. Within this move, there is head robot with `head` colour. So, if all the other robots are in their target position then by the move of tail in its horizontal line, C_0 will be true. In this procedure only one robot will move, so collision also does not occur and the coordinate system remains unchanged. In case-IIA, tail will move upward, and when the y-coordinate matches then move in the horizontal line. Here also, only the tail robot will move and there exists a head robot with `head` colour. So symmetry and collision will not occur and the global coordinate will be maintained. In the last case, when tail target is on the downside of the tail robot, the tail will first move in its horizontal line. In the path of tail's move, there will be no other robot, as in this procedure, all the other inner robots are in their target positions. As the embedding of target is done in such a way that there will be no other robot in the tail's move. When the x coordinate matches, then the tail moves downwards. So in all the cases, by the move of tail robot $C_0 \vee C_1$ will be true. □

4. **Procedure-IV:** *Input*: $C_4 \wedge C_5 \wedge \neg C_3 \wedge \neg C_7$

   In this procedure the tail moves right until its x-coordinate becomes $\max\{AB+1, A'B'+1, k\}$. Then the tail moves upward until its y-coordinate becomes $\max\{BC, B'C'\}$.

   Output: C_7 is true.

Discussion This procedure gets executed when at least one inner robot is not at its target position and the head robot is at its origin. In this procedure, the robot moves rightwards in order to ensure that there is enough big smallest rectangle of the current configuration for procedure-V and procedure-VI to execute and also to ensure that $AB > BC$ so that the coordinate system does not change.

Theorem 3.4. *If we have an asymmetric configuration $\mathcal{C}$ at time t, then in procedure-IV,*

(a) by the move of tail robot the global coordinate system will not changed.

(b) after finite move of tail robot, C_7 will be true.

Proof. This procedure will start when there exists a head robot with `head` colour at the origin. So the asymmetry will be maintained. Now also, here only the tail robot will move, so there will be no collision. Now the tail robot moves up to x- coordinate matches $\max\{AB+1, A'B'+1, k\}$ and y-coordinate becomes $\max\{BC, B'C'\}$. By this move of tail robot, $AB > BC$ will be maintained and also a robot with `head` colour is at origin. So the global coordinate will be maintained. So by the finite move of tail robot, C_7 will be true. □

5. **Procedure-V:** *Input*: $C_4 \wedge C_5 \wedge \neg C_3 \wedge C_7 \wedge \neg C_6$

   In this procedure, if an inner robot is not on a saturated horizontal line then it checks whether its `line` color is on or not. If the `line` color is not on, then it counts the number of robots below it. Let's say the number is b. Then the robot counts the number, say, l of robots on the left side to it in its horizontal line. Let $v = b + l + 1$. Then the robot tries to move to the v^{th} vertical line in its horizontal line. If v^{th} vertical line is in its left (right) and its left (right) grid point is empty then it moves to left (right). When a robot reaches the v^{th} vertical line, the robot turns on its `line` color.

   Output: C_6 is true (Figure 3.2).

Discussion These procedures are executed when the head robot is at the origin with its `head` color on and the tail robot's coordinate is $(\max\{AB+1, A'B'+1, k\}, \max\{BC, B'C'\})$ and there is at least one inner robot which is not on the saturated line and with no line color. In this procedure, such robot reaches at v^{th} vertical line and turns on its `line` color. At the end of this procedure in the configuration, the head and tail remain at their starting position and each inner robot is either with `line` color on or on a saturated line. In these procedures since no robot jumps through horizontal lines, so no collision of robots takes place.

Theorem 3.5. *If we have a configuration $\mathcal{C}$ at time t, then in procedure-V,*

(a) by the move of inner robots, coordinate system will be maintained.

(b) no collision will occur.

(c) after finite move by inner robots, C_6 will be true.

Proof. This procedure will start when head robot is at origin with `head` colour and tail at a position satisfying C_7 true. So by the movements of inner robots, co-ordinate system will not change. In this procedure, inner robots move in horizontal lines. In one horizontal line, robots will calculate their destination point and move left or right. In this move, if it sees that its destination is left or right, it will only move if it sees it's left or right is empty. Here, robots are numbered in such a way, that if a robot r wants to move to v^{th} vertical line and its destination is on the left, then the robots on the left of r's destination will be also left of v^{th} vertical line. So one by one robots on a particular horizontal line will move to its vertical line. So as the movement in each horizontal line is sequential, collision will not occur. After the move of robots, there will be one robot in each vertical line with colour `line`. So after finite moves of inner robots, C_6 will be true. □

6. **Procedure-VI:** *Input*: $C_4 \wedge C_5 \wedge \neg C_3 \wedge C_7 \wedge C_6$

   In this procedure, there are two exhaustive cases.

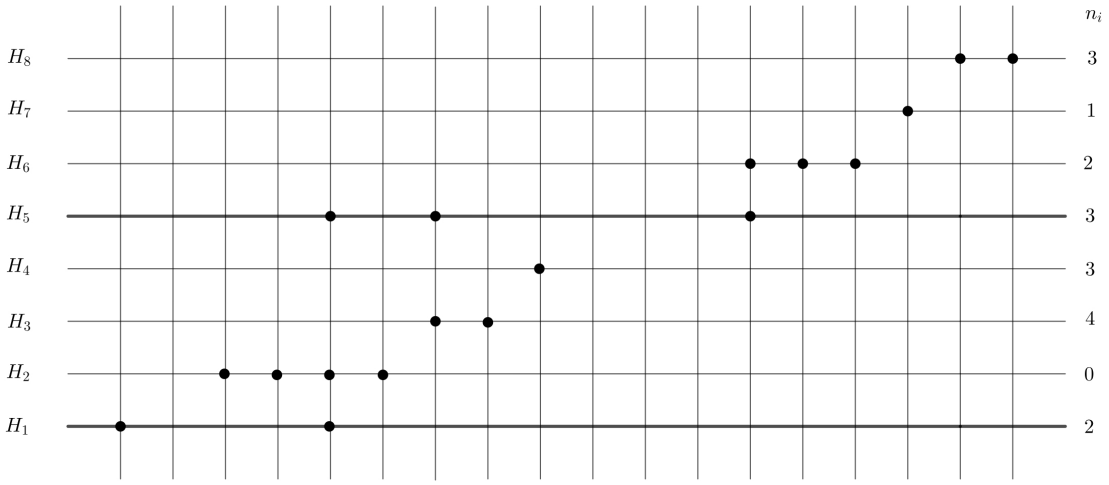

Figure 3.2: When C_6 true

Case-I: This is the case when the robot sees that its own horizontal line is not saturated. In such a case, the robot counts its vertical line number. Let the robot be at v_i^{th} vertical line. Then it moves to the horizontal line which contains the v_i^{th} target position by vertical movements until it reaches there (Figure 3.3).

Case-II: This is the case when the robot sees that its own horizontal line is saturated. In this case firstly if its `line` color is on, then it turns it `off`. Otherwise, if the robot is the k^{th} robot on its horizontal line from the left, then it tries to reach the k^{th} target position on that horizontal line. If k^{th} target position in its horizontal line is in its left (right) and its left (right) grid point is empty then it moves to left (right).

Output: C_3 is true.

Discussion In the input configuration of this procedure an inner robot is either on a saturated line or not on a saturated line but with its `line` color on. In this procedure, no two inner robots with `line` color on are on the same vertical line. In this procedure, two types of movements are happening simultaneously. Firstly robots with line color on but not on a saturated line do vertical movements and rests do horizontal movements. Eventually, all

robots will reach their destined horizontal line and all horizontal lines will become saturated. Next, we need to show no two robots collide in this procedure. We can guarantee that if no robot with `line` color reaches a saturated line. Now from the definition of a saturated line, one can note that every robot below a saturated line has its target horizontal line below the saturated line. And also every robot above a saturated line has its target horizontal line above the saturated line. Hence no robot making vertical movement will reach a saturated line where horizontal movement is possibly happening. Hence this procedure is collision-free.

Theorem 3.6. *If we have a configuration $\mathcal{C}$ at time t, then in procedure-VI,*

(a) by the move of inner robots global coordinate will be maintained and no collision will occur.

(b) by finite move of inner robots C_3 will be true.

Proof. In this procedure inner robots first move in its vertical line and after this move when it is in a saturated line, then it will move left or right and reach its target position. After procedure-V, there exists exactly one robot in each vertical line with `line` colour, or the robots are on saturated line. In case-I, when in each vertical line exactly one robot exists, then robot moves up or down by calculating its target position. As one robot will move in each vertical line, so collision will not occur. Another case is, when robots see that it is in a saturated line. In that case, robots similarly move left or right one by one sequentially. So collision will not happen. As within this move of inner robots, head and tail are at positions maintaining $C_4 \wedge C_5$ and C_7 true. So global coordinate will be maintained. After finite moves of inner robots, C_3 will be true and the procedure terminates. □

The next subsection formally presents the algorithm `FastAPF` in flow chart form.

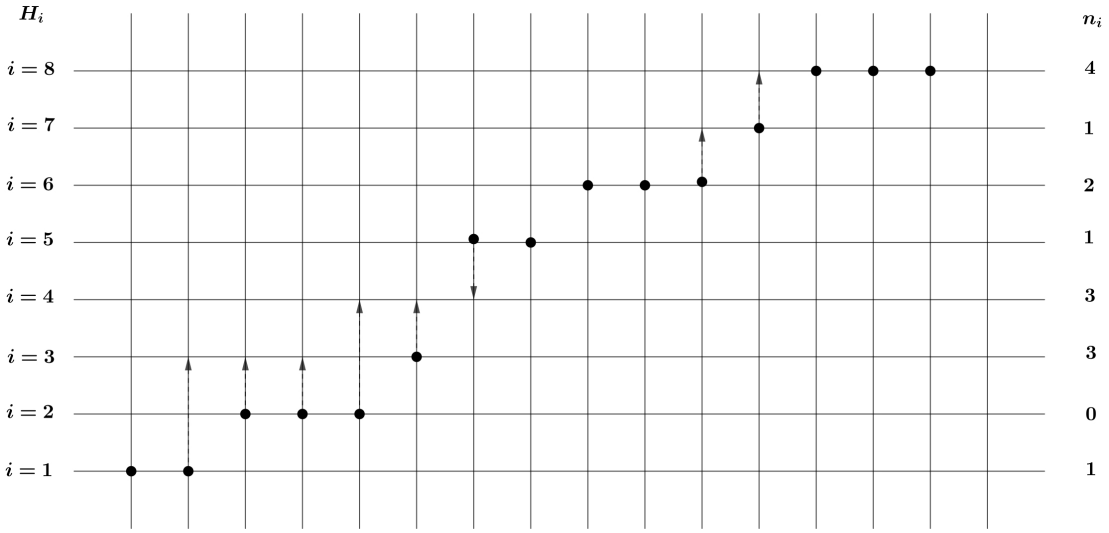

Figure 3.3: Procedure VI illustration

3.3.4 Algorithm FastAPF

The main algorithm FastAPF is presented below in the flow chart in Figure 3.4.

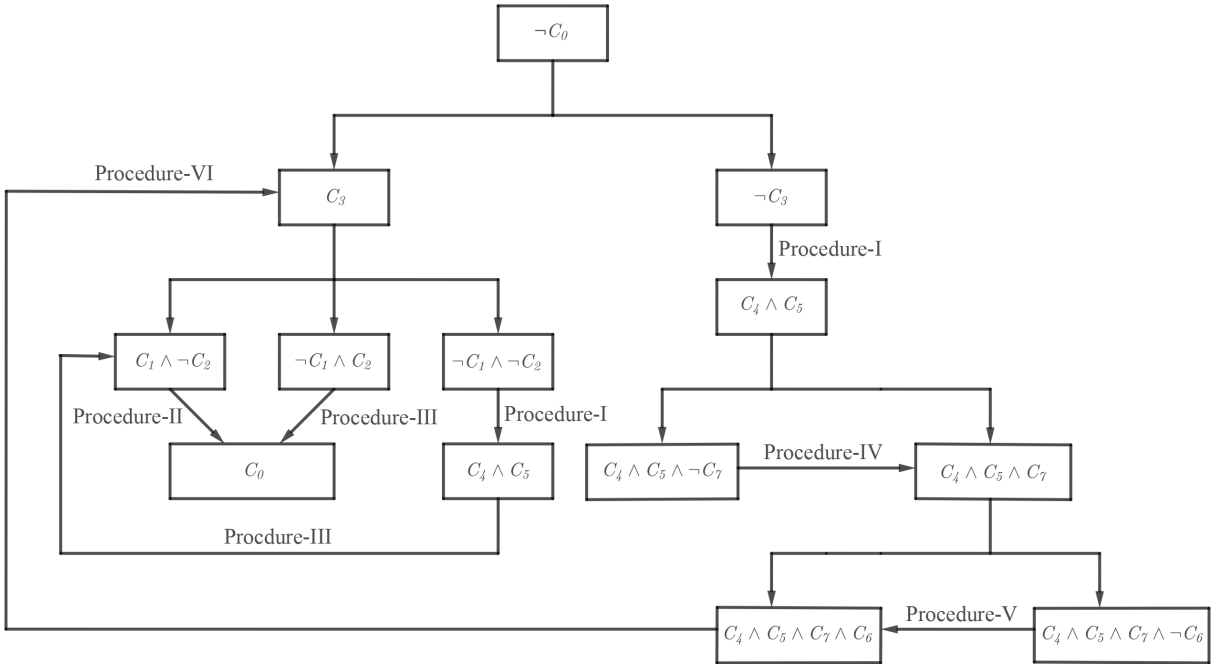

Figure 3.4: Algorithm FastAPF flow chart

Starting from any possible configuration where C_0 is not true, in the flow chart, we can observe that the path ends to the C_0 configuration passing through some procedures. Hence we conclude the following.

Theorem 3.7. *The* FastAPF *algorithm solves the* Apf *problem in* $\mathcal{LUMI}$ *model.*

3.3.5 Time Complexity and Movement analysis of FastAPF algorithm

We show that each of six procedures included in FastAPF algorithm takes $O(\mathcal{D}')$ epoch. The flow chart in the Figure 3.4 shows that starting from any asymmetric initial configuration the algorithm terminates via executing a finite number of these procedures. This will prove our claim that, the algorithm FastAPF solves Apf problem in $O(\mathcal{D}')$ epochs.

In Procedure-I, Procedure-II, Procedure-III, and Procedure-IV only one robot is making its move. And in each of these procedures, a robot does $O(\mathcal{D})$ moves, so all the procedures can take $\mathcal{D}$ epochs in the worst case to complete. In Procedure-V in each horizontal line, robots are making horizontal parallel movements. On a horizontal line H_i, for all robots to reach the destined vertical line maximum it will take $2\mathcal{D}'$ epochs. Hence Procedure-V takes at most $2\mathcal{D}'$ epochs to complete. In procedure-VI, a robot with `line` color makes vertical movements. All vertical movements in this procedure happen simultaneously. So all the vertical movements are done in $\max\{BC, B'C'\}$ epochs. Then in a saturated line horizontal movements happen in order to reach the target positions. Similarly, this also takes $2\mathcal{D}'$ epochs in the worst case. Hence this procedure also takes $O(\mathcal{D}')$ epochs. Hence we conclude this discussion with the following theorem.

Theorem 3.8. *The* FastAPF *algorithm solves the* Apf *problem in* $\mathcal{LUMI}$ *model in* $O(\mathcal{D}')$ *epochs.*

Next, we show that the algorithm proposed in [10] takes $O(\mathcal{D}'^2)$ epochs to complete, which will prove that the FastAPF algorithm is faster. In phase 4 of

the proposed algorithm in [10] a robot might need to make $\Omega(\mathcal{D}^2)$ movements. Therefore this algorithm requires $\Omega(\mathcal{D}'^2)$ epoch time to complete. We state this in the following result.

Result 3.1. *Algorithm proposed in [10] requires $\Omega(\mathcal{D}'^2)$ epoch time to solve Arbitrary Pattern Formation problem.*

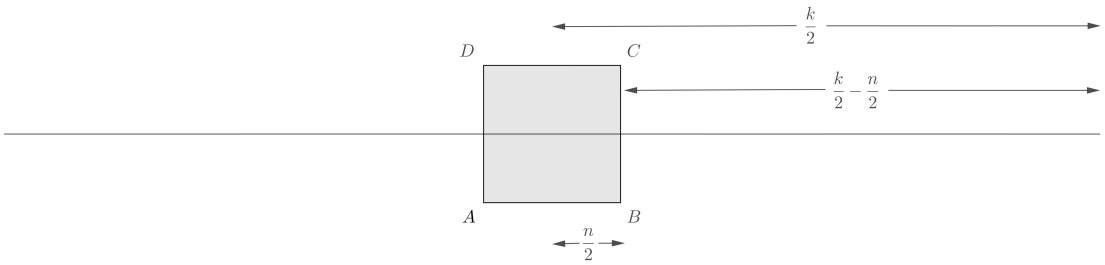

Figure 3.5: An image related to Theorem 3.9

Now we show that our algorithm is asymptotically optimal in time. Let's consider an initial configuration of robots has the smallest enclosing rectangle a square of length n and each grid point of the square has a robot. Thus there are $k = n^2$ robots in total. Let the target pattern be a compact line. Then we can see that the farthest point on the line (wherever it may be placed) from the initial configuration is at least $\frac{k-n}{2} = \frac{k}{2} - \frac{\sqrt{k}}{2}$ (See Figure 3.5). Hence there is a robot that at least needs to move $\frac{k}{2} - \frac{\sqrt{k}}{2}$ steps. Hence at least $\frac{k}{2} - \frac{\sqrt{k}}{2}$ epochs (assume ceiling value) is necessary. In this case, note that $\mathcal{D} = k$. Hence, we can state the following.

Theorem 3.9. *Any algorithm solving* Apf *problem requires $\Omega(\mathcal{D}')$ epochs.*

The above result shows that the algorithm FastAPF is time optimal asymptotically. Next, we show that algorithm FastAPF is also move optimal.

Theorem 3.10. *Algorithm* FastAPF *all total requires at most $O(\mathcal{D}'^2)$ robot movements.*

Proof. Any robot participating in Procedure-I, Procedure-II, Procedure-III, or Procedure-IV makes at most $2\mathcal{D}'$ moves. Next any robot participating in Procedure-

V only makes a horizontal move of length at most $\mathcal{D}'$. Next any robot participating in Procedure-VI makes at most a vertical move (if the robot is not on a saturated horizontal line) of maximum length $\mathcal{D}'$ followed by a horizontal move of maximum length $\mathcal{D}'$. Hence for each procedure, a robot participating in that procedure makes $O(\mathcal{D}')$ moves in that procedure. Since starting from any asymmetric configuration the algorithm FastAPF terminates via passing through a finite number of procedures. Hence for each robot, it needs to make $O(\mathcal{D}')$ moves. Thus, total number of required robot moves in algorithm FastAPF is $O(k\mathcal{D}') = O(\mathcal{D}'^2)$. □

3.4 Concluding Remarks

This chapter studied algorithm for Arbitrary Pattern Formation (Apf) problem on an infinite rectangular grid by a robot swarm starting from any asymmetric initial configuration in classical $\mathcal{LUMI}$ robotic model. This work proposed algorithm for the Apf problem in $\mathcal{LUMI}$ model which is asymptotically time optimal and move optimal as well. Suppose the dimensions of the smallest enclosing rectangles of the initial configuration of the robots and the pattern to be formed are respectively $m \times n$ and $M \times N$. Let $\mathcal{D} = \max\{m, n, M, N\}$ and $\mathcal{D}' = \max\{\mathcal{D}, k\}$, then the algorithm takes total $O(\mathcal{D}')$ epoch time and total $O(\mathcal{D}'^2)$ moves to solve the Apf problem. The algorithm in this chapter which is time optimal and move optimal is in $\mathcal{LUMI}$ model, so studying for an algorithm in $\mathcal{OBLOT}$, which is both move optimal and time optimal, could be a possible direction from this work. Also another version of Apf problem where the initial or the target configuration of robots can have multiplicity points can consider for further research.

Chapter 4

Oblivious Robots Performing Different Tasks on Grid Without Knowing their Team Members

The Gathering and Arbitrary pattern formation are two vastly studied problems by researchers in the field of swarm robot algorithms. These are one of the fundamental tasks which can be done by autonomous robots in different settings. In the gathering problem, n number of robots initially positioned arbitrarily meets at a point not fixed from earlier within finite time. It is not always easy to meet at a point with very weak robots in the distributed system. Similarly, the Arbitrary pattern formation problem is such that robots have to form a given pattern that is given as input to the robots within a finite time. In literature, there are several works that have considered either gathering or arbitrary pattern formation problem separately. But none of those works consider robots deployed in the same environment working on two different tasks. The environment of robot swarm needs periodic maintenance for making the environment robust from faults and some other factors. So if the same robot swarm deployed in the environment can do the maintenance apart from doing the specific task assigned to them, it would be more cost-effective. From this motivation and practical interest, in [6] authors first studied the problem where two teams of oblivious robots work on two different tasks, namely gathering and circle formation on a plane. Here the crucial part is that a robot knows to which team

it belongs, but it can not recognize another robot's team. The novelty of the problem would have gone away a bit if the robots are luminous and gathering team robots put a color on a light to indicate which team they belong to. But the main motivation of this problem is to extend the work for more than two different tasks and for assigning different colors for different tasks would make the number of colors unbounded. For this reason, it is convenient to solve the problem with the least possible capabilities for the robot swarm. Also $\mathcal{OBLOT}$ model is more self-stabilized and fault tolerant. For these reasons, in our work, we also consider robots to be oblivious. Now it is challenging to design a distributed algorithm by which two different teams of oblivious robots can do two different tasks simultaneously on a discrete domain because, in any discrete graph, the movements of robots become restricted. So avoiding collision becomes a great challenge.

In this work, we have provided a collision-less distributed algorithm following which two teams of oblivious robots with weak multiplication detection ability can do two different tasks namely gathering and arbitrary pattern formation simultaneously on an infinite rectangular grid under the asynchronous scheduler. Here we assume that the initial configuration is asymmetric.

4.1 Robot Model

In this section, we shall first formally describe the model. Then we introduce the problem statement of this chapter.

Robots Robots are anonymous, identical, and oblivious, i.e. they have no memory of their past rounds. They can not communicate with each other. There are two teams between the robots. One is $\mathcal{T}_{Apf}$ and the other team is $\mathcal{T}_g$. A robot r only knows that in which team it belongs to between this two. But a robot can not identify to which team another robot belongs. All robots are initially in distinct positions on the grid. The robots can see the entire grid and all other robots' positions which means they have global visibility. This implies the robots are transparent and hence the visibility of a robot can not be obstructed by

another robot. Robots have no access to any common global coordinate system. They have no common notion of chirality or direction. A robot has its local view and it can calculate the positions of other robots with respect to its local coordinate system with the origin at its own position. There is no agreement on the grid about which line is x or y-axis and also about the positive or negative direction of the axes. As the robots can see the entire grid, they will set the axes of their local coordinate systems along the grid lines. Also, robots have weak multiplicity detection capability, which means a robot can detect a multiplicity point but cannot count the number of robots present at a multiplicity point.

Look-Compute-Move cycles An active robot operates according to the Look-Compute-Move cycle. In each cycle a robot takes a snapshot of the positions of the other robots according to its own local coordinate system (Look); based on this snapshot, it executes a deterministic algorithm to determine whether to stay put or to move to an adjacent grid point (Compute); and based on the algorithm the robot either remain stationary or makes a move to an adjacent grid point (Move). When the robots are oblivious they have no memory of past configurations and previous actions. After completing each Look-Compute-Move cycle, the contents in each robot's local memory are deleted.

Scheduler We assume that robots are controlled by a fully asynchronous adversarial scheduler (ASync). The robots are activated independently and each robot executes its cycles independently. This implies the amount of time spent in Look, Compute, Move, and inactive states are finite but unbounded, unpredictable, and not the same for different robots. The robots have no common notion of time.

Movement In discrete domains, the movements of robots are assumed to be instantaneous. This implies that the robots are always seen on grid points, not on edges. However, we do not need this assumption. In the proposed algorithm, we assume the movements are to be instantaneous for simplicity. However, this

algorithm also works without this. The movement of the robots is restricted from one grid point to one of its four neighboring grid points.

4.2 Problem Description

We define a problem on an infinite rectangular grid where n oblivious, identical, autonomous robots are dispersed on the vertices of the infinite rectangular grid. In [6] they solved two conflicting tasks by robots on a plane where one team of robots gather at a point and another team of robots form a circle on the plane. But here robots are on an infinite rectangular grid and solve two different distributed problems. Here are two teams of oblivious robots where one team is $\mathcal{T}_g$ and the other team is $\mathcal{T}_{Apf}$. The goal of the robots of $\mathcal{T}_g$ is to meet all the robots of this team to a point on an infinite rectangular grid. Another team is $\mathcal{T}_{Apf}$ where the goal of this team is to form a particular pattern that is given as input. The next section provides the algorithm for solving this problem.

4.3 The Main Algorithm

4.3.1 Global Coordinate Agreement

Here we have to first fix the global coordinate system and then we aim to gather the $\mathcal{T}_g$ robots to the origin and form the arbitrary pattern by $\mathcal{T}_{Apf}$ robots with respect to the coordinate $(0, 2)$. So let us consider an infinite rectangular grid G as a cartesian product $P \times P$, where P is an infinite (from both sides) path graph. The infinite rectangular grid G is embedded in the Cartesian Plane R^2. Here, some robots will gather at a point and other robots will form an arbitrary pattern on the grid. Here we are assuming that the initial configuration is asymmetric. A robot can form a local coordinate system aligning the axes along the grid lines but the robots do not have an access to any global coordinate system even. To form the target pattern the robots need to reach an agreement on a global coordinate system. In this subsection, we will provide the details of the procedure that allows

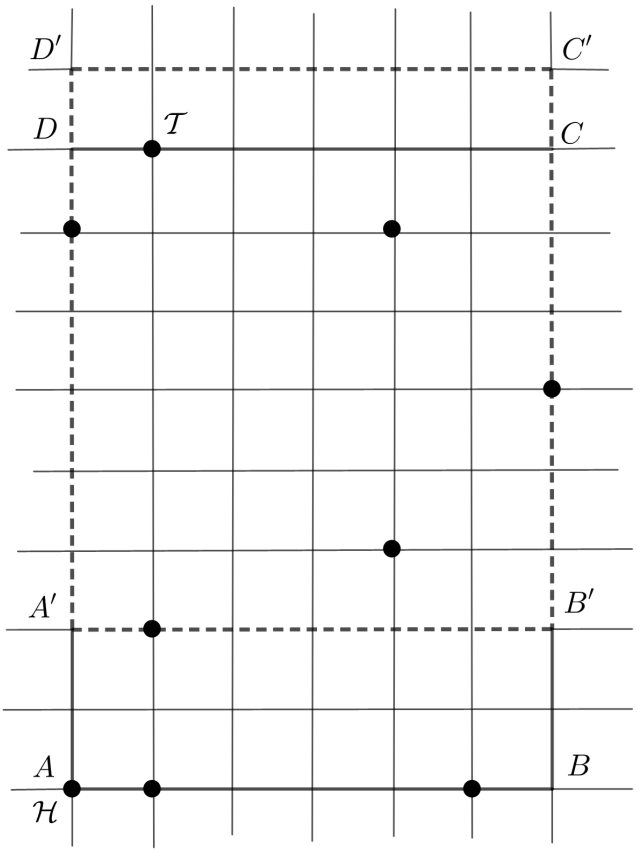

Figure 4.1: $ABCD$ is the smallest enclosing rectangle of initial configuration. $\mathcal{H}$ and $\mathcal{T}$ are head and tail of the configuration. A is the origin and the target configuration will be embedded with respect to $(0, 2)$. $A'B'C'D'$ is the smallest enclosing rectangle of target configuration

the robots to reach an agreement on a global coordinate system.

For a given configuration ($\mathcal{C}$) formed by the robots, let the smallest enclosing rectangle, denoted by $s.rect(\mathcal{C})$, be the smallest grid-aligned rectangle that contains all the robots. Suppose the $s.rect$ of the initial configuration $\mathcal{C}_I$ is a rectangle $\mathcal{R} = ABCD$ of size $m \times n$, such that $m > n > 1$. Let $|AB| = n$. Then consider the binary string $\{p_i\}$ associated with a corner A, λ_{AB} as follows. Scan the grid from A along the shorter side AB to B and sequentially all grid lines of $s.rect(\mathcal{C}_\mathcal{I})$ parallel to AB in the same direction. And $p_i = 0$, if the position is unoccupied and $p_i = 1$ otherwise. Similarly construct the other binary strings λ_{BA}, λ_{CD} and

λ_{DC}. Since the initial configuration is asymmetric we can find a unique lexicographically largest string. If λ_{AB} is the lexicographically largest string, then A is called the leading corner of $\mathcal{R}$.

Next, suppose $\mathcal{R}$ is an $m \times m$ square, then consider the eight binary strings λ_{AB}, λ_{BA}, λ_{CD}, λ_{DC}, λ_{BC}, λ_{CB}, λ_{AD}, λ_{DA}. Again since the initial configuration is asymmetric, we can find a unique lexicographically largest string among them. Hence we can find a leading corner here as well.

Next, let $\mathcal{C}_I$ be a line AB, we will have two strings λ_{AB} and λ_{BA}. Since $\mathcal{C}_I$ is asymmetric then λ_{AB} and λ_{BA} must be distinct. If λ_{AB} is lexicographically larger than λ_{BA}, then we choose A as the leading corner.

Now for either case, if λ_{AB} is the lexicographically largest string then the leading corner A is considered as the origin, and the $x-$ axis is taken along the AB line. If $\mathcal{C}_I$ is not a line then the $y-$ axis is taken along the AD line. If $\mathcal{C}_I$ is a line then the $y-$ coordinate of all the positions of robots is going to be zero and in this case, the $y-$ axis will be determined later. For any given asymmetric configuration $\mathcal{C}$ if λ_{AB} is the largest associated binary string to $\mathcal{C}$ then the robot causing the first non-zero entry in λ_{AB} is called *head* let $\mathcal{H}$ and the robot causing last non zero entry in λ_{AB} is called as *tail* let $\mathcal{T}$. We denote the i^{th} robot of the λ_{AB} string as r_{i-1}. A robot other than the *head* and *tail* is called ***inner robot***. Further we denote $\mathcal{C}' = \mathcal{C} \setminus \{tail\}$ and $\mathcal{C}'' = \mathcal{C} \setminus \{tail, head\}$ and $\mathcal{C}''' = \mathcal{C} \setminus \{head\}$.

Let $\mathcal{C}_{target}$ be the target configuration for the $\mathcal{T}_{Apf}$ robots and $s.rect(\mathcal{C}_{target}) = \mathcal{R}_{target}$. Let $\mathcal{R}_{target}$ is a rectangle of size $M \times N$ with $M \geq N$. We can calculate the binary strings associated with corners in the same manner as previously. $\mathcal{C}_{target}$ is expressed in the coordinate system with respect to the point $(0, 2)$, where $(0, 2)$ will be the leading corner. Let the $A'B'C'D'$ be the smallest rectangle enclosing the target pattern with $A'B' \leq B'C'$. Let $\lambda_{A'B'}$ be the largest (may not be unique) among all other strings. Then the target pattern is to be formed such that A is the origin and pattern embedded with respect to the position $(0, 2)$, $A'B'$ direction is along the positive x axis and $A'D'$ direction is along the positive y axis. If the target pattern has symmetry then we have to choose any one among the larger string and fixed the coordinate system. So as previously said

$head_{target}$ will be the first one and $tail_{target}$ will be the last one in the $s.rect$ of $\mathcal{C}_{target}$. Also, we define $\mathcal{C}_{final}$ is the configuration when all $\mathcal{T}_g$ robots are at same point and $\mathcal{C}_{target}$ configuration is formed. $\mathcal{C}'_{final} = \mathcal{C}_{final} \setminus \{tail_{target}\}$, $\mathcal{C}''_{final} = \mathcal{C}_{final} \setminus \{head_{target}, tail_{target}\}$, $\mathcal{C}'''_{final} = \mathcal{C}_{final} \setminus \{head_{target}\}$. We denote the $head_{target}$ position as t_1 and $tail_{target}$ position as t_k. Let H_i be the horizontal line having the height i from the x-axis. Let for each i there are $p(i)$ target positions on H_i. We denote the target positions of H_0 as $t_1.....t_{p(0)-1}$ from left to right. For H_1 we denote the target positions as $t_{p(0)}$ to $t_{p(0)+p(1)-1}$ from right to left. For H_2 we denote the target positions as $t_{p(0)+p(1)}$ to $t_{p(0)+p(1)+p(2)-1}$ from right to left. Similarly, we can denote all other target positions on H_i, $i > 0$ except $tail_{target}$.

4.3.2 Brief Discussion of Algorithm

Let the initial configuration is $\mathcal{C}_{\mathcal{I}}$, the final configuration is $\mathcal{C}_{final}$. Robots are operating on an infinite grid. There are two teams of robots where one is $\mathcal{T}_g$ and another one is $\mathcal{T}_{Apf}$ (let us assume that $|\mathcal{T}_g| > 2$). $\mathcal{T}_g$ robots will gather at the same point on the grid and the robots of $\mathcal{T}_{Apf}$ will form a pattern on the grid. Note that the gathering point will not be a target point of the target pattern. A robot only knows in which team it belongs between $\mathcal{T}_g$ and $\mathcal{T}_{Apf}$. A robot has no information about to which team other robots belong. Robots first fix the global coordinate system. The target will form with respect to the point (0,2) and the gathering will occur at the origin. When a robot awakes up it first calculates the *head* robot and *tail* robot. In the first three stages, the *head* robot will move to the origin and the *tail* robot will expand the smallest enclosing rectangle for maintaining the asymmetry of the configuration. Then in the stage 4 all the inner robots now move to the x-axis and make a line on the x-axis. Note that a line is called ***compact line*** when there is no empty grid point between two robots. So robots on the x-axis first make the line compact then one by one inner robot from upward horizontal lines move down to the x-axis. After this in stage 5 the robot which is not on a line say r, will move to its closest endpoint of the line if it $\in \mathcal{T}_g$ or it will move one step upward if it belongs to $\mathcal{T}_{Apf}$. In the next stage after a multiplicity point is formed or calculating the position of the *tail*, robots

on the x-axis move to the fixed target positions which are either the origin or the target positions. Robots will move from right to left with respect to the position of the *head* sequentially. As all inner robots are on the x-axis so the robot $\in \mathcal{T}_g$ can always find the neighboring grid lines empty, so the robot can move to the origin by choosing any path to the origin if all the robots are on a line. If a robot can see the *tail* robot then it will choose the neighboring line of the x-axis in the direction of the *tail* for its movement. In this way, one by one all inner robots move to their fixed target positions. Next if *tail* sees that all inner robots move to their target positions it will move to their fixed position. In the last stage if the *head* robot is in the gathering team then it will not move but if it is in the Apf team then when without its position all the pattern formation is done it will move to its position. Note that within the algorithm no two robots collide without the gathering point and the coordinate system remains unchanged. Within finite time all $\mathcal{T}_g$ robots move to the same point and the $\mathcal{T}_{Apf}$ robots form the fixed pattern that is given as input.

4.4 Correctness

4.4.1 Description of the Stages

The main difficulty of this problem is a robot does not know which team another robot belongs to. The robot only knows about its own team. Here we will show that there will not arise any symmetry and no collision will occur. The global coordinate system will also not change. The algorithm is divided into eight stages. In this situation, the global coordinate will fix as we mention in sec 4.3.1

Stage-1 *Input:* $\{P_4 \wedge \neg P_{14} \wedge \neg(P_5 \wedge P_6)\}$ is true.
The *tail* robot will move upwards and all other robots will remain static.
Aim: The aim is to make $\{P_5 \wedge P_6\}$ = true. After finite number of moves by *tail* robot stage-1 completes with $\{P_4 \wedge \neg P_{14} \wedge P_5 \wedge P_6\}$ is true.

Theorem 4.1. *If we have an asymmetric configuration $\mathcal{C}$ in stage 1 at some time*

P_0	$\mathcal{C} = \mathcal{C}_{final}$
P_1	$\mathcal{C}' = \mathcal{C}'_{final}$
P_2	All $\mathcal{T}_g$ robots are at a same point
P_3	$\mathcal{C}_{Apf} = \mathcal{C}_{target}$
P_4	The current configuration is asymmetric
P_5	$m = \max\{N, n\} + 2$
P_6	$m = 2.\max\{M, V\}$ where V is the length of the vertical side of the smallest enclosing rectangle of $\mathcal{C}'$
P_7	The *head* in $\mathcal{C}$ is at the origin
P_8	$n \geq \max\{N+1, H+1, k\}$ where H is the length of the horizontal side of the smallest enclosing rectangle of $\mathcal{C}'$
P_9	Line formation on x-axis without *tail*
P_{10}	$\mathcal{C}'$ has a non-trivial reflectional symmetry with respect to a vertical line
P_{11}	$\mathcal{C}''' = \mathcal{C}'''_{final}$
P_{12}	$\mathcal{C}'' = \mathcal{C}''_{final}$
P_{13}	$m \geq \max\{N, n\} + 2$, $m \geq 2.\max\{M, V\}$ and m is odd
P_{14}	There exist a multiplicity point

Table 4.1: If any of the Boolean variable P_i where $0 \leq i \leq 14$ on the left column is true then the corresponding condition on the right column is satisfied and vice versa.

t, then

1. *after one move by the tail towards upward, the new configuration is still asymmetric and the coordinate system remains unchanged.*

2. *after a finite number of moves by the tail towards upward, stage 1 terminates with* $(P_5 \wedge P_6)$*= true.*

Proof. Let $\mathcal{C}$ be the asymmetric configuration where $ABCD$ be the smallest enclosing rectangle of the initial configuration and $|AB| = n$, $|AD| = m$, $m \geq n$. By one move of the tail robot towards upward, the new smallest enclosing rectangle is $ABC'D'$. Let r be the tail robot and after the movement of r it is now on the edge $C'D'$. As λ_{AB} is the largest lexicographically string. So $\lambda_{AB} > \lambda_{BA}$. Now

the new strings associated to the corner A of the smaller side of $ABC'D'$ be λ'_{AB} and λ'_{BA}.

Let r be the only robot on CD. Then it is easy to see that $\lambda'_{AB} > \lambda'_{BA}$. But when there are more than one robot on CD, then we have to show that by movement of tail robot $\lambda'_{AB} > \lambda'_{BA}$ is true. Let r corresponds to the i^{th} term and j^{th} term of λ_{AB} and λ_{BA}. If $i = j$ then r is the middle robot of CD. Then when tail moves upward one step, the increase number of 0's of λ'_{AB} and λ'_{BA} are same. So $\lambda'_{AB} > \lambda' BA$. Now if $i < j$ then if we calculate the binary strings of λ_{AB} and λ_{BA} then *tail* will be appear earlier in BA, than AB. Now if we calculate the binary strings of first j term of AB and BA then $\lambda_{BA}|_j > \lambda'_{BA}|_j$. Also $\lambda_{AB}|_j > \lambda'_{AB}|_j$. But $\lambda_{AB} > \lambda_{BA}$. So we have $\lambda_{AB}|_j > \lambda_{BA}|_j$. Finally we can say $\lambda'_{AB}|_j > \lambda'_{BA}|_j$, so $\lambda'_{AB} > \lambda'_{BA}$. When $i < j$ in that case when *tail* robot move upward then in the new $\mathcal{SER}$ we can calculate that in this case also λ'_{AB} and λ'_{BA}.

So in all the cases $\lambda'_{AB} > \lambda'_{BA}$. Now we show that the binary string of AB is larger than $C'D'$. As we know that the binary string of AB is larger than CD, but when the tail robot moves one step upward in that case as there is no robot other than the tail in $C'D'$ so if we calculate binary string it will be $\lambda'_{AB} > \lambda'_{C'D'}$.

In this case, $ABC'D'$ is a non-square grid, so four binary strings to consider here. By calculating we can say that AB is the largest binary string in this new smallest enclosing rectangle. So the new configuration is still asymmetric. So the coordinate system is unchanged. As the tail moves upward so the x-coordinate remains unchanged.

□

Stage-2 *Input:* $\{P_4 \wedge \neg P_{14} \wedge P_5 \wedge P_6 \wedge \neg P_7\}$ is true.
The *head* robot will move to origin. So *head* robot will move towards left if it is not initially at the origin. In this move *head* robot will remain *head*.
Aim: After finite number of moves by the *head* robot stage-2 completes when $\{P_4 \wedge \neg P_{14} \wedge P_5 \wedge P_6 \wedge P_7\}$ is true.

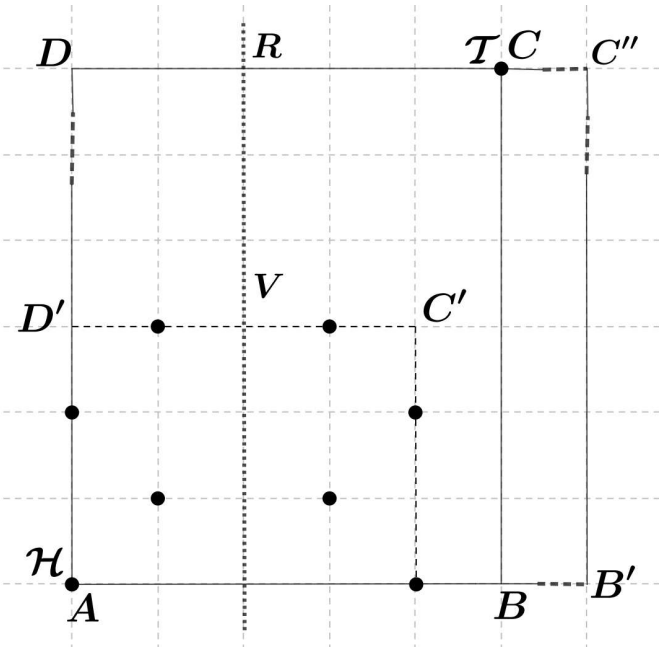

Figure 4.2: Case-1: $\mathcal{C}'$ has a vertical symmetry and *tail* will move rightwards

Theorem 4.2. *If we have an asymmetric configuration* $\mathcal{C}$ *in stage 2 at some time* t*, then*

1. *after one move by the head leftwards, the new configuration is still asymmetric and the coordinate system is unchanged.*

2. *after finite number of moves by the head,* P_7 *becomes true, and Stage 2 terminates with* $\{P_4 \wedge \neg P_{14} \wedge P_5 \wedge P_6 \wedge P_7\}$ *true.*

Proof. Let $\mathcal{C}$ be the initial configuration and $ABCD$ be the smallest enclosing rectangle. As λ_{AB} is the largest string so the position of head robot is the first 1 in the largest string. Let i be the first position of 1 in λ_{AB} string. So there is no 1 before i^{th} position. Now when head robot moves left then the 1 will occur earlier than the i^{th} position. So λ_{AB} remains the lexicographically largest string upto head reaches origin. So 1) and 2) holds. □

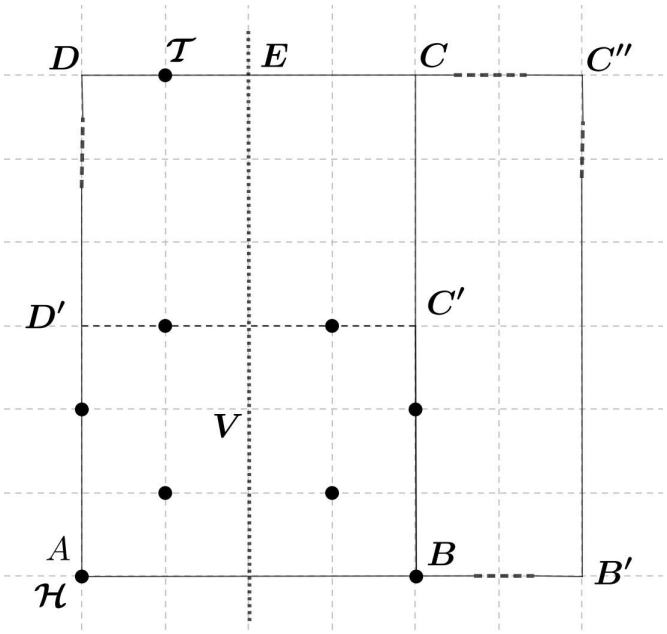

Figure 4.3: Case-2: $\mathcal{C}'$ has a vertical symmetry and *tail* will move leftwards

Stage-3 *Input:* $\{P_4 \wedge \neg P_{14} \wedge P_5 \wedge P_6 \wedge P_7 \wedge \neg P_8\}$ is true.
The *tail* robot will move rightwards.
Aim: The main aim of this stage is to make P_8 true. After movement of robots $\{P_4 \wedge \neg P_{14} \wedge P_5 \wedge P_6 \wedge P_7 \wedge P_8\}$ is true.

Theorem 4.3. *If we have an asymmetric configuration $\mathcal{C}$ in stage 3 at some time t with P_{10} is false, then*

1. *after one move by the tail rightwards, the new configuration is still asymmetric and coordinate system is unchanged.*
2. *after one move by the tail robot rightwards, P_8 becomes true.*
3. *after finite number of moves by the tail, $\{P_4 \wedge \neg P_{14} \wedge P_5 \wedge P_6 \wedge P_7 \wedge P_8\}$ is true.*

Proof. Let $ABCD$ is the smallest enclosing circle at time t, where λ_{AB} is the largest string. After one move by the tail rightward there may arise two cases.

Case-1: Let the smallest enclosing circle remain unchanged after one move by the tail rightward. As in this phase, the head is in origin and the tail has moved until P_5 and P_6 is true, in the binary string of CD or DC is smaller than AB. Also in this phase, P_{10} is not true. So we must have AB larger string than BA, so finally we get $\lambda'_{AB} > \lambda'_{BA}$.

Case-2: Suppose now tail robot is at C, then by one move of tail the new $\mathcal{SER}$ is $APQD$, where $|AP|= (n+1)$. Now it is easy to check that as $m \geq n+2$ so $m > n+1$. So we get that $AD > AP$. This implies that the new configuration is still not square, so we have to consider here only four binary strings, and as earlier λ_{AP} will be a larger string. So we can conclude that the configuration is still asymmetric and the coordinate system is not changed by one move of the tail. It is easy to check that P_6 and P_7 are true here but not P_{12}. After the movement of the tail, P_5 may become false, so we are in stage 1 then. Then the tail moves upwards and one upwards move still has $P_5 \wedge P_6 \wedge P_7 \wedge P_8 \neg P_{12}$ is true.

Note that P_{10} is either true or false in stage 3 by a finite number of moves of the tail the configuration remains asymmetric.

□

In this stage the robots will check if P_{10} is true or false. If P_{10} is true then there may arise two cases:

Case-1 If *tail* is in the rightwards of the vertical symmetric line they it will move and make P_8 true.

Case-2 If the *tail* robot is in the left direction of the vertical symmetric line then it will not move rightwards. $Tail$ will then move in leftwards upto one more step than D (fig 4.3). In this case the co-ordinate system will be changed.

Theorem 4.4. *If we have an asymmetric configuration $\mathcal{C}$ in stage 3, then after finite number of moves by the tail robot, the configuration remains asymmetric.*

Stage-4 *Input:*$\{P_4 \wedge \neg P_{14} \wedge P_5 \wedge P_6 \wedge P_7 \wedge P_8 \wedge \neg P_9\}$ is true.
The inner robots will move to the x-axis and form a line. Other than the *tail* robot, all other inner robots form a line on the x-axis. In stage four, the *head* is in origin, and all the robots on the x-axis first make the line compact, i.e. there is no empty grid point between two robots. After the x-axis becomes compact, when a robot r_i is on H_i and there are no robots in between H_i and the x-axis and the right part of r_i is empty in its horizontal line, then the robot moves to the x-axis. This procedure is done one by one by robots. In between this movement, no collision will occur. So all the inner robots other than the *tail* form a line on the x-axis.
Aim: When all the inner robots move to x-axis then $\{P_4 \wedge \neg P_{14} \wedge P_5 \wedge P_6 \wedge P_7 \wedge P_8 \wedge P_9\}$ is true.

Theorem 4.5. *If we have an asymmetric configuration $\mathcal{C}$ at some time t, with $\{P_4 \wedge \neg P_{14} \wedge P_5 \wedge P_6 \wedge P_7 \wedge P_8 \wedge \neg P_9\}$ is true, then*

1. *after any move by the inner robots the configuration is asymmetric and coordinate system is unchanged.*
2. *after finite number of moves by the inner robots P_9 becomes true and stage 4 terminates with with $\{P_4 \wedge \neg P_{14} \wedge P_5 \wedge P_6 \wedge P_7 \wedge P_8 \wedge P_9\}$ is true.*

Proof. In stage 4, head robot is in origin and tail robot is at a position satisfying $P_5 \wedge P_6 \wedge P_8$ is true. In this scenario, when robots on x axis move left then the string associated to AB becomes larger than the previous one. Also in this stage one by one horizontal line's robots will move to the x axis. The movement of robots on a horizontal line will be from right to left. So in this movement by the inner robots, coordinate system will not change and as the movement is sequential so no collision will occur by the movement of inner robots. By finite moves by the robots all inner robots are on x axis without the tail robot. Then stage 4 terminates. □

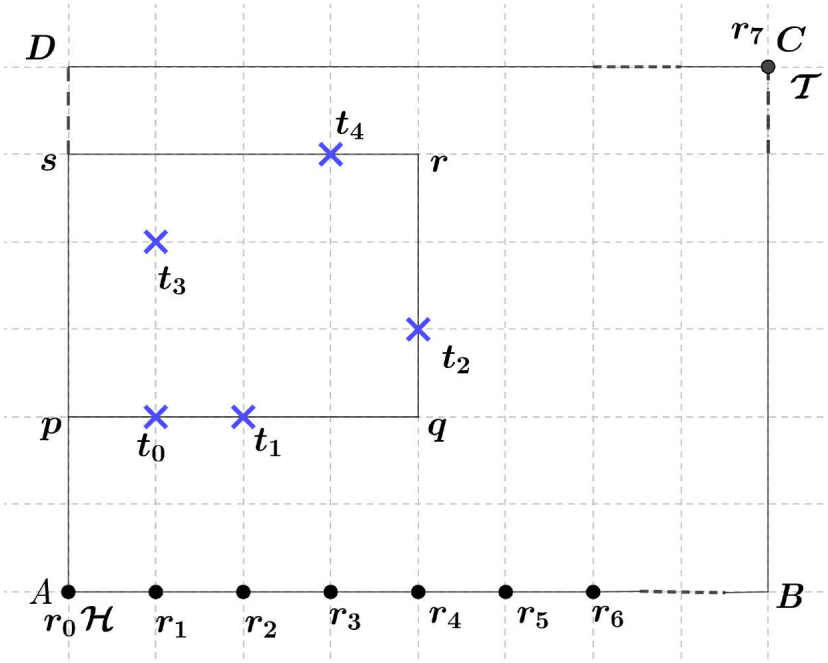

Figure 4.4: Any $r_i \in \mathcal{T}_g$ moves to A and $r_i \in \mathcal{T}_{Apf}$ moves to fixed target positions one by one.

Stage-5 *Input:* $P_4 \wedge \neg P_{14} \wedge \neg P_{13} \wedge P_7 \wedge P_9$ is true.
One robot which is not on a compact line will move to its closest end point of that line following the shortest path if it $\in \mathcal{T}_g$. As without one robot all other robots are on line so that one robot will move downwards. Then P_{14} is true. Or when $(P_4 \wedge \neg P_{14} \wedge P_7 \wedge P_9) \wedge (P_5 \wedge P_6 \wedge P_8)$ is true then that one robot will move upward from its position if it $\in \mathcal{T}_{Apf}$. By this move P_{13} will be true.

Aim: $P_4 \wedge \neg P_{14} \wedge P_7 \wedge P_9 \wedge (P_{13} \vee P_{14})$ is true..
After this stage if P_{14} is true then all robots will fix the multiplicity point as origin, the compact line is as x axis and the other perpendicular axis as y axis.

Theorem 4.6. *If we have an asymmetric configuration $\mathcal{C}$ in stage 5, then after movement of one robot either one multiplicity point will create or P_{13} will be true.*

Proof. After the stage 4, all the inner robots are on x axis. When the tail robot is in team $\mathcal{T}_{Apf}$ then it will move upward one step and makes P_{13} true. Without

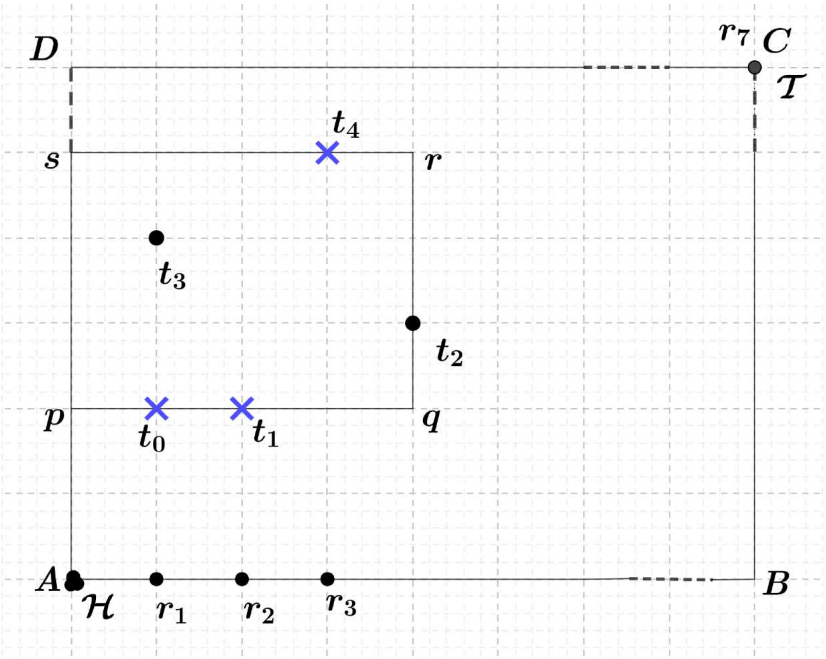

Figure 4.5: A multiplicity point at A and some of the $\mathcal{T}_{Apf}$ robots form the pattern

the tail all other robots are on line so head robot will not change and the λ_{AB} remains the lexicographically largest string. So the coordinate system will not change. Now when the tail robot is in team $\mathcal{T}_g$ then it will move downwards. In this case also, tail will move downwards to its nearest corner. As all other robot is in x axis, when tail robot move downwards, the coordinate system does not change untill it reaches the nearest corner robot. But when tail robot reaches its nearest corner robot, then one multiplicity point will create. Afterwards the multiplicity point will be treated as origin and the line where other robots are placed treated as x axis. Any line perpendicular to the line treated as y axis. So in this way by the move of the tail robot either P_{14} that is one multiplicity point or P_{13} will be true. □

Stage-6 *Input:* $P_{13} \vee P_{14}$ is true.
When the *tail* robot will see that P_{13} is true then it will not move. An inner robot when sees that it is the rightmost robot on x-axis and P_{13} is true and there exists robots at the positions $t_{k-1}, t_{k-2}.....t_{k-i+1}$ where $1 \leq i \leq k$ (let t_k be the

position of *tail*) then that inner robot will move upwards to t_{k-i} if it $\in \mathcal{T}_{Apf}$. If the robot $\in \mathcal{T}_g$ then it will move to the origin. Note that in this case robot can fix the global coordinate so it will move to the origin by choosing the first horizontal line in the positive direction of the y-axis. When a rightmost robot of team APF on x-axis sees that P_{13} is true and no robot on target positions then it will move to t_{k-1}. Note that when a left most robot without *head* on x axis of APF team sees that only $tail_{target}$ or $tail_{target}$ and $head_{target}$ are not occupied by robots then the robot will move to $tail_{target}$. Again if P_{14} is true then the rightmost robot on x-axis sees that there exist robots on $t_k, t_{k-1}.....t_{k-i+1}$ where $1 \leq i \leq k$ then that inner robot moves upward to t_{k-i} if it $\in \mathcal{T}_{Apf}$. If it $\in \mathcal{T}_g$ then move to multiplicity point. Here when a robot can see all robots on a line then it will fix that line as the x-axis and anyone perpendicular line as the y-axis and the multiplicity point as the origin. So the robot $\in \mathcal{T}_g$ moves to the origin by choosing the first horizontal line in the positive direction of y-axis. In this way, the inner robots move to their target positions one by one.

Aim: After this stage $P_{14} \wedge P_{12}$ is true.

After this stage, the multiplicity will be the origin and as the target pattern will be formed with respect to $(0, 2)$ so either there will be a robot at $tail_{target}$ or the *tail* robot will be at a position maintaining P_{13} true. So we can choose the larger side as the y-axis and the other one as the x-axis.

Theorem 4.7. *If we have an asymmetric configuration $\mathcal{C}$ in stage 6 at time t, then after a finite number of moves by the inner robots stage 6 terminates and P_{12} is true.*

Proof. In this stage, all the inner robots on the line x-axis will move to its destination point which will be either on the target position or the origin. Let, P_{13} is true. In that case, the tail robot will be at a position maintaining the P_{13} is true. So the inner robots on the x axis will move one by one. As P_{13} is true, so the rightmost robot on x axis first moves either at t_{k-1} if it is $\in \mathcal{T}_{Apf}$ or it will move to origin. The robot will move to the origin choosing the first horizontal line. As without tail, all robots are on x axis, and the target embedding is with respect to $(0, 2)$ position. So there will be no robot on the first horizontal line

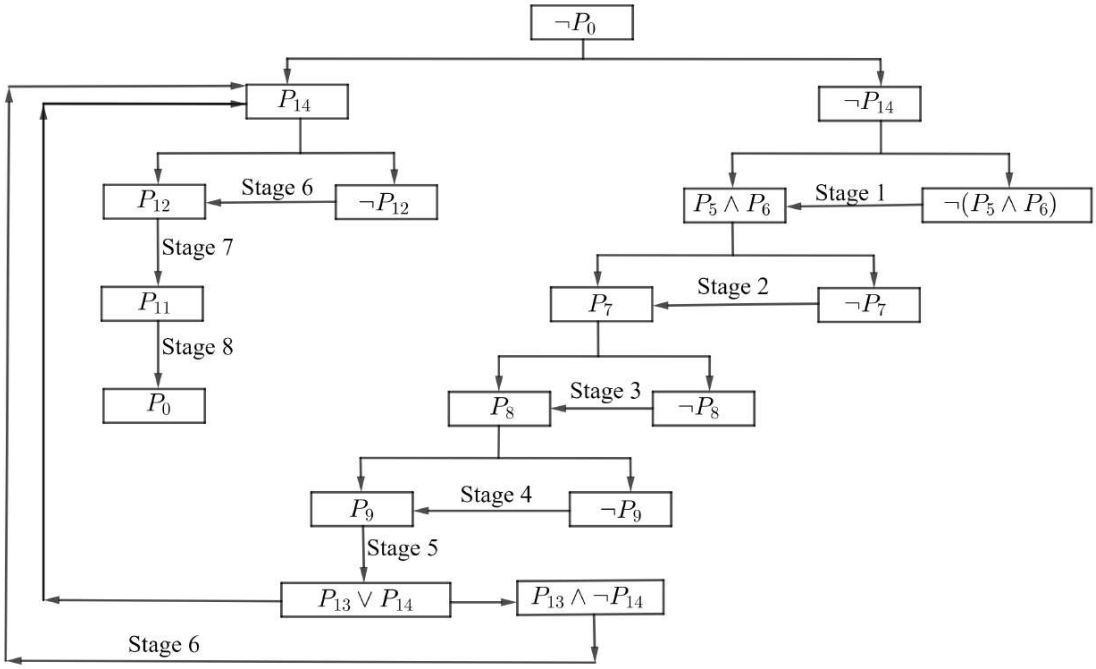

Figure 4.6: Flow chart of the algorithm

above x axis. So the robots of x axis from right to left will move to the position, either at t_{k-1}, t_{k-2}..... or move to origin. So within this sequential movements of robots, no collision will occur and configuration remain asymmetric. Now when P_{14} is true. In that case, the multiplicity point will be origin and the robots on a line will be x axis. The perpendicular any direction of x axis will be y axis. So the rightmost robot on x axis will move t_k if it is $\in \mathcal{T}_{Apf}$ or move to the origin. Here also, robots move one by one. So no collision and symmetric configuration will occur. If P_{14} true and head and tail both are $\in \mathcal{T}_g$, then by this stage P_0 will be true. Also when tail is in $\mathcal{T}_g$ but head is in $\mathcal{T}_{Apf}$, then by this stage P_{11} will be true and then it will be in stage 8. But if head and tail are in $\mathcal{T}_{Apf}$ and P_{13}, then by the movement of inner robots, P_{12} will be true and stage 6 terminates. □

Stage-7 *Input:*In this stage $P_{14} \wedge P_{12} \wedge \neg P_{11}$ is true.
If $tail \in \mathcal{T}_{Apf}$ then move to the position of the $tail_{target}$. But if $tail \in \mathcal{T}_g$ then the *tail* will move to the origin. If $tail \in \mathcal{T}_g$ then the *tail* will move downwards up to the x-axis and then move towards the multiplicity point (By P_{13} condition we

can say that there will be no robot in this path of *tail*'s movement).
Aim: P_{11} is true.

Theorem 4.8. *If we have an asymmetric configuration $\mathcal{C}$ in stage 7 at time t then after the finite number of moves stage 7 terminates with P_{11} is true.*

Proof. When in stage 7, without head and tail robots, all the other robots are in target positions. So if $tail \in \mathcal{T}_{Apf}$ then there is no robot on the position of $tail_{target}$. The tail robot will first moves left up to the x-coordinate of $tail_{target}$ and then move down towards $tail_{target}$. In this move, as P_{14} already true, so nu symmetry will occur and only one robot will move in this stage. So collision will not occur also. When the tail robot reaches the $tail_{target}$, then the P_{11} will be true. But when the tail robot is in team gather and P_{12} is true, then all inner robots form the pattern with respect to $(0, 2)$. So there is no robot on the x axis. Tail robot first move downwards and then move left through the x axis towards the origin. In the path of tail robots move, there is no other robot will be present as P_{14} and P_{12} are true. So after the movement of tail robot, P_{11} will be true and stage 7 terminates. □

Stage-8 *Input:* $P_{11} \wedge P_{14}$ is true.
If *head* robot is in gather team then it will not move. So then P_0 is true. But if it belongs to $\mathcal{T}_{Apf}$ then it moves to upward and then to the position of $head_{target}$.
Aim: After the movement of *head* then P_0 is true.

Theorem 4.9. *If we have an asymmetric configuration $\mathcal{C}$ in stage 8 at time t then by the move of head robot, stage 8 terminates and P_0 is true.*

Proof. When in stage 8, then head robot if is in team gather then P_0 is already true. As head will not move in this case. But if head is in $\mathcal{T}_{Apf}$ then in this stage all the other robots are in their destination points without the head robot. As already P_{14} is true, and there are $|\mathcal{T}_g| > 2$), so here by the move of head robot P_{14} remains true. As the pattern will be form with respect to the $(0, 2)$ point, so

head will upward and then move to its destination point. If After the movement of head robot, P_0 becomes true. □

So after completing these stages P_2 and P_3 conditions will be true. No collision will occur during the movement of robots throughout the algorithm. So the gathering and arbitrary pattern formation will be done by the robots simultaneously on an infinite grid by oblivious robots.
The proposed algorithm is depicted in the flowchart in fig 4.6. Starting from any configuration where $\neg P_0$ from the diagram fig 4.6 each directed path starting from the node where $\neg P_0$ ends at the node where P_0 is true. Hence we can conclude the theorem.

Theorem 4.10. *Gathering and Arbitrary pattern formation are solvable in* ASYNC *by $\mathcal{T}_{Apf}$ and $\mathcal{T}_g$ robots from any asymmetric initial configuration.*

4.5 Concluding Remarks

In this work, we claim that by our algorithm two different teams of robots can simultaneously gather and form an arbitrary pattern on an infinite rectangular grid. A robot only knows that in which team it belongs to between gathering and APF. To our knowledge in a grid network, this is the first work where two different tasks are performed simultaneously by two different teams of robots. Here we assume the grid is infinite and the visibility is full. So for further work, we can extend this work by assuming limited visibility and also when the grid is finite.

Chapter 5

Gathering of Finite Memory Robots on a Circle under Limited Visibility

In swarm robotics, robots achieving some tasks with minimum capabilities is the main focus of interest. In the last two decades, there is a huge research interest in robots working on coordination problems. It is not always easy to use robots with strong capabilities for real-life applications, as making these robots is not at all cost-effective. If a swarm of robots with minimum capabilities can do the same task then it is effective to use swarm robots rather than using robots with many capabilities, as designing these robots of the swarm is very much cheaper and simple than making robots with many capabilities.

The gathering is a very vastly studied problem by researchers. This is one of the fundamental tasks of autonomous robots in the distributed system. The gathering problem requires n robots that are initially positioned arbitrarily must meet at a single point within finite time. Note that the meeting point is not fixed initially. When there are two robots, then this task is called *rendezvous*. It is not always easy to meet at a point by very weak robots in the distributed system. So, it is challenging to design a distributed algorithm to gather some robots that are inexpensive and with fewer capabilities.

In this work, we investigate the gathering of robots on a circle. The robots are on

a perimeter of a circle of fixed radius and they can only move along the perimeter of that circle. Here the robots have limited visibility which means each robot can see only the points on the circle that have an angular distance strictly smaller than a constant θ from the robot's current location, where $0 < \theta \leq \pi$. Here angles are expressed in radians. This problem becomes trivial if a unique leader can be chosen who remains the leader throughout the algorithm. But even with chirality, electing a unique leader is not so much trivial even when the robots can not see only one point on the circle. Also even if a unique leader is elected, making sure that the leader remains the leader throughout the execution of the algorithm is another challenge that has to be taken care of. This challenge makes this problem quite nontrivial and interesting. Here we investigate the gathering of robots in an asynchronous scheduler on a circle with limited visibility with $\mathcal{FSTA}$ robots.

5.1 Robot Model

In the problem, we are considering the $\mathcal{FSTA}$ robot model. The robots are anonymous and identical, but not oblivious. Robots have finite persistent memory. Robots cannot communicate with each other. Robots have weak multiplicity detection capability, i.e., robots can detect a multiplicity point, but cannot determine the number of robots present at a multiplicity point. All robots are placed on a circle of fixed radius. The robots agree on a global sense of handedness. All robots move at the same speed and their movement is rigid. Robots operate in Look-Compute-Move cycle. In each cycle, a robot takes a snapshot of the positions of the other robots according to its own local coordinate system (Look); based on this snapshot, it executes a deterministic algorithm to determine whether to stay put or to move to another point on the circle. (Compute); and based on the algorithm the robots either remain stationary or make a move to a point (Move). We assume that robots are controlled by a fully asynchronous adversarial scheduler (ASync). The robots are activated independently and each robot executes its cycles independently. This implies the amount of time spent in

look, compute, move, and inactive states are finite but unbounded, unpredictable, and not necessarily the same for different robots. We assume that the look and compute phase together in an LCM cycle of a robot is not instantaneous. The robots have no common notion of time. Here the initial configuration is asymmetric. Robots have limited visibility which means a robot cannot see the entire circle. Let a and b be two points on a circle $\mathcal{C}$, then the angular distance between a and b is the measure of the angle subtended at the center of $\mathcal{C}$ by the shorter arc with endpoints a and b. A robot has visibility π, which means that it can see all the other robots which are with angular distance less than π.

5.2 Our contribution

Under full visibility, gathering on a circle by robots in an asynchronous scheduler is not so difficult when robots agree on a global sense of handedness. But excluding one point from visibility makes the gathering problem a bit difficult. Here we have proposed an algorithm solving the gathering of robots on a circle with $\mathcal{FSTA}$ robot model under ASync scheduler and visibility π. The work that is most related to our work is done by Di Luna *et al.* [46]. In their work, they have shown that robots on a circle can not gather if the visibility is less or equal to $\frac{\pi}{2}$ and provided an algorithm under SSync scheduler with $\mathcal{OBLOT}$ robots considering a robot's having π visibility. In their given algorithm, a robot can cross its neighbor's position. But for an asynchronous scheduler, it is not possible. By crossing the neighbor's position two multiplicity points may be created at an angular distance π under the asynchronous scheduler. So the configuration can become symmetric. So motivated by this if we want to extend this work, the first challenge is to design an algorithm where $\mathcal{OBLOT}$ robots can gather at a point on the circle under ASync scheduler. But to design that, it has to be made sure that if two multiplicity points are formed, the angular distance between them must not be π. Finally from that two multiplicity points, all robots gather at a point. But as the visibility is π, a robot can not distinguish whether its antipodal exists or not. In the $\mathcal{OBLOT}$ model, with an asynchronous scheduler, it may

happen that two antipodal robots decide to move by an algorithm. But we can not allow two antipodal robots to move to their neighbors' positions in the same direction, as neighbors also can be antipodal. If any one robot moves clockwise, but not to its neighbor's position or counterclockwise then, as the robots are oblivious, they can not distinguish whether its antipodal existed or not, in the initial configuration. So, it is not easy to design such an algorithm where all robots finally gather at a point on the circle with oblivious robots, maintaining the asymmetric configuration. Now, if a robot has finite memory it can distinguish by its state. So we consider here the $\mathcal{FSTA}$ model where robots have finite persistent memory. However, we only compromise the obliviousness of the robots to get the liberty of asynchronous scheduler.

However, we do not consider a general asynchronous scheduler. Because if two robots are at angular distance π and if the time taken to complete the look and compute phase is instantaneous then by considering $\mathcal{FSTA}$ robots it may happen that a robot can not distinguish whether there exists a robot outside its visibility or not. Precisely, we assumed that in the LCM cycle of a robot, it takes a nonzero time (non instantaneous) to finish its look and compute phase together. If an algorithm can be designed that works under asynchronous scheduler by introducing finite memory, it would be of great practical interest as equipping robots with memory is not at all hard or much costlier. In this work, our main achievement is that, by equipping the robots with finite memory, we have gained a deterministic gathering algorithm that works under asynchronous scheduler. So this assumption is fairly realistic in a practical scenario.

5.3 Definitions and preliminaries

Definition 5.1 (Configuration)**.** *A configuration is a pair (C_p, f_p) where C_p is the set of all points on the circle and $f_p : C_p \to \{0, 1, 2\}$ is a function defined as:*

$$f_p(x) = \begin{cases} 0 & \textit{if there is no robot on } x \\ 1 & \textit{if there is exactly one robot on } x \\ 2 & \textit{if there is more than one robot on } x \end{cases} \tag{5.1}$$

Definition 5.2 (Rotationally Symmetric Configuration)**.** *A configuration with no multiplicity point is said to be rotationally symmetric if there is a nontrivial rotation with respect to the center which leaves the configuration unchanged.*

Definition 5.3 (Antipodal robot)**.** *A robot r is said to be an antipodal robot if there exists a robot r' on the angular distance π of the robot. In such a case, r and r' are said to be antipodal robots to each other.*

Note that a robot that is not antipodal is said to be non antipodal robot.

Definition 5.4. *Let r' and r'' be two robots in a configuration positioned at distinct positions. Then $cwAngle(r', r'')$ ($ccwAngle(r', r'')$) is the angular distance from r' to r'' in clockwise (counter clockwise) direction.*

Definition 5.5 (Angle sequence)**.** *Let r be a robot in a given configuration with no multiplicity point and let $r_1, r_2, \ldots, r_n$ be the other robots on the circle in clockwise order. Then the angular sequence for robot r is the sequence*

$$(cwAngle(r, r_1), cwAngle(r_1, r_2), cwAngle(r_2, r_3), \ldots, cwAngle(r_n, r)).$$

We denote this sequence as $\mathcal{S}(r)$. We further denote the sub sequence

$$(cwAngle(r, r_1), cwAngle(r_1, r_2), cwAngle(r_2, r_3), \ldots, cwAngle(r_{i-1}, r_i))$$

of $\mathcal{S}(r)$ as $\mathcal{S}(r, r_i)$. Further, we call $cwAngle(r, r_1)$ as the leading angle of r.

Note that as the configuration is initially rotationally asymmetric, by results from the paper [46] we can say that all the robots have distinct angle sequences.

Definition 5.6 (Lexicographic Ordering)**.** *Let $\tilde{a} = (a_1, \ldots, a_n)$ and $\tilde{b} = (b_1, \ldots, b_n)$ be two finite sequences of reals of same length. Then $\tilde{a}$ is said to be lexicographically strictly smaller sequence if $a_1 < b_1$ or there exists $1 < k < n$ such that $a_i = b_i$ for all $i = 1, 2, \ldots, k$ and $a_{k+1} < b_{k+1}$. $\tilde{a}$ is said to be lexicographically smaller sequence if either $\tilde{a} = \tilde{b}$ or $\tilde{a}$ is lexicographically strictly smaller sequence.*

Definition 5.7 (True leader)**.** *In a configuration, a robot with the lexicographically smallest angular sequence is called a true leader.*

If the configuration is rotationally asymmetric and contains no multiplicity point, there exists exactly one robot which has strictly the smallest lexicographic angle sequence. Hence, there is only one true leader for such a configuration. Since a robot on the circle cannot see whether its antipodal position is occupied by a robot or not. So a robot can assume two things: 1) the antipodal position is empty, let's call this configuration $C_0(r)$ 2) the antipodal position is nonempty, let's call this configuration $C_1(r)$. So a robot r can form two angular sequences. One considering $C_0(r)$ configuration and another considering $C_1(r)$. The next two definitions are from the viewpoint of a robot. If the true leader robot can confirm itself as the true leader, we call it *Sure Leader*. If the true leader or some other robot has an ambiguity of being a true leader depending on the possibility of $C_0(r)$ or $C_1(r)$ configuration, then we call it *Confused Leader*. There may be the following possibilities.

- Possibility-1: $C_0(r)$ configuration has rotational symmetry, so $C_1(r)$ is the only possible configuration.
- Possibility-2: $C_1(r)$ configuration has rotational symmetry, so $C_0(r)$ is the only possible configuration.
- Possibility-3: Both $C_0(r)$ and $C_1(r)$ has no rotational symmetry, so both $C_0(r)$ and $C_1(r)$ can be possible configurations.

Definition 5.8 (Sure leader)**.** *A robot r in a rotationally asymmetric configuration with no multiplicity point is called Sure Leader if r is the true leader in $C_0(r)$ and $C_1(r)$ configurations.*

Note that, the Sure leader is definitely the true leader of the configuration. Hence at any time if the configuration is asymmetric and contains no multiplicity point, there is at most one Sure leader.

Definition 5.9 (Confused leader)**.** *A robot r in a rotationally asymmetric configuration with no multiplicity point is called a Confused Leader if both $C_0(r)$ and $C_1(r)$ are possible configurations and r is a true leader in one configuration but not in another.*

Definition 5.10 (Follower robot)**.** *A robot in an asymmetric configuration with no multiplicity point is said to be a follower robot if it is neither a sure leader nor a confused leader.*

Definition 5.11 (Expected leader)**.** *A robot in an asymmetric configuration with no multiplicity point is said to be an expected leader if it is not a follower robot. That is, an expected leader is either a sure leader or a confused leader.*

Note that the above definitions are set in such a way that the sure leader (or, a confused leader or, a follower robot) can recognize itself as a sure leader (or, a confused leader or, a follower robot).

Definition 5.12. *For two robots r and r' situated at different positions on the circle such that $cwAngle(r, r') = \theta$ we define $[r, r']$ as the set of points x on the circle such that $0 \leq cwAngle(r, x) \leq \theta$ and (r, r') as the set of points x on the circle such that $0 < cwAngle(r, x) < \theta$.*

Definition 5.13. *Let r be a robot in a configuration, then another robot r_1 is said to be situated at the left of r if $cwAngle(r, r_1) > \pi$ and said to be at right if $cwAngle(r, r_1) < \pi$.*

Note that, the true leader of the configuration can become a confused leader and a robot other than the true leader can also become a confused Leader. The next results lead us to find the maximum possible number of expected leaders.

Proposition 5.1. *Let $(a_1, \ldots, a_k)$ be the angular sequence of the true leader, say r_0, in a rotationally asymmetric configuration with no multiplicity point. Then there cannot be another robot r' with all of the following properties.*

1. *r' is at left side to r_0,*
2. *$\mathcal{S}(r', r_0) = (a_1, a_2, \ldots, a_i)$,*
3. *First i angles of $\mathcal{S}(r_0)$ respectively are $a_1, a_2, \ldots, a_{i-1}$ and a_i.*

Proof. We prove this result by contradiction. If possible let there be such a robot r'. Then note that $\mathcal{S}(r_0)$ is $\tilde{a} = (a_1, \ldots, a_i, a_{i+1}, \ldots, a_{k-i}, a_1, \ldots, a_i)$ and $\mathcal{S}(r')$ is $\tilde{a}_1 = (a_1, \ldots, a_i, a_1, \ldots, a_i, a_{i+1}, \ldots, a_{k-i})$. Since r_0 has the strictly smallest angular sequence, so a_1 is the smallest angle in the configuration and also $a_{i+1} \leq a_1$, which leads to $a_{i+1} = a_1$. Next, we show that $a_2 = a_{i+2}$. Since the $\tilde{a}$ is strictly the smallest angular sequence so $a_{i+2} \leq a_2$. If $a_{i+2} < a_2$, then the angular sequence $(a_{i+1}, \ldots, a_{k-i}, a_1, \ldots, a_i, a_1, \ldots, a_i)$ is smaller than $\tilde{a}$, which is a contradiction. Hence $a_2 = a_{i+2}$. Therefore by a similar argument, we can show that $a_{i+j} = a_j$ for $j = 3, \ldots, i$. Now if $2i = k$ then we see that $\tilde{a} = \tilde{a}_1$, which contradicts the fact that the configuration is rotationally asymmetric. Otherwise, proceeding similarly we can show that $a_{2i+j} = a_j$, for $j = 1, 2, \ldots, i$. Repeating the same argument we can show that $a_{pi+j} = a_j$, for $j = 1, 2, \ldots, i$. Since there are finitely many angles, so after finite number of steps we must end up having that $k = ti$ where $t \geq 2$ and $\tilde{a} = \tilde{a}_1 = (a_1, \ldots, a_i, a_1, \ldots, a_i, \ldots, a_1, \ldots, a_i)$, which is again a contradiction. □

Next, we state a simple observation in the following Proposition 5.2.

Proposition 5.2. *Suppose there is a rotationally asymmetric configuration with no multiplicity point where the true leader is r_0, then on including a robot, say r, on the circle at an empty point without bringing any rotational symmetry, the true leader of the new configuration must be in $[r_0, r]$.*

For a confused leader r, there may be two possibilities. The first one is when r is a true leader in $C_0(r)$ configuration, but not in $C_1(r)$. The second one is when r is a true leader in $C_1(r)$ configuration, but not in $C_0(r)$. We show that the second possibility cannot occur. We formally state the result in the following Proposition.

Proposition 5.3. *If a robot r is a confused leader in asymmetric configuration with no multiplicity point, r is the true leader in $C_0(r)$ configuration but r is not the true leader in $C_1(r)$ configuration.*

From Proposition 5.3 one can observe that if the true leader of an asymmetric configuration with no multiplicity point is a confused leader then its antipodal

position must be empty. And also, if a robot, which is not the true leader of the configuration, becomes a confused leader then its antipodal position must be non-empty. We record these observations in the following Corollaries.

Corollary 5.0.1. *In a rotationally asymmetric configuration with no multiplicity point if the true leader of the configuration is a confused leader then its antipodal position must be empty.*

Corollary 5.0.2. *In a rotationally asymmetric configuration with no multiplicity point if a robot other than the true leader becomes a confused leader then its antipodal position must be non-empty.*

Proposition 5.4. *Let $\mathcal{C}$ be a rotationally asymmetric configuration with no multiplicity point and L be the true leader of the configuration, then another expected leader r of $\mathcal{C}$ must satisfy $cwAngle(L, r) \geq \pi$.*

Proof. If possible let there be another expected leader r such that $cwAngle(L, r) < \pi$. Since r is not the leader of the configuration, so r must be a confused leader. Hence from Proposition 5.3 we have that r is true leader in $C_0(r)$ configuration and L is the true leader of $C_1(r)$ configuration. This contradicts the Proposition 5.2.

□

Result 5.1. *For any given rotationally asymmetric configuration with no multiplicity point, there can be at most one confused leader other than the true leader.*

Let $\mathcal{C}$ be a rotationally asymmetric configuration with no multiplicity point. Then $\mathcal{C}$ will have a true leader and from above Proposition 5.2 there can be at most one more confused leader. Hence we can have the following four exhaustive cases for $\mathcal{C}$.

1. $\mathcal{C}$ has exactly one expected leader and that is a sure leader.
2. $\mathcal{C}$ has exactly one expected leader and that is a confused leader.
3. $\mathcal{C}$ has exactly two expected leaders and both are confused leaders.

4. $\mathcal{C}$ has exactly two expected leaders. One of them is a sure leader and another one is a confused leader.

The Fig. 5.1 and Fig. 5.2 give the existence of all four above cases. For the third case, we observe two properties in the following Propositions.

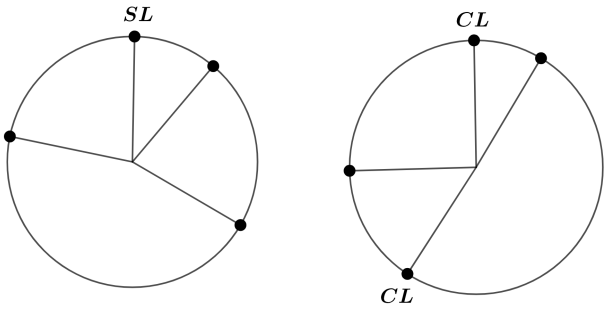

Figure 5.1: In the left figure only one sure leader (SL) and the right figure two confused leaders (CL) in the configuration.

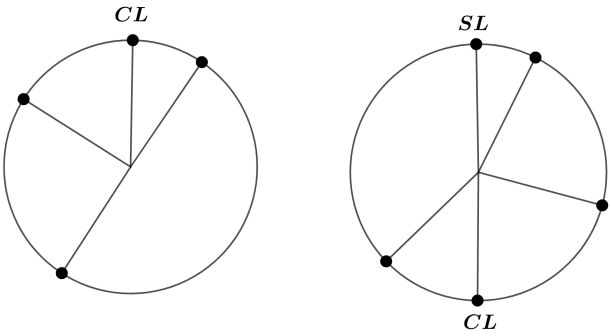

Figure 5.2: In the left figure only one confused leader (CL) and the right figure one sure leader (SL) and one confused leader (CL).

Result 5.2. *If for a rotationally asymmetric configuration with no multiplicity point if there are two confused leaders then they cannot be antipodal of each other.*

Proof. Let p and q be two confused leaders of a rotational asymmetric configuration with no multiplicity point. Now one of p and q must be the true leader of the configuration. Without loss of generality let p be the leader of the configuration.

Then from the Corollary 5.0.1 antipodal position of p must be empty. Hence another confused leader q cannot be at the antipodal position of p. □

Result 5.3. *If for a rotationally asymmetric configuration with no multiplicity point if there are two confused leaders then their clockwise neighbors cannot be antipodal to each other.*

Proof. If possible let the clockwise neighbors of two confused leaders be antipodal to each other. Let L be the true leader of the configuration and r be another confused leader. Then from Proposition 5.4 and Result 5.2 we have $cwAngle(L, r) > \pi$. Then the leading angle of L becomes strictly greater than the leading angle of r (Figure 5.3). This contradicts the fact that L is the true leader of the configuration. □

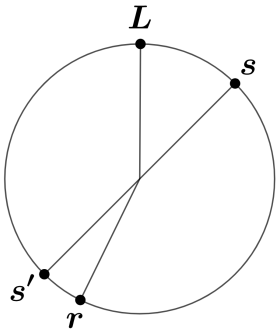

Figure 5.3: An image related to Result 5.3

5.4 Proposed Algorithm

In this section, we describe the proposed algorithm that works for asynchronous robots with finite memory. The initial configuration is rotationally asymmetric and the robots are at distinct points on the circle. Each robot has π visibility, which means, a robot cannot see its antipodal position. Each robot can move at a uniform speed on the arc of the circle. The proposed algorithm is provided in the Algorithm 1. Before discussing this algorithm, we define a *safe* clockwise neighbor.

Definition 5.14 (Safe neighbor)**.** *Suppose r is an expected leader which is confused and s is the first clockwise neighbor of r. The robot s is said to be a safe neighbor of r if the first clockwise neighbor of the true leader of $C_1(r)$ configuration is not antipodal to s.*

Discussion of the Algorithm: Robots are initially dispersed on distinct points of a circle and the initial configuration is rotationally asymmetric. Robots have limited visibility. A robot cannot see its antipodal position. As the initial configuration is rotationally asymmetric, each robot has distinct angle sequences. On getting activated a robot r first checks whether it is the only robot in its visibility. If yes, then it moves $\pi/2$ angle clockwise. Otherwise, it looks for a multiplicity point. If there is any multiplicity point in the visible configuration then there are two possibilities. Possibility number one, r itself is located at a multiplicity point, and possibility number two is not located at a multiplicity point. If r is not located at a multiplicity point, and one of the multiplicity points is a clockwise or anticlockwise neighbor of r, then r moves to the closer multiplicity point. Suppose r is located on a multiplicity point. For such a case, if there is another multiplicity point other than where r is located, and that multiplicity point is at a clockwise distance less than π then r moves to that multiplicity point. Next, if there is no multiplicity point in the visible configuration, then it decides whether it is an expected leader or a follower robot. If r is a follower robot then, it does nothing. If r is a sure leader and its state is `off`, then it moves to its first clockwise neighbor. If r is a confused leader then it checks its state. If its state is `terminate` then it does nothing. If its state is `off` then there are several cases. Case-I: If the first clockwise neighbor of r is safe, then r moves to its neighbor's position. Case-II: If the first clockwise neighbor of r is not safe and $C_0(r)$ configuration has another confused leader other than r, then r does nothing. Case-III: If neither of Case-I or Case-II holds then r changes its state to `moveHalf` and moves $\theta/2$ angular distance clockwise.

Now suppose, on getting activated, r is at the state `moveHalf`. If the first clockwise neighbor of r, say r', is not antipodal then r changes its state to `terminate`

and does not move. Otherwise, let s be the robot at the antipodal position of r' and the leading angle of r is $\theta/2$. Then if there is a robot visible to r in the arc $[s - \frac{\theta}{2}R, s + \frac{\theta}{2}R)$ (let R is the radius of the circle) then r changes its state to terminate and move $\theta/2$ angular distance counterclockwise. If r' is antipodal and there is no robot visible to r in the arc $[s - \frac{\theta}{2}R, s + \frac{\theta}{2}R)$, then r changes the state to `moveMore` and moves $\theta/4$ angular distance clockwise.

Next, suppose on getting activated, r is at the state `moveMore`. If the first clockwise neighbor of r, say r', is not antipodal, then r changes its state to `terminate` and does not move. Otherwise, let s be the robot at the antipodal position of r' and the leading angle of r is $\theta/4$. If there is a robot visible to r in the arc $[s - \frac{\theta}{4}R, s + \frac{\theta}{4}R)$, then r changes its state to `terminate` and move $3\theta/4$ angular distance counterclockwise. If r' is antipodal and there is no robot visible to r in the arc $[s - \frac{3\theta}{4}R, s + \frac{\theta}{4}R)$, then r changes the state to `off` and moves at the position of its first clockwise neighbor.

Note that, according to the definition of sure leader (confused leader, follower robot), a robot can identify itself whether it is a sure leader (confused leader, follower robot) or not.

Now, we formally present the proposed algorithm in the form of a pseudo code (Algorithm 1). Next, we categorize an initial configuration that is rotationally asymmetric with no multiplicity points in the following:

- *Configuration-A:* Only the expected leader is the true leader of the configuration. If the expected leader is a confused leader then its clockwise first neighbor is safe.

- *Configuration-B:* There are two expected leaders in the configuration.

  - *Configuration-BI:* when the confused leader which is not the true leader, finds its clockwise first neighbor safe.

  - *Configuration-BII:* when the confused leader which is not the true leader, finds its clockwise first neighbor unsafe.

Algorithm 1: Gathering algorithm for visibility π

```
1  The algorithm is executed by a generic robot r with initial state off;
   Input: The set of points occupied by robots visible to r
   Output: Destination point for robot r
2  if there is a robot visible then
3      if there is no multiplicity point then
4          if if the robot r has state off then
5              if the robot r is the sure leader then
6                  move to its clockwise neighbor's position;
7              else if the robot r is a confused leader then
8                  if the clockwise first neighbor of r is safe then
9                      move to the position of its clockwise first neighbor ;
10                 else if C_0(r) configuration does not have another confused leader
                    other than r then
11                     θ ← leading angle of r ;
12                     change the state to moveHalf and move θ/2 angular distance
                        clockwise ;
13         else if the robot r has the state moveHalf then
14             θ ← 2 × leading angle of r;
15             s ← the antipodal position of r;
16             if clockwise neighbor of r is non antipodal then
17                 change the state to terminate;
18             else if there is no robot in the arc [s − (θ/2)R, s + (θ/2)R) then
19                 change the state to moveMore;
20                 move θ/4 angular distance clockwise;
21             else if there is a robot in the arc [s − (θ/2)R, s + (θ/2)R) then
22                 change the state to terminate;
23                 move θ/2 angular distance counterclockwise;
24         else if the robot r has the state moveMore then
25             θ ← 4 × leading angle of r;
26             s ← antipodal position of r;
27             if clockwise neighbor of r is non antipodal then
28                 change the state to terminate;
29             else if there is no robot in the arc [s − (3θ/4)R, s + (θ/4)R) then
30                 change the state to off;
31                 move to the position of its clockwise first neighbor;
32             else if there is a robot in the arc [s − (3θ/4)R, s + (θ/4)R) then
33                 change the state to terminate;
34                 move 3θ/4 angular distance counterclockwise;
35     else if there is a multiplicity point but the robot r is not at any multiplicity point
        then
36         if its clockwise or counter-clockwise neighbor is a multiplicity point then
37             Move to the closer multiplicity point;
38     else if Only visible position is a multiplicity point then
39         if clockwise angular distance from the multiplicity point is < π/2 then
40             Move to the multiplicity point;
41 else
42     Move π/2 distance in clockwise direction;
```

- *Configuration-C:* One expected leader which is a confused leader, sees that its clockwise first neighbor is not safe.

5.5 Correctness and the Main Results

We will prove that, if all robots are initially placed on a rotationally asymmetric configuration with no multiplicity point then, execution of Algorithm 1 ensures that all robots eventually meet at a point within finite time and no longer move under the asynchronous scheduler. First, we show that, from the initial configuration, within finite execution of Algorithm 1, at least one and at most two multiplicity points will be created by robots. Then, from a configuration with one or two multiplicity points, the robots eventually gather at one of the multiplicity points and do not move further.

Lemma 5.1. *If the initial configuration is of type configuration-A, then within finite execution of Algorithm 1, at least one multiplicity point will form.*

Proof. In the initial configuration, let L be the only expected leader which is either a sure leader or, a confused leader, that finds its first clockwise neighbor safe. Then from the Algorithm 1, L moves to its clockwise neighbor, say, s. All robots other than L in the initial configuration are follower robots. If on activation, a robot recognizes itself as a follower robot, then it does not move. Therefore, until L starts its move towards s, no other robot will move. Also, we claim that s does not decide to move on activation until L finishes its move. Thus, if the claim is correct, a multiplicity point will form where s is located, in the initial configuration.

To prove the above claim, let's consider a moment when L is moving toward s and s gets activated. At this moment, the leading angle of L is strictly smaller (not even equal) than the leading angle of s, and s can see the leading angle of L. Hence, at this moment, s remains a follower, and s does not decide to move while L is moving. □

Lemma 5.2. *If the initial configuration is type configuration-B, then after finite time execution of Algorithm 1 at least one multiplicity point will form.*

Proof. Let in the initial configuration there be two expected leaders. Then there are two possibilities, either there is one sure leader and one confused leader or, there are two confused leaders.

One sure leader and one confused leader: Let L and r be respectively the sure leader and confused leader in the initial configuration. Let s and r' be the first clockwise neighbors of L and r respectively.

Case-I (When L starts moving first) In this case, if the sure leader L, gets activated, when the confused leader has not started moving, the robot L moves where s is located. If L starts moving first then the leading angle of L remains strictly smallest at any point of time after the move (because the speed of each robot is the same) until L reaches s. The leading angle of L is visible to s, and s remains a follower robot until L reaches s. Hence, s does not move when L is approaching it. Thus, a multiplicity point is formed where s is located initially.

Case-II (When r starts moving first or simultaneously with L) In this case, if r starts moving first then it moves clockwise at a distance depending on r finds its clockwise neighbor safe or not.

Case-IIA (When r finds its first clockwise neighbor safe) In this case, r starts moving towards r' in the clockwise direction. There can be two possibilities, either the leading angle of r is equal to that of L or, greater than that of L. For the first case, L and r must not be antipodal to each other. While r is moving the leading angle of r remains the smallest angle in the configuration (even if L starts moving simultaneously with r). Hence, r' remains a follower robot until r reaches r'. So, a multiplicity point is formed where r' is located in the initial configuration. For the second case, the leading angle of L is visible to r', which is strictly smaller than its own leading angle. Hence, r' remains a follower robot and remains static when r is approaching r' until L moves and creates a multiplicity point at the location of s.

Case-IIB (When r finds its first clockwise neighbor unsafe) There are two possibilities in this case; (1) robot L wakes up before r starts moving (2) robot L wakes up after r starts moving. For the first possibility, we show that a multiplicity point will be formed where s is located. Here, according to the proposed Algorithm, r only moves in the major arc joining the initial position of the r and r'. All the follower robots except s can see the leading angle of r which is the strictly smallest angle at the time. Also, for the robot s, it can see the leading angle of L. So except L, no robot moves when r is moving in the major arc joining the initial position of r and r' until a multiplicity point is formed. Hence, when L moves to s a multiplicity point must be formed. Now consider the second case, when r starts moving before L wakes up for the first time. In this case, r moves $\frac{\theta}{2}$ angle clockwise turning its state to `moveHalf`, where θ is the leading angle of r in the initial configuration. While making this move if L gets activated then it identifies itself as a follower robot because the leading angle of r is strictly smaller than that of L. And for a similar reason as discussed in the previous case, no other initial follower robot also moves. After finishing the move, when r next time gets activated, L robot is visible to r (If condition in line 23 is true), then it returns to its initial positions turning its state to `terminate`. Hence the initial configuration is restored and further on waking up r does not do anything, as it becomes a confused leader with the state `terminate`. After the initial configuration is restored, when L wakes up, it moves to s forming a multiplicity point there.

Two confused leaders: Let L and r be two confused leaders in the initial configuration such that, L is the true leader of the configuration. Then first we show that r must find its clockwise first neighbor to be safe. From Proposition 5.2, we have that L and r are not antipodal of each other. And from Proposition 5.4, we have $cwAngle(L, r) > \pi$. Thus, the first clockwise neighbors of r and L cannot be antipodal to each other. Since L is the true leader of the present configuration, so $C_1(r)$ is the present configuration. Hence, from the definition of the safe neighbor, the first clockwise neighbor of r is safe. If L starts moving first (this happens when L finds its first clockwise neighbor safe), then using a

similar reason as used in Lemma 5.1 a multiplicity point will be formed where s is located initially. If r starts moving first, then r' can see both the leading angles of L and r, so unless L forms a multiplicity point at the location of s, r' does not move while r is approaching it.

□

Lemma 5.3. *If the initial configuration is type configuration-C, then after finite time execution of Algorithm 1 at least one multiplicity point will form.*

Proof. When the initial configuration is of type configuration-C then the only expected leader, say r, is a confused leader with its first clockwise neighbor, say s, not safe. In this case, the confused leader r changes its state to `moveHalf` and moves $\frac{\theta}{2}$ angle clockwise where θ is the leading angle of r initially. Let the antipodal position of s be s'. Observe that after r starts its move, s' robot may be elected as a confused leader as s' cannot see the angle $cwAngle(r, s)$. As r is the true leader of the configuration even when it is moving, for s', r is also the true leader in the configuration $C_1(s')$. Thus, s' finds its first clockwise neighbor as safe. Suppose, before the next activation of r, s' starts moving then the clockwise first neighbor, say s'', can see the leading angle of r which is smaller than that of s'. Hence s'' does not move while s approaches it. Thus a multiplicity point is formed where s'' is initially located. Next, suppose s' does not move until r gets activated next, then by the condition of line 20 in algorithm 1, r moves $\frac{\theta}{4}$ angle clockwise. Similarly, if before the third activation of r, s' starts moving, then a multiplicity point is formed where s'' is located initially. Otherwise, on the third activation, by line 31 of algorithm 1, r moves to its first clockwise neighbor. Now, throughout this, s can see the leading angle of r, so it does not move unless it sees a multiplicity point somewhere else. Hence, a multiplicity point is formed where s is located initially when r reaches there. □

Theorem 5.1. *Let $\mathcal{C}$ be a rotationally asymmetric configuration with no multiplicity point, then after finite execution of Algorithm 1 at least one multiplicity point will be created.*

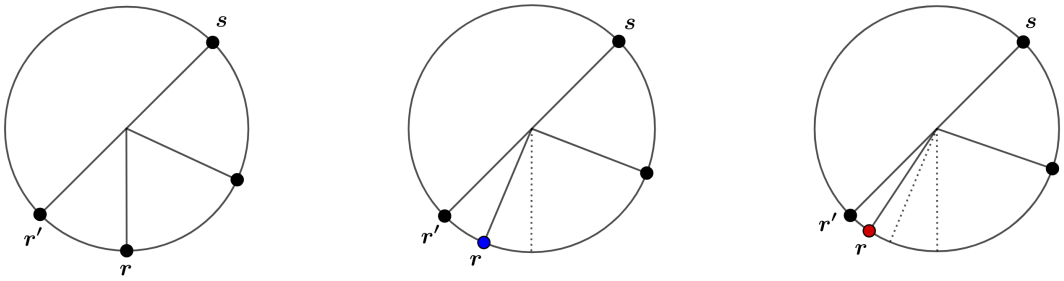

Figure 5.4: Movement of one confused leader (r) in type configuration-C where initial color of r is `off`, blue color indicates `moveHalf` and red color indicates `moveMore`

Proof. Let $\mathcal{C}$ be a rotationally asymmetric configuration with no multiplicity point. Then there can be three exhaustive possible configurations, Configuration-A, Configuration-B, and Configuration-C. From lemma 5.1, lemma 5.2, and lemma 5.3 we can say that, for any of these configurations, at least one multiplicity point will form. □

Lemma 5.4. *From any rotationally asymmetric configuration with no multiplicity point, within finite execution of Algorithm 1, the robots can form at most two multiplicity points, and then all robots gather at a point on the circle.*

Proof. From Theorem 5.1, we can say that at least one multiplicity point will form by the robots on the circle. We will now show that, from any rotationally asymmetric initial configuration with no multiplicity point, at most two multiplicity points will be formed by finite execution of Algorithm 1. From the two multiplicity points all robots will eventually gather at a point on the circle.

Configuration-A

Suppose the initial configuration is of type Configuration-A. Then the initial configuration contains only one expected (true) leader, say L. The expected leader is either a sure leader or a confused leader who finds its clockwise first neighbor safe to move. Let s be the clockwise first neighbor of L. Let r be the robot (if exists) antipodal to L. On getting activated, L starts moving toward

s. In the initial configuration, all robots but L are follower robots so, they don't move until L has not started moving. While L is approaching s, the leading angle of L is the strictly smallest angle in the configuration, because the moving speed of each robot is the same, and in the initial configuration, the leading angle of L is the smallest angle of the configuration. While L is approaching s, the leading angle of L is also visible to s. Hence, s remains a follower robot and does not move, while L is moving toward s. Thus, a multiplicity point will be formed where s is located.

Now, we investigate the possible location where another multiplicity point will be formed. If the antipodal position of s is empty in the initial configuration, then all robots can see the leading angle of L, when L is moving toward s. Thus, all robots except L remain followers and do not move until L reaches s. Hence, only one multiplicity point is formed at the location of s, and also, all robots can see the multiplicity point. After the formation of the multiplicity point, according to the proposed algorithm (line $35-37$ in algorithm 1), the neighbor robots of the multiplicity point reach their location one by one. Hence, after a finite time, all robots gather at the multiplicity point.

Now, suppose, the antipodal position of s is occupied by a robot say, r'. Note that, there cannot be any robot in the interval (r, r') in the initial configuration, otherwise L wouldn't be the true leader of the initial configuration. Now, when L is moving toward s, only r' cannot see the leading angle of L. So r' can become an expected leader, precisely, a confused leader at this time. Also, if r' becomes a confused leader then it must find its first clockwise neighbor, say s', safe to move. Hence, r' might decide to move toward s'. Thus, a multiplicity point can be formed where s' is located in the initial configuration. Since s and s' are not antipodal to each other, the two formed multiplicity points cannot be antipodal to each other. The rest of the robots only move to reach a multiplicity point, thus no new multiplicity point is formed.

Next, we show that, if the multiplicity point formed first at the location of s then it always remains a multiplicity point. As we discussed, s does not decide to move if it is activated while L is approaching s. Also, after the formation of

the multiplicity point at s, neither L nor s moves from that position. A robot getting activated while located at a multiplicity point only decides to leave the multiplicity point through the proposed algorithm (the line $38-40$ of algorithm 1). But the if condition at line 42 is not true for a robot situated at the multiplicity point formed at the initial position of s, because the other possible multiplicity point is at more than π distance from it.

Except for r', all robots can see the leading angle of L when L is approaching s. And after a multiplicity is formed, all robots but r' can see the multiplicity point at s, and we show that this multiplicity point remains. So, none of these robots will move with the intention to form a new multiplicity point. They only move with the purpose of reaching a multiplicity point. Now, the only robot that might move to form a multiplicity point is r'.

If r' moves to form a multiplicity point then after the move either a multiplicity is formed where s' is located in the initial configuration, or before r' reaches, s' decides to move at the multiplicity point where s is located. So, for the latter case, the multiplicity point is not ultimately formed at the initial location of s'. And only one multiplicity point is formed at s which is visible to all the remaining robots. So all the robots are gathered after a finite time. Now, for the former case, two multiplicity points will be formed. The multiplicity point at s is visible to all because the antipodal position is vacated when r' moves from there. But the other multiplicity point might not be visible by a robot, say s'', situated at the antipodal position of s'. But anyway, all the robots eventually reach a multiplicity point. Here, $cwAngle(s', s) < \frac{\pi}{2}$. So, if a robot at the multiplicity points s' is activated it reaches s. Now for the robot at s'', since the multiplicity point at s is closest to s'', so after finite time it reaches there as well. Hence after a finite time, all robots gather at the location of s.

If r' does not move to form a multiplicity point then all robots but r' can see the multiplicity point at s. They all move to the multiplicity point eventually. If r' does not move at all while all the robots are approaching the multiplicity point, then after all the robots reached the multiplicity point at s, then for r', else condition at line 42 of the algorithm becomes true. Then it moves $\frac{\pi}{2}$ and

after the move when it wakes up it can see the multiplicity point. And then it will reach the multiplicity point which completes the gathering.

Configuration-B

One sure leader and one confused leader: Let L and r be respectively the sure leader and confused leader of the initial configuration. Let the first clockwise neighbors of L and r be s and r' respectively. Let l be the location of L in the initial configuration.

Case-I: (When r sees its first clockwise neighbor safe) Let first L and r are not antipodal. Leading angles of r and L may be equal or not. Let first the leading angles are not the same. In this case, the position of r must be in the antipodal of s. So if L starts its move first, in that current configuration r remains an expected leader, and its neighbor is safe. So r moves to its clockwise neighbor. Here without r, all followers can see the angle $cwAngle(L, s)$. So, when L starts its move, then $cwAngle(L, s)$ is strictly the smallest angle of the configuration. So, no other follower robots will move except r until a multiplicity point is formed at s. Now, let r start its move first, then it may happen that L is either elected as a sure leader or, becomes a follower. So if L is elected as a sure leader and starts its move then two multiplicity points will form. Here, without s and the antipodal position of r', all followers can see the leading angle of r, but s and r' will not move as it can see the robot L as leader.

If leading angles are the same then r may or may not be in the antipodal of s. Let r is at the antipodal of s. So, if L starts its move first, then in that current configuration, r will be elected as confused leader. So when it moves to its safe neighbor, two non antipodal multiplicities are formed. But, if r starts its move first, and after that L wakes up then L will not be an expected leader. Now, if L wakes up earlier but does not start its move then two multiplicity points will form when it moves. Also, if r is not at the antipodal of s, then r and L both can see the leading angle of each other. So, if anyone starts their move and after that, the other one wakes up, then the other one will not be an expected leader.

Thus if both L and r are in the move phase then they move to their neighbors forming two multiplicity points.

If L and r are antipodal but the neighbors are safe (i.e neighbors are not antipodal) then if L starts its move earlier, and after that r wakes up, then r will not be a confused leader. But if r is in the move phase then, when it reaches its neighbor, two non antipodal multiplicity points will form. If r starts its move earlier then L may become a follower or confused leader. So if L is elected as an expected leader, and its neighbor is safe then two multiplicity points will form. But after r starts its move it may happen that L is elected as a confused leader and L sees that its neighbor is unsafe. Then L will change its state to `moveHalf` and move to the middle of its leading angle. But as r is in its move phase, so it will reach r'. Now, when L wakes up next, it will see its neighbor is not antipodal. So L changes its state to `terminate`. In that case, only one multiplicity will form. We can conclude that in all the above mentioned cases, if two multiplicity points are formed, they are not antipodal.
Observe that in all the above cases, if two multiplicity points are formed then they are exactly at the position s and r'. If only one multiplicity point is formed at the position between r' or s then all the followers will move to that position, and finally gathering of robots will occur at that multiplicity point. It may happen that there exists a robot in the antipodal position of that multiplicity point. Then either that robot cannot see any robot or it may be elected as an expected leader. If it cannot see any robot, it will move clockwise $\frac{\pi}{2}$, and then it can see the multiplicity point. If it is elected as an expected leader then it will move to its neighbor and so two multiplicity points are formed which are visible to each other. From these two multiplicity points finally, robots gather at the initially formed multiplicity point. If both the two multiplicity points are formed, they are not antipodal so they are visible to each other. Then the follower robots will move to a multiplicity point that is nearest to them. So, finally there exist only two multiplicity points in the configuration and as the clockwise distance of r' is smaller with the other multiplicity point at s, robots at r' finally move to the position at s. Within this move, if a robot sees more than two multiplicity points then it will wait until it sees two multiplicity points in the configuration.

So finally all robots gather at position s.

Case-II: (When r sees its first neighbor is unsafe) Before discussing this case we first establish an observation.

Let r_1 and r_2 be two robots where r_1 is moving clockwise, and r_2 takes a snapshot at an LCM cycle when r_1 is at the antipodal position of r_2 in its move. Let r_2 decide to move an angular distance θ clockwise. Since we assumed that for a robot, the Look and Compute phase together in an LCM cycle takes a nonzero time, so if r_1 keeps on moving throughout the LCM cycle of r_2 then r_1 travels an angle that is strictly greater than θ.

When r sees its first neighbor is unsafe, then it will move to the midpoint of its leading angle changing its state to `moveHalf`. After this move, if r can see the L robot, it will move back to its initial position by changing its state to `terminate`. But if in the current configuration, r cannot see the L robot then L must be in the antipodal of r. So, r changes its state to `moveMore`, and move to the midpoint of its leading angle of that current configuration. From the above observation, we can conclude that after completing this move, r can see either the robot L at the antipodal arc (i.e., the arc from l to the location of s in the clockwise direction) of its leading angle or a multiplicity point at s. So r changes its state to `terminate`, and it will stop at that position. Thus, in this case, only one multiplicity point will form when the L robot moves to its neighbor s. So from this one multiplicity point, the gathering of robots will occur at s, in the same manner as we mentioned above in case-I.

Two confused leaders: Let L be the confused leader which is the true leader of the configuration. r be another confused leader of the configuration. s and r' be the first clockwise neighbors of L and r respectively. Let A be the antipodal position of L and B is the antipodal position of s. We show in Proposition 5.2 that L and r cannot be antipodal of each other. The position of r will be in $[B, L)$, and r cannot be in (A, B). If r is in (A, B) then, as it is confused its antipodal must exist, but that antipodal position will be in (L, s) which is a contradiction as s is the first clockwise neighbor of L. We show in lemma 5.2 that r will find

its clockwise first neighbor safe. When L sees its clockwise neighbor is not safe and in its $C_0(L)$ configuration another confused leader exists, then L will not move. But if L also sees its clockwise neighbor is safe then it will move to its clockwise neighbor. So as we have discussed thoroughly in the above case-I, either one or two multiplicity points will form on the circle. So from that one or two multiplicity points all robots finally gather at a point on the circle similarly as we argued above in case-I.

Configuration-C

If the initial configuration is $\mathcal{C}$, then initially the configuration contains only one expected leader, say r. The expected leader is a confused leader, and it finds its first clockwise neighbor unsafe. In this case, according to our algorithm, r first decides to move clockwise $\theta/2$ distance after changing its state to `moveHalf`, where θ is the leading angle of r initially. Let r' be the clockwise first neighbor of r. First, suppose the antipodal position of r' is unoccupied initially. After r starts its move, the leading angle of r is visible to all and, it is the strictly smallest angle in the configuration. Thus, while this move is on and till when r gets activated next time, all other robots remain followers, and thus do not move. Next, when r gets activated, according to the proposed algorithm, r again moves clockwise an angular distance $\theta/4$, changing the state to `moveMore`. While this move is going on, still the leading angle of r is visible to all and, it is the strictly smallest angle in the configuration. Thus, while this move is on and till when r gets activated next, all other robots remain followers. Thus they do not move. Next, when r gets activated, according to the proposed algorithm, r moves to its first clockwise neighbor. Thus, one multiplicity point is formed where r' is located. This multiplicity point is visible to all the remaining robots. Hence, following the proposed algorithm, all robots eventually gather at this multiplicity point.

Secondly, let the antipodal position of r' is occupied by a robot, say s. Then while r is moving, s can get activated and become a confused leader. Then s decides to start its move. When r gets activated, it changes its state to `moveHalf` and starts

to move clockwise $\theta/2$ angular distance. While this move is going on, s might become a confused leader. If so, then s must find its clockwise first neighbor to be safe. So s would move to its clockwise neighbor. If s does not move at all, then similarly from the argument of the last paragraph, r moves to its clockwise neighbor forming a multiplicity point there. Otherwise, if s starts to move, then if r gets its snapshot, while s is moving, r will find its clockwise neighbor is not antipodal. Hence, r does not move to form the multiplicity point. And only one multiplicity point will be formed at the position of the first clockwise neighbor, say s', of s. If the first multiplicity point is formed at the position of r', then another multiplicity point can be formed at the position of s'. Note that, both the multiplicity points are not antipodal to each other.

If one multiplicity point is formed, then following the argument from the initial Configuration, eventually, all robots gather at the multiplicity point. If two multiplicity point is formed then according to the algorithm eventually, all the robots reach one of the multiplicity points. Since $cwAngle(s', r') < \pi$, according to the proposed algorithm, eventually, robots at the multiplicity point s', gather at the multiplicity point where r' is located. Finally, all robots gather at the multiplicity point at r'.

□

Hence, we can conclude the following theorem.

Theorem 5.2. *There exists a gathering algorithm that gathers any set of robots with finite memory and π visibility from any initial rotationally asymmetric configuration under asynchronous scheduler.*

5.6 Concluding Remarks

In this work, we present a gathering algorithm of robots with finite memory on a circle under an asynchronous scheduler with visibility π. Robots are initially at distinct positions on the circle, forming any rotationally asymmetric configuration. We assume that each robot has finite persistent memory. For future

studies on this problem, it will be interesting, if one can give a gathering algorithm when robots are oblivious or the visibility is less than π. Here we consider the non-standard asynchronous scheduler, so it will be interesting if one can give an algorithm under standard asynchronous scheduler.

For oblivious robots, the gathering algorithm of this chapter will not work under asynchronous scheduler, as the confused leader can not distinguish whether its antipodal robot exists or not by moving half of the angle with its neighbor. But we only consider here the clockwise movement of the robots. So if one robot moves counterclockwise then there may exist a gathering algorithm. But by the counterclockwise move, it is not easy to design such an algorithm, where two non antipodal multiplicity points form while maintaining the asymmetric configuration. So, it is an open problem to show, if there exists any gathering algorithm with the $\mathcal{OBLOT}$ robot model under an asynchronous scheduler.

Chapter 6

Conclusion

In this chapter, we conclude the thesis by subsuming all the technical results from the previous chapters and also highlight some interesting directions for future research.

In Chapter 2, we studied algorithm for Arbitrary Pattern Formation (Apf) problem on an infinite rectangular grid by a robot swarm starting from any asymmetric initial configuration in classical $\mathcal{OBLOT}$ robotic model. This work gives an algorithm for the Apf problem in $\mathcal{OBLOT}$ model which is asymptotically move optimal. Suppose the dimensions of the smallest enclosing rectangles of the initial configuration of the robots and the pattern to be formed are respectively $m \times n$ and $M \times N$. Let $\mathcal{D} = \max\{m, n, M, N\}$ and $\mathcal{D}' = \max\{\mathcal{D}, k\}$ where k is the total number of robots, then the algorithm uses total $O(\mathcal{D}'^2)$ movements to solve the Apf problem. Unfortunately the move optimal algorithm in $\mathcal{OBLOT}$ model is not time optimal, so studying for an algorithm in $\mathcal{OBLOT}$, which is both move optimal and time optimal, could be a possible direction from this work. Another direction of open problem is, if we consider the symmetric configuration, then in which types of configuration, we can provide the algorithm by which arbitrary pattern formation is solvable.

Open Problem 1. *Provide deterministic algorithm which is time optimal by oblivious robots and characterize the symmetric configurations from which* Arbitrary Pattern Formation *is solvable on infinite rectangular grid.*

In Chapter 3, we studied the algorithm for Arbitrary Pattern Formation (Apf) problem on an infinite rectangular grid by a robot swarm starting from any asymmetric initial configuration in classical $\mathcal{LUMI}$ robotic model. This work proposed algorithm for the Apf problem in $\mathcal{LUMI}$ model which is asymptotically time optimal and move optimal as well. Suppose the dimensions of the smallest enclosing rectangles of the initial configuration of the robots and the pattern to be formed are respectively $m \times n$ and $M \times N$. Let $\mathcal{D} = \max\{m, n, M, N\}$ and $\mathcal{D}' = \max\{\mathcal{D}, k\}$ where k is the number of robots, then the algorithm takes total $O(\mathcal{D}')$ epoch time and total $O(\mathcal{D}'^2)$ moves to solve the Apf problem. The algorithm in this chapter which is time optimal and move optimal is in $\mathcal{LUMI}$ model, so studying for an algorithm in $\mathcal{OBLOT}$, which is both move optimal and time optimal, could be a possible direction from this work. Also another version of Apf problem where the initial or the target configuration of robots can have multiplicity points can consider for further research.

Open Problem 2. *Can* Arbitrary Pattern Formation *be solved with* $\mathcal{OBLOT}$ *robots under asynchronous scheduler which is time and move optimal on an infinite rectangular grid? Another one open problem is to solve* Apf *by considering the multiplicity points in the initial or target configuration.*

In Chapter 4, in this work, we claim that k oblivious, identical, autonomous robots are dispersed on the vertices of the infinite rectangular grid and robots can do two different tasks namely gathering and arbitrary pattern formation simultaneously on an infinite rectangular grid under the asynchronous scheduler. In this work, we have provided a collision-less distributed algorithm following which two teams of oblivious robots with weak multiplication detection ability. Here we assume that the initial configuration is asymmetric. There are two teams of oblivious robots where one team is $\mathcal{T}_g$ and the other team is $\mathcal{T}_{Apf}$. A robot only knows to which team it belongs, but a robot does not know to which team another robot belongs to. The goal of the robots of $\mathcal{T}_g$ is to meet all the robots of this team to a point on an infinite rectangular grid. Another team is $\mathcal{T}_{Apf}$ where the goal of this team is to form a particular pattern that is given as input. Here we assume the grid is infinite and the visibility is full. So for further work, we can extend

this work by assuming limited visibility and also when the grid is finite.

Open Problem 3. *Investigate* Arbitrary Pattern Formation *and* Gathering *by two different teams of oblivious robots when the visibility is limited and also in finite grid. Another direction of research involves exploring the capabilities of robots to perform more than two tasks simultaneously on a grid network.*

In Chapter 5, we present a gathering algorithm of robots with finite memory on a continuous circle under an asynchronous scheduler with visibility π. Robots are initially at distinct positions on the circle, forming any rotationally asymmetric configuration. We assume that each robot has finite persistent memory. For future studies on this problem, it will be interesting, if one can give a gathering algorithm when robots are oblivious or the visibility is less than π. Here we consider the non-standard asynchronous scheduler, so it will be interesting if one can give an algorithm under standard asynchronous scheduler.

For oblivious robots, the gathering algorithm of this chapter will not work under asynchronous scheduler, as the confused leader can not distinguish whether its antipodal robot exists or not by moving half of the angle with its neighbor. But we only consider here the clockwise movement of the robots. So if one robot moves counterclockwise then there may exist a gathering algorithm. But by the counterclockwise move, it is not easy to design such an algorithm, where two non antipodal multiplicity points form while maintaining the asymmetric configuration. So, it is an open problem to show, if there exists any gathering algorithm with the $\mathcal{OBLOT}$ robot model under an asynchronous scheduler.

Open Problem 4. *Investigate* Gathering *by $\mathcal{OBLOT}$ robots under asynchronous scheduler with π visibility. Another direction is to investigate the gathering of robots when the visibility is less than π.*

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. *Comput. Sci.*, 22(1), 2021. doi:10.7494/csci.2021.22.1.3840.

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

[4] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita. Distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Transactions on Robotics and Automation*, 15(5):818–828, 1999. doi:10.1109/70.795787.

[5] Subhash Bhagat, Abhinav Chakraborty, Bibhuti Das, and Krishnendu Mukhopadhyaya. Gathering over meeting nodes in infinite grid*. *Fundam. Informaticae*, 187(1):1–30, 2022. doi:10.3233/FI-222128.

[6] Subhash Bhagat, Paola Flocchini, Krishnendu Mukhopadhyaya, and Nicola Santoro. Weak robots performing conflicting tasks without knowing who is

in their team. 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 29:1–29:6. ACM, 2020. doi:10.1145/3369740.3369794.

[7] Subhash Bhagat, Krishnendu Mukhopadhyaya, and Srabani Mukhopadhyaya. Computation under restricted visibility. In 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*, pages 134–183. Springer, 2019. doi:10.1007/978-3-030-11072-7_7.

[8] Kaustav Bose, Ranendu Adhikary, Sruti Gan Chaudhuri, and Buddhadeb Sau. Crash tolerant gathering on grid by asynchronous oblivious robots. *CoRR*, abs/1709.00877, 2017. URL: http://arxiv.org/abs/1709.00877, arXiv:1709.00877.

[9] Kaustav Bose, Ranendu Adhikary, Manash Kumar Kundu, and Buddhadeb Sau. Arbitrary pattern formation by opaque fat robots with lights. *CoRR*, abs/1910.02706, 2019. URL: http://arxiv.org/abs/1910.02706, arXiv:1910.02706.

[10] 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. doi:10.1016/j.tcs.2020.02.016.

[11] Kaustav Bose, Archak Das, and Buddhadeb Sau. Pattern formation by robots with inaccurate movements. In Quentin Bramas, Vincent Gramoli, and Alessia Milani, editors, *25th International Conference on Principles of Distributed Systems, OPODIS 2021, December 13-15, 2021, Strasbourg, France*, volume 217 of *LIPIcs*, pages 10:1–10:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.OPODIS.2021.10.

[12] Kaustav Bose, Manash Kumar Kundu, Ranendu Adhikary, and Buddhadeb Sau. Arbitrary pattern formation by asynchronous opaque robots with lights. *Theor. Comput. Sci.*, 849:138–158, 2021. doi:10.1016/j.tcs.2020.10.015.

[13] Sébastien Bouchard, Yoann Dieudonné, and Bertrand Ducourthial. Byzantine gathering in networks. *CoRR*, abs/1504.01623, 2015.

[14] Quentin Bramas and Sébastien Tixeuil. Probabilistic asynchronous arbitrary pattern formation (short paper). 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 88–93, 2016. doi:10.1007/978-3-319-49259-9_7.

[15] Quentin Bramas and Sébastien Tixeuil. Arbitrary pattern formation with four robots. 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 333–348. Springer, 2018. doi:10.1007/978-3-030-03232-6_22.

[16] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *CoRR*, abs/1012.0009, 2010.

[17] Serafino Cicerone, Alessia Di Fonso, Gabriele Di Stefano, and Alfredo Navarra. Arbitrary pattern formation on infinite regular tessellation graphs. *CoRR*, abs/2010.14152, 2020. URL: https://arxiv.org/abs/2010.14152, arXiv:2010.14152.

[18] 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. doi:10.1007/s00446-018-0325-7.

[19] Serafino Cicerone, Gabriele Di Stefano, and Alfredo Navarra. Embedded pattern formation by asynchronous robots without chirality. *Distributed Comput.*, 32(4):291–315, 2019. doi:10.1007/s00446-018-0333-7.

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

[21] Jurek Czyzowicz, Adrian Kosowski, and Andrzej Pelc. How to meet when you forget: log-space rendezvous in arbitrary graphs. *Distributed Comput.*, 25(2):165–178, 2012.

[22] Jurek Czyzowicz, Adrian Kosowski, and Andrzej Pelc. Deterministic rendezvous of asynchronous bounded-memory agents in polygonal terrains. *THEORY OF COMPUTING SYSTEMS*, 52(2):179–199, 2013.

[23] Gianlorenzo D'Angelo, Gabriele Di Stefano, Ralf Klasing, and Alfredo Navarra. Gathering of robots on anonymous grids and trees without multiplicity detection. *Theor. Comput. Sci.*, 610:158–168, 2016. doi:10.1016/j.tcs.2014.06.045.

[24] Shantanu Das, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Forming sequences of patterns with luminous robots. *IEEE Access*, 8:90577–90597, 2020. doi:10.1109/ACCESS.2020.2994052.

[25] 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. doi:10.1007/s00446-014-0220-9.

[26] Shantanu Das, Nikos Giachoudis, Flaminia L Luccio, and Euripides Markou. Gathering of robots in a grid with mobile faults. In *International Conference on Current Trends in Theory and Practice of Informatics*, pages 164–178. Springer, 2019.

[27] Xavier Défago, Maria Gradinariu Potop-Butucaru, Julien Clément, Stéphane Messika, and Philippe Raipin Parvédy. Fault and byzantine tolerant

self-stabilizing mobile robots gathering - feasibility study -. *CoRR*, abs/1602.05546, 2016.

[28] Bastian Degener, Barbara Kempkes, Tobias Langner, Friedhelm Meyer auf der Heide, Peter Pietrzyk, and Roger Wattenhofer. A tight runtime bound for synchronous gathering of autonomous robots with limited visibility. In *Proceedings of the Twenty-Third Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '11, page 139–148, New York, NY, USA, 2011. Association for Computing Machinery. doi:10.1145/1989493.1989515.

[29] 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.

[30] Ofer Feinerman, Amos Korman, Shay Kutten, and Yoav Rodeh. Fast rendezvous on a cycle by agents with different speeds. In *Distributed Computing and Networking - 15th International Conference, ICDCN 2014, Coimbatore, India, January 4-7, 2014. Proceedings*, pages 1–13, 2014. doi:10.1007/978-3-642-45249-9_1.

[31] 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.

[32] Paola Flocchini, Ryan Killick, Evangelos Kranakis, Nicola Santoro, and Masafumi Yamashita. Gathering and election by mobile robots in a continuous cycle. In Pinyan Lu and Guochuan Zhang, editors, *30th International*

Symposium on Algorithms and Computation, ISAAC 2019, December 8-11, 2019, Shanghai University of Finance and Economics, Shanghai, China, volume 149 of *LIPIcs*, pages 8:1–8:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ISAAC.2019.8.

[33] 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.

[34] 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. doi:10.1016/j.tcs.2005.01.001.

[35] 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. doi:10.1016/j.tcs.2008.07.026.

[36] 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.

[37] Pritam Goswami, Avisek Sharma, Satakshi Ghosh, and Buddhadeb Sau. Time optimal gathering of myopic robots on an infinite triangular grid. In Stéphane Devismes, Franck Petit, Karine Altisen, Giuseppe Antonio Di Luna, and Antonio Fernández Anta, editors, *Stabilization, Safety, and Security of Distributed Systems - 24th International Symposium, SSS 2022, Clermont-Ferrand, France, November 15-17, 2022, Proceedings*, volume 13751 of *Lecture Notes in Computer Science*, pages 270–284. Springer, 2022. doi:10.1007/978-3-031-21017-4_18.

[38] Samuel Guilbault and Andrzej Pelc. Gathering asynchronous oblivious agents with local vision in regular bipartite graphs. 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 162–173. Springer, 2011. doi:10.1007/978-3-642-22212-2_15.

[39] Rory Hector, Gokarna Sharma, Ramachandran Vaidyanathan, and Jerry L. Trahan. Optimal arbitrary pattern formation on a grid by asynchronous autonomous robots. In *2022 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2022, Lyon, France, May 30 - June 3, 2022*, pages 1151–1161. IEEE, 2022. doi:10.1109/IPDPS53621.2022.00115.

[40] Evan Huus and Evangelos Kranakis. Rendezvous of many agents with different speeds in a cycle. In Symeon Papavassiliou and Stefan Ruehrup, editors, *Ad-hoc, Mobile, and Wireless Networks - 14th International Conference, ADHOC-NOW 2015, Athens, Greece, June 29 - July 1, 2015, Proceedings*, volume 9143 of *Lecture Notes in Computer Science*, pages 195–209. Springer, 2015. doi:10.1007/978-3-319-19662-6_14.

[41] Manash Kumar Kundu, Pritam Goswami, Satakshi Ghosh, and Buddhadeb Sau. Arbitrary pattern formation by asynchronous opaque robots on infinite grid. *CoRR*, abs/2205.03053, 2022. arXiv:2205.03053, doi:10.48550/arXiv.2205.03053.

[42] Manash Kumar Kundu, Pritam Goswami, Satakshi Ghosh, and Buddhadeb Sau. Arbitrary pattern formation by opaque fat robots on infinite grid. *International Journal of Parallel, Emergent and Distributed Systems*, pages 1–29, jun 2022. URL: https://doi.org/10.1080%2F17445760.2022.2088750, doi:10.1080/17445760.2022.2088750.

[43] Tamás Lukovszki and Friedhelm Meyer auf der Heide. Fast collisionless pattern formation by anonymous, position-aware robots. In Marcos K. Aguilera, Leonardo Querzoni, and Marc Shapiro, editors, *Principles of Distributed Systems - 18th International Conference, OPODIS 2014, Cortina d'Ampezzo, Italy, December 16-19, 2014. Proceedings*, volume 8878 of *Lecture Notes in Computer Science*, pages 248–262. Springer, 2014. doi:10.1007/978-3-319-14472-6_17.

[44] Giuseppe Antonio Di Luna, Paola Flocchini, Linda Pagli, Giuseppe Prencipe, Nicola Santoro, and Giovanni Viglietta. Gathering in dynamic rings. *CoRR*, abs/1704.02427, 2017.

[45] Giuseppe Antonio Di Luna, Paola Flocchini, Nicola Santoro, Giovanni Viglietta, and Masafumi Yamashita. Meeting in a polygon by anonymous oblivious robots. *Distributed Comput.*, 33(5):445–469, 2020.

[46] Giuseppe Antonio Di Luna, Ryuhei Uehara, Giovanni Viglietta, and Yukiko Yamauchi. Gathering on a circle with limited visibility by anonymous oblivious robots. In Hagit Attiya, editor, *34th International Symposium on Distributed Computing, DISC 2020, October 12-16, 2020, Virtual Conference*, volume 179 of *LIPIcs*, pages 12:1–12:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.DISC.2020.12.

[47] Debasish Pattanayak, Klaus-Tycho Foerster, Partha Sarathi Mandal, and Stefan Schmid. Conic formation in presence of faulty robots. In Cristina Maria Pinotti, Alfredo Navarra, and Amitabha Bagchi, editors, *Algorithms for Sensor Systems - 16th International Symposium on Algorithms and Experiments for Wireless Sensor Networks, ALGOSENSORS 2020, Pisa, Italy, September 9-10, 2020, Revised Selected Papers*, volume 12503 of *Lecture Notes in Computer Science*, pages 170–185. Springer, 2020. doi:10.1007/978-3-030-62401-9_12.

[48] Pavan Poudel and Gokarna Sharma. Universally optimal gathering under limited visibility. 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 323–340. Springer, 2017. doi:10.1007/978-3-319-69084-1_23.

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

[50] Ichiro Suzuki and Masafumi Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing*, 28(4):1347–1363, 1999. arXiv:https://doi.org/10.1137/S009753979628292X, doi:10.1137/S009753979628292X.

[51] Ichiro Suzuki and Masafurni Yamashita. Distributed anonymous mobile robots - formation and agreement problems. In *Problems, in the Proceedings of the 3rd International Colloquium on Structural Information and Communication Complexity (SIROCCO '96*, pages 1347–1363, 1996.

[52] Ramachandran Vaidyanathan, Gokarna Sharma, and Jerry Trahan. On fast pattern formation by autonomous robots. *Information and Computation*, page 104699, 2021. URL: https://www.sciencedirect.com/science/article/pii/S0890540121000146, doi:https://doi.org/10.1016/j.ic.2021.104699.

[53] Masafumi Yamashita and Ichiro Suzuki. Characterizing geometric patterns formable by oblivious anonymous mobile robots. *Theoretical Computer Science*, 411(26):2433–2453, 2010. URL: https://www.sciencedirect.com/science/article/pii/S0304397510000745, doi:https://doi.org/10.1016/j.tcs.2010.01.037.

[54] 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.

List of Publications

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

- Satakshi Ghosh, Pritam Goswami, Avisek Sharma, 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). https://doi.org/10.1080/17445760.2022.2124411

- Satakshi Ghosh, Avisek Sharma, Pritam Goswami, Buddhadeb Sau: **Oblivious Robots Performing Different Tasks on Grid Without Knowing Their Team Members.** ICARA 2023: 180-186. https://doi.org/10.1109/ICARA56516.2023.10125816

- Satakshi Ghosh, Avisek Sharma, Pritam Goswami, Buddhadeb Sau: **Brief Announcement: Asynchronous Gathering of Finite Memory Robots on a Circle Under Limited Visibility.** SSS 2023: 430-434. https://doi.org/10.1007/978-3-031-44274-2_32

Index