Distributed k-Circle Formation by Mobile Robots

Bibhuti Das

Indian Statistical Institute

November 2023

INDIAN STATISTICAL INSTITUTE

DOCTORAL THESIS

Distributed k-Circle Formation by Mobile Robots

Author

Bibhuti Das

Supervisor

Professor Krishnendu Mukhopadhyaya

A thesis submitted to the Indian Statistical Institute in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Computer Science

Advanced Computing & Microelectronics Unit

Indian Statistical Institute, Kolkata

November 2023

Dedicated to My Parents

and

All My Teachers

Acknowledgements

A few words of appreciation cannot express my gratitude towards all the people who have helped in this journey. I am extremely grateful to my supervisor, Professor Krishnendu Mukhopadhyaya, for his invaluable advice, continuous inspiration, and patience during my PhD journey. I would like to express my deepest gratitude for his immense support and motivation during this long journey. My sincere thanks to my coursework instructors and professors at the ACM Unit. I would like to extend my sincere thanks to the Indian Statistical Institute for providing all the required facilities at every stage of this journey.

I would like to thank Professor Partha Sarathi Mandal for teaching the course on distributed computing at IIT Guwahati. I would like to express my deepest appreciation to Dr. Subhash Bhagat for his collaborative work with me. I am also grateful to Abhinav Chakraborty, my friend, labmate, and collaborator. I would like to express my thanks to Satya Da, Avirup Da, Dibakar Da, Kaustav Da, Subhadeep Da, Sanjana Di, Koustabha, Drimit, and Arun. A special thanks to my dear friends Rathin and Subhojit. I would like to thank Sankar, Sucheta, Sukriti, Surochita, and Debendra.

During this journey, my stay at the RS hostel would not have been so joyful and memorable without my helpful seniors, my dear friends, and my lovely juniors. I would like to mention my seniors: Gopal Da, Satya Da, Pinaki Da, Avisek Da, Biltu Da, Indra Da, Sukrit Da, Tanujit Da, Nadim Da, Sarbendu Da, and Aparajita Di. I cannot thank enough my friends Priyanka, Mainak, Sankar, Anjan, Susanta, Joginder, and Kaushik. I am extremely happy to have juniors Faizan, Gourab, Gourab, Suman, and Meghna.

I am deeply indebted to my father, Malay Kumar Das, and my mother, Lalita Das, for their immense love and support. They have always been my strength in every aspect of my life. A lot of thanks to my elder sisters, Usha and Sudha, for believing in me. I am also thankful to my brother-in-law, Satya Bagchi, for his encouragement during my PhD journey. Lastly, I would like to mention my sweet niece, Archisha.

Bibhuti Das

Bibhuti Das

November, 2023

Abstract

The *k-circle formation* problem asks a group of robots to form disjoint circles. Each circle is restricted to being centered at one of the pre-fixed points given in the plane, and each circle should have exactly k distinct robot positions. In this thesis, we investigate the solvability of the *k-circle formation* by a swarm of mobile robots in a deterministic manner. The robots are autonomous, and they execute Look-Compute-Move (LCM) cycle under a fair asynchronous scheduler. They are anonymous, i.e., they do not have any unique idenitfier, and homogeneous, i.e., they execute the same deterministic algorithm. The robots are assumed to be *oblivious* and *silent* or may have limited persistent memory.

We begin by investigating the *k-circle formation* problem in a setting where the robots have global agreement on the y-axis. In this setting, all the *initial* configurations and values of k for which the *k-circle formation* problem is deterministically unsolvable are characterized. For the remaining configurations and values of k, a deterministic distributed algorithm is proposed that solves the *k-circle formation* problem within finite time. It is shown that if the *k-circle formation* problem is deterministically solvable, then the k-EPF problem (a generalized version of the *embedded pattern formation* problem) can also be solved deterministically.

We proceed by dropping the assumption of global y-axis agreement, where we assume that the robots do not have any agreement on the orientations and directions of any of the axes of a global coordinate system. In this setting, we provide a deterministic solution for the *k-circle formation* problem by characterizing all the deterministically unsolvable configurations.

If the robots are opaque, when three robots are collinear, then the terminal robots cannot see one another. In this setup, we consider two cases, namely, complete knowledge of fixed points and zero knowledge of fixed points. When the robots have complete knowledge of fixed points, a distributed algorithm is proposed that solves *k-circle formation* problem for oblivious and silent robots in a deterministic manner. For robots with zero knowledge of fixed points, a deterministic distributed solution is presented by assuming that the robots have one bit of persistent memory.

In the real world, a robot cannot be dimensionless. We study the *k-circle formation* problem for unit disk robots. We propose a deterministic distributed solution under the assumption of global y-axis agreement. We conclude this thesis by discussing some future research directions related to the *k-circle formation* problem.

Publications

- Journals

  J1 Subhash Bhagat, Bibhuti Das, Abhinav Chakraborty, Krishnendu Mukhopadhyaya: k-Circle Formation and k-epf by Asynchronous Robots. **Algorithms 14(2): 62 (2021)**. Chapter 3 is based on this work.

  J2 Bibhuti Das, Abhinav Chakraborty, Subhash Bhagat, Krishnendu Mukhopadhyaya: k-Circle formation by disoriented asynchronous robots. **Theoretical Computer Science. 916: 40-61 (2022)**. Chapter 4 is based on this work.

  J3 Bibhuti Das, Krishnendu Mukhopadhyaya: k-Circle Formation by Asynchronous Opaque Robots. (Submitted to the journal Theoretical Computer Science, Manuscript Number: TCS-D-23-00770). Chapter 5 is based on this work.

- Conferences

  C1 Bibhuti Das, Krishnendu Mukhopadhyaya: Uniform k-Circle Formation by Fat Robots. **SSS 2023: 359-373**
  Chapter 6 is based on this work.

  C2 Bibhuti Das, Krishnendu Mukhopadhyaya: k-Circle Formation by Oblivious Mobile Robots. **ICDCN 2022: 238-239**

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Contents

1.1 Overview

The study of swarm robotics has received a lot of attention over the last two decades and primarily focuses on systems of multiple autonomous mobile robots (also known as robot swarms). A swarm of mobile robots is a multi-robot system consisting of small and inexpensive mobile robots working together in a cooperative environment to achieve some specific goal. The collective behavior of social animals like ants, bees, and fish serves as inspiration for the behavior of these robots. Each of the robots is assumed to be weak, i.e., equipped with very limited capabilities. The robots cooperate in a distributed manner to complete a task.

One of the motivations behind this research direction is to avoid the difficulty and often high cost of designing and deploying a small number of problem-specific robots that are capable of solving the specific problem. Another common motivation behind building autonomous multi-robot systems is the need to perform different tasks in adverse situations where human intervention is not possible. Such a multi-robot system is designed to work in a decentralized manner so that it can be deployed in adverse and unknown environments. Assuming the robots are inexpensive (and hence produced in large quantities), they can be deployed in harsh and hostile environments. Such a large number of robots have the potential to find applications in many fields like risky and hazardous scenarios, such as in the fields of search and rescue operations [1–3], military operations [4], fire fighting [5,6], agriculture [7], etc. The common distributed models assume relatively weak and simple robots. In particular, these robots are only capable of sensing their immediate surroundings, performing simple computations on the sensed data, and moving towards the computed destination. They follow a simple cycle of sensing, computing, moving, and being inactive. In spite of their limitations, the robots should be able to perform rather complex tasks. In computational terms, the primary focus is to determine the minimal robot capabilities that are necessary to perform the required task. The feasibility of solving different problems depends on each set of assumptions about the capabilities of the robots. There is a trade-off between the model of computation and the solvability of a problem.

Suzuki et al. [8] were the first to study multi-robot systems from a computational point of view. In the research field of distributed computing by mobile entities [9], a large volume of work has been reported over the last two decades that primarily focuses on the computational and complexity issues for a distributed system of mobile entities. These mobile entities are assumed to be deployed in either a discrete domain (*mobile agents*) or a continuous domain (*mobile robots*). The research is still focusing on basic tasks such as *gathering* [10–22], *flocking* [23–27], *pattern formation* [28–37], *scattering* [38–42], etc.

1.2 Computational Model

The classical model of distributed computing by mobile robots models each robot as a point in the Euclidean plane. Each robot has a local coordinate system and sensory capabilities to determine the positions of other robots. Such a distributed system of multiple mobile robots works in a coordinated manner to achieve a specific goal. The primary goal is to find essential capabilities to solve a given problem. The idea is to identify the minimal sets of capabilities that are required for designing such mobile robots. In general, the robots are assumed to be:

- *autonomous*, i.e., they do not have any centralized controller;
- *anonymous*, i.e., they have no unique identifier;
- *oblivious*, i.e., they do not remember anything about past events;
- *homogeneous*, i.e., they execute the same algorithm.
- *silent*, i.e., they do not have any direct explicit communication.

However, some of the reported results have considered heterogeneous robots [43, 44]. In such a model, each group of homogeneous robots is represented by a color from a pre-defined finite set of colors. In the literature, some of the studies [45–47] consider robots with persistent memories and explicit communication capabilities provided by the presence of lights.

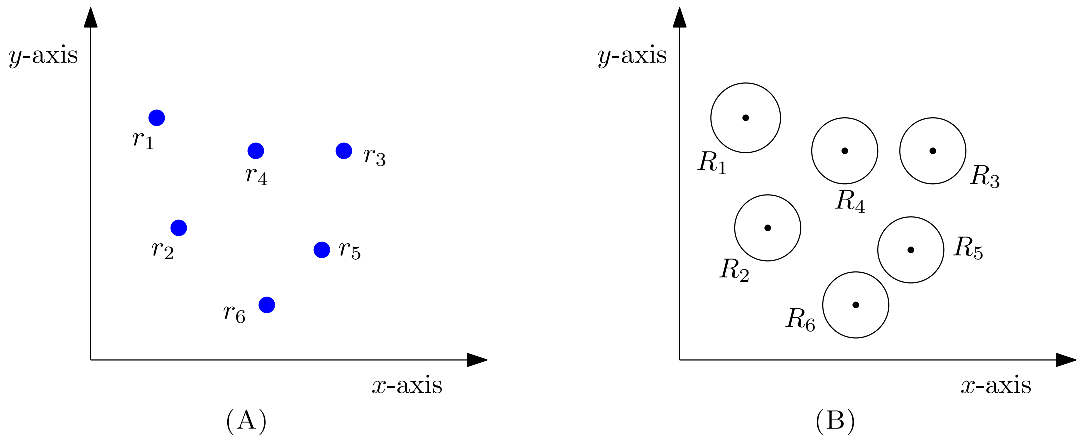


FIGURE 1.1: (A) Blue points represent dimensionless robots. (B) Disks represent fat robots.

1.2.1 Deployment Space

In general, the mobile robots are assumed to be deployed in either a discrete domain, i.e., on the nodes of a graph, or a continuous domain, i.e., in the d-dimensional Euclidean space. In the discrete domain, the robots are allowed to move along the edges of the graph. The movements of the robots are instantaneous, i.e., the robots are not visible on the edges. In the continuous domain, the robots move in the d-dimensional Euclidean space.

1.2.2 Dimension

In the standard model, the robots are assumed to be dimensionless, i.e., they are represented by points in the d-dimensional space (Figure 1.1(A)). However, some of the models have been considered in which the robots are represented by unit disks in the d-dimensional space (Figure 1.1(B)).

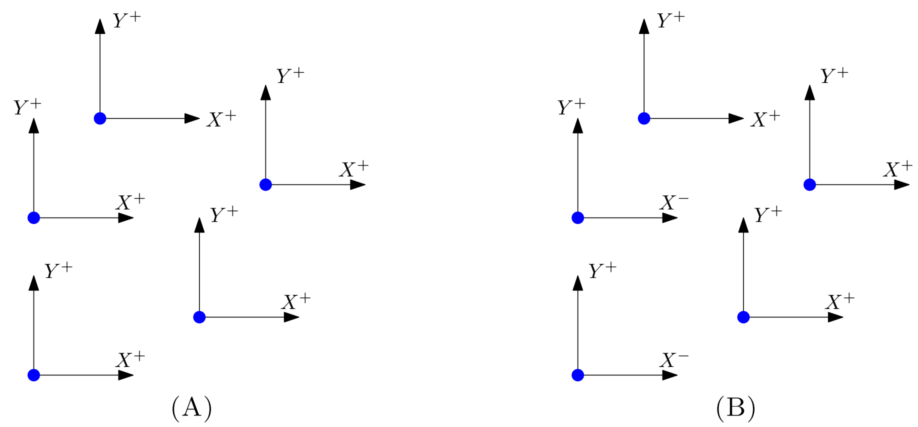


FIGURE 1.2: (A) Full-Axis agreement (B) One-Axis agreement.

1.2.3 Agreement

In general, the robots have their own local coordinate system, whose origin is the position of the robot. They may not have any agreement on the orientations and directions of any of the axes of a global coordinate system. However, in some of the models, the robots are assumed to have some agreement on the global coordinate system. Depending on the type of agreement on the global coordinate system, the following different types of models are common:

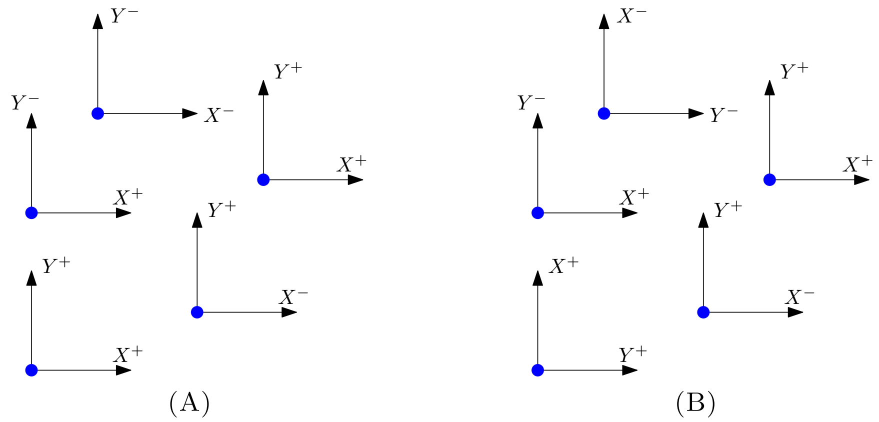


FIGURE 1.3: (A) Direction-Only agreement (B) Axes-Only agreement.

1. **Full-Compass:** The robots have complete agreement on the direction and orientation of both axes of the global coordinate system (Figure 1.2(A)).

2. **Half-Compass:** The robots agree on the direction and orientation of one of the axes of the global coordinate system (Figure 1.2(B)).

3. **Direction-Only:** The robots have agreement on the direction of both axes of the global coordinate system. However, they do not have any agreement on the orientation of any of the global axes (Figure 1.3(A)).

4. **Axes-Only:** The robots have an agreement on the direction of both axes of the global coordinate system. However, they do not have any agreement on the orientation of any of the global axes. In addition, the robots do not agree on which of the two axes is the x-axis (Figure 1.3(B)).

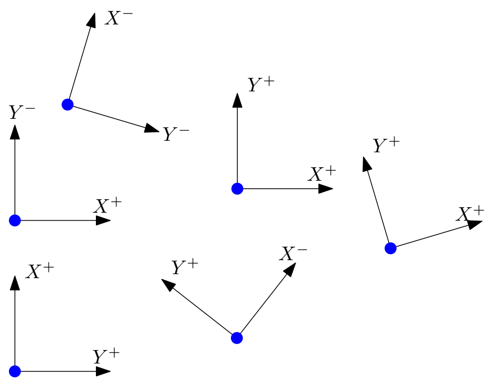


FIGURE 1.4: No-Compass agreement.

5. **No-Compass:** The robots do not have any agreement on the orientations and directions of any of the axes of a global co-ordinate system (Figure 1.4).

Note that the robots may not share a common unit distance [48] or a common origin even in the full-compass model. Furthermore, the robots may not have any agreement on a common clockwise or counter-clockwise directions. The robots are said to have a common *chirality*, if they agree on a common clockwise direction.

1.2.4 Visibility

The robots are assumed to be equipped with sensors (known as the visibility of a robot) that allow them to detect the positions of other robots. The visibility of a robot allows for an implicit way of communicating with other robots. In general, the robots are assumed to have *unlimited* visibility, i.e., they can observe the entire domain. However, there are some restricted visibility models, as described below:

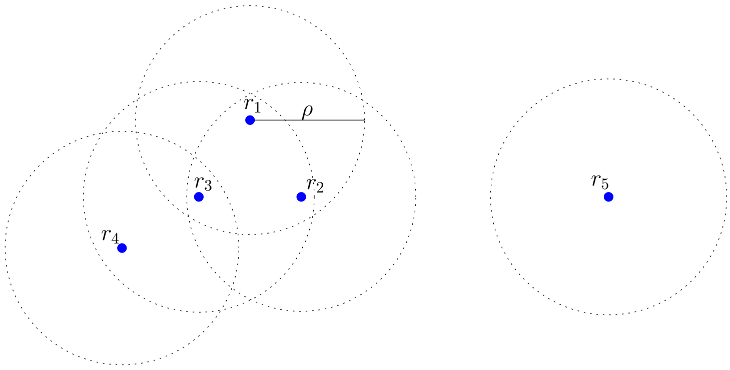


FIGURE 1.5: Limited visibility of robots.

1. **Limited visibility:** The robots have a sensing range. They can detect the positions of other robots up to a fixed radius around them. In Figure 1.5, each robot can see another robot that lies within ρ distance from its position. The robot r_3 is visible to all the robots, and r_4 is only visible to r_3. The robot r_5 is not visible to any other robots, namely r_1, r_2, r_3 and r_4.

2. **Obstructed visibility:** In general, the robots are assumed to be *transparent*, i.e., their visibility is not blocked by the presence of other robots. Under the *obstruted* visibility model, the robots are assumed to be *opaque*, i.e., if three robots are collinear, then the corner robots cannot see one another. In Figure 1.6, the robots r_1, r_3 and r_4 are collinear. The robot r_3 can see both r_1 and r_4 whereas r_1 and r_4 cannot see one another.

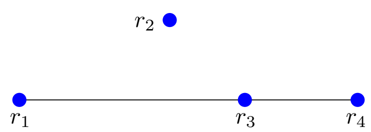


FIGURE 1.6: Obstructed visibility of robots.

1.2.5 Computational Cycle

The state of a robot can be either active or inactive. Each robot operates in Look-Compute-Move (LCM) cycle. An active robot observes its surroundings, computes a destination point, and moves towards the computed destination point.

1. **Look:** The robot takes a snapshot of the domain within its visibility range. The snapshot is instantly taken in its own local coordinate system.

2. **Compute:** It computes a destination point based on the snapshot taken in its *look* phase. The computed destination point may be its current location.

3. **Move:** The robot moves towards its destination point in its *move* phase in a straight line. A moving robot can be seen anywhere on the line segment between its current location and destination point at a particular instant of time. If the destination point is the current location, then the robot makes null movement. The following types of motion are considered:

   (a) **Rigid:** The robot is guaranteed to reach its destination point.

   (b) **Non-rigid:** The adversary can stop the robot before it reaches its destination point. However, it is assumed that the distance traveled by a robot is not infinitesimally small. This is to ensure that if a robot keeps computing the same destination point, then it will reach its destination point within a finite time. Suppose $d > 0$ denotes the distance between the destination point and the robot. There exists a fixed but unknown $\delta > 0$ such that if $d > \delta$, then the robot is guaranteed to move at least δ amount towards its destination. If $d < \delta$, the robot is guaranteed to reach the destination point.

1.2.6 Scheduler

It is assumed that a scheduler determines the durations of inactivity phases and LCM cycles for all the robots. The scheduler is assumed to be fair, i.e., each robot is activated infinitely often. This prevents the scenario where the sheduler always forces one robot to remain idle. Additionally, it is assumed that each robot completes its LCM cycle within a finite time. Otherwise, the scheduler can make a robot continue an LCM cycle indefinitely. The following types of schedulers are commonly used:

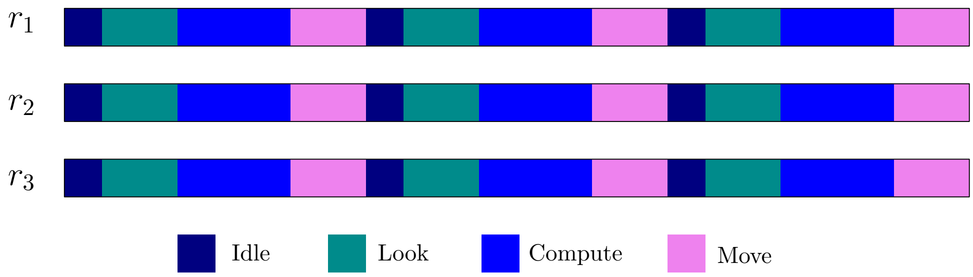


FIGURE 1.7: FSYNC scheduler

1. **Fully-synchronous (FSYNC):** The robots have a common notion of time. All the robots are activated simultaneously and perform all operations synchronously (Figure 1.7).

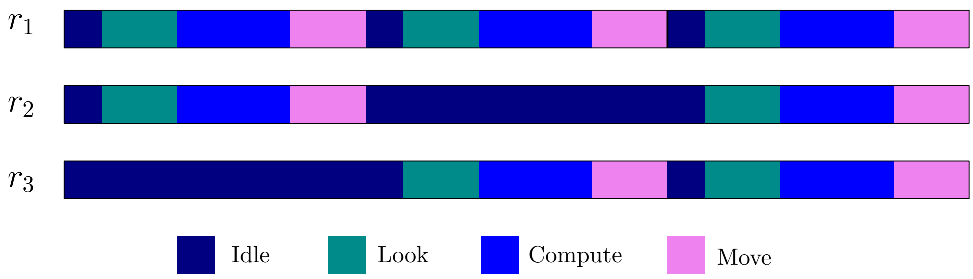


FIGURE 1.8: SSYNC scheduler

2. **Semi-synchronous (SSYNC):** It is similar to the FSYNC scheduler, with the only difference that not all the robots are activated in each round (Figure 1.8). In each round, a subset of robots are activated.

3. **Asynchronous (ASYNC):** The robots do not have a common notion of time. They are activated independently, and the duration of each *look*, *compute*, *move*, and inactivity phase is finite but unbounded (Figure 1.9). During the *look* phase

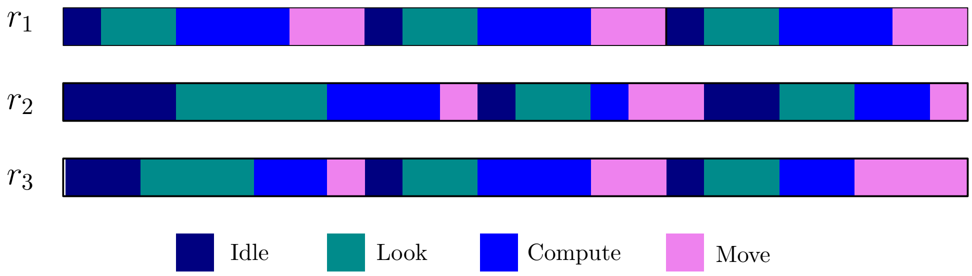


FIGURE 1.9: ASYNC scheduler

of an active robot, some other robot may be in *move* phase. As a result, it might start its *move* phase, considering an outdated perceived snapshot.

1.2.7 Faulty Robots

A robot may become faulty at any arbitrary point of time during an execution. A faulty robot deviates from its specified behavior; for example, it may stop moving. However, the robots do not have the capability to detect whether other robots are faulty or not. The following types of faults are being considered in the model:

1. **Transient Fault:** A robot becomes faulty due to corruption of its memory for a temporary point of time. If the robots are assumed to be *oblivious*, then the distributed system is self-stabilizing against transient faults.

2. **Crash Fault:** A robot crashes and stops working forever. It stops moving and remains in the environment.

3. **Byzantine Fault:** This type of fault occurs when a robot starts to behave arbitrarily. For example, a faulty robot can stop moving, move to arbitrary locations, or prevent deliberately non-faulty robots from moving.

1.2.8 Multiplicity Detection

If the robots are assumed to be dimensionless, i.e., they are represented by points, then multiple robots can share the same location. The *multiplicity detection* capability allows the robots to identify such a multiplicity point. The following are the different types of *multiplicity detection* capability:

1. **Local Weak:** A robot determines whether or not its current location is a multiplicity point. However, it cannot exactly count the total number of robots in its current position. In addition, it cannot recognize other multiplicity points aside from its current location.

2. **Global Weak:** The robots can identify any multiplicity point in the domain. But, it is unable to calculate the total number of robots present at a multiplicity location.

3. **Local Strong:** A robot can count the exact number of robots that are present at its location. However, a robot is unable to count this for other multiplicity points.

4. **Global Strong:** The robots know the exact number of robots that are present at any multiplicity point.

1.2.9 Memory and Communication

In general, the robots are assumed to be *oblivious* and *silent*. However, there are some variants of the model where persistent memory and communication capabilities are provided by the presence of lights [47]. These lights can assume a finite number of pre-defined colors. Each color indicates a different state of the robot. The following kinds of light models are being considered:

1. **$\mathcal{F}$-STATE:** The robots can remember their state from their previous cycle but do not have knowledge of the states of other robots. In this model, the robots are *non-oblivious* as they have a persistent memory. However, they are *silent*.

2. **$\mathcal{F}$-COMM:** The robots cannot remember their own states but can identify the states of other robots. The robots are not *silent* as they can explicitly communicate using lights. However, they are *oblivious*.

3. **$\mathcal{F}$-ALL:** The robots can remember their state set in their previous cycle as well as identify the states of other robots. In this model, the robots are neither *oblivious* nor *silent*.

1.3 Geometric Problems

The primary focus of the research has been on issues related to solving fundamental geometric problems. Some of the well-known geoemtric problems are discussed below:

1. **Gathering:** The *gathering* problem [10–22] asks all the robots to meet at a single point not known a priori within finite time. The *convergence* problem and *near gathering* problem are very closely related to the *gathering* problem. To solve the convergence problem, the robots need to be as close as possible. A solution to the *near gathering* problem requires that all the robots reach and remain inside a disk of a pre-fixed radius. A variant of the *gathering* problem known as the *gathering on meeting* points has been studied in which the robots need to gather at one of the pre-fixed *meeting* points.

2. **Pattern Formation:** A solution to the *pattern formation* problem [29–31, 35–37] asks the robots to position themselves so that they form a given pattern within a finite time. The *initial* requirement is that no two robots share the same location, and the number of points prescribed in the pattern is exactly equal to the number of robots. To solve the *circle formation* problem, the robots must position themselves on a circle whose center is not fixed a priori at distinct locations [28, 34, 49–52]. The task must be completed within a finite time. To solve the *line formation* problem [53, 54] the robots need to reach and remain in a straight line. The *plane formation* problem [55–57] asks a swarm of robots moving in three-dimensional Euclidean space to land on a common plane that is not defined a priori.

3. **Mutual Visibility:** The *mutual visibility* problem [46, 58–68] is considered under *obstructed* visibility model. A swarm of robots must arrange themselves in distinct positions such that no three robots are collinear.

4. **Scattering:** To solve the *scattering* problem [38–42], the robots need to re-position themselves so that no two robots share the same location.

5. **Flocking:** *Flocking* [23–27] is relatively a more complex task compared to the above discussed geometric problem. The *flocking* problem requires the formation of

a pattern as well as maintaining the pattern while moving together as one flock. A solution to the *flocking* problem demands more coordination among the robots.

1.4 Thesis Contributions

In this thesis, we study the *k-circle formation* problem in the Euclidean plane. The *k-circle formation* problem considers $m > 0$ pre-fixed points (called as *fixed points*) and n number of mobile robots in the Euclidean plane. The *fixed points* are visible to the robots like landmarks. The *k-circle formation* problem is a hybrid problem that connects the *partitioning*, *circle formation* and *embedded pattern formation* problems. A generalized version of the *embedded pattern formation* problem is the k-EPF problem, which requires the robot to reach and remain in a final configuration in which each fixed point contains exactly k robots.

Problem Definition: For some positive integer k, the *k-circle formation* problem asks a group of n autonomous mobile robots to form m disjoint circles. Each such circle is restricted to being centered at one of the *fixed points* given in the plane. Each circle must have k robots in distinct positions. The circles need not be uniform. In general, the circles can have different radii. However, the circles are assumed to have equal radii in this thesis, which is a special case for the *k-circle formation* problem.

The feasibility of the problem is investigated under different sets of assumptions. For a particular set of assumptions, all the deterministically unsolvable cases are characterized. For the rest of the cases, a deterministic distributed algorithm that solves the problem within a finite time is proposed. The correctness of all the proposed deterministic algorithms is discussed. In this thesis, the *k-circle formation* problem is investigated for the set of assumptions presented in the Table 1.1. All the problems being addressed in this thesis are considered under the ASYNC scheduler with non-rigid motion. Also, there is no assumption of a common *chirality* in all the results obtained.

Agreement	Visibility	Knowledge of Fixed points	Dimension	Chapter
One-Axis	Unlimited	Complete	Point	Chapter 3
No-Axis	Unlimited	Complete	Point	Chapter 4
No-Axis	Obstructed	Complete	Point	Chapter 5
No-Axis	Obstructed	Zero	Point	Chapter 5
One-Axis	Unlimited	Complete	Fat	Chapter 6

Table 1.1: Thesis Contributions

1.4.1 k-Circle Formation and k-EPF

In Chapter 3, we consider that the robots have an agreement on the direction and orientation of one of the axes. The robots are assumed to be dimensionless. They have *unlimited* visibility, and they are *silent* and *oblivious*. The contributions of this work are as follows:

Result 1: All the *initial* configurations and values of k for which the problem is deterministically unsolvable are characterized when $n = km$.

Result 2: A deterministic distributed algorithm is proposed that solves the problem within finite time when $n = km$.

Result 3: All the *initial* configurations and values of k for which the problem is deterministically unsolvable are characterized when $n > km$. In this case, there will be $n - km$ surplus robots that will not be assigned to any circle.

Result 4: All the *initial* configurations and values of k for which the problem is deterministically unsolvable are characterized when $n < km$. In this case, the objective is to maximize the number of circles containing exactly k robots.

Result 5: It is shown that if the k-circle formation problem is deterministically solvable than the k-EPF problem is also deterministically solvable.

1.4.2 k-Circle Formation by Disoriented Robots

In Chapter 4, the robots are assumed to be completely *disoriented*, i.e., they neither have any agreement on a global coordinate system nor have any agreement on a common *chirality*. The robots are assumed to be dimensionless. They have *unlimited* visibility, and they are *silent* and *oblivious*. The contributions of this work are as follows:

Result 1: All the *initial* configurations and values of k for which the problem is deterministically unsolvable in this setting are characterized.

Result 2: A deterministic distributed algorithm is proposed that solves the problem within finite time.

1.4.3 k-Circle Formation by Opaque Robots

In Chapter 5, we investigate the *k-circle formation* problem under *obstructed visibility* model. The robots are assumed to be *opaque*, i.e., a robot can not see another robot if a third robot is positioned on the line segment joining them. They are assumed to be dimensionless and completely *disoriented*. Based on the visibility of the *fixed* points, the following two different settings are considered:

1. Complete knowledge of fixed points. A robot cannot obstruct the visibility of a fixed point for other robots. The positions of all the fixed points are known to the robots. As a consequence, the robots have the knowledge of the total number of fixed points. The robots are *silent* and *oblivious*.

2. Zero knowledge of fixed points. A robot can obstruct the visibility of a fixed point for other robots. If a robot lies on the line segment joining a fixed point and another robot, then the other robot can not see the fixed point. The robots do not have the knowledge of the total number of fixed points. They are assumed to be equipped with lights which provides persistent memory and communication capabilities.

The contributions of this chapter are as follows:

Result 1: All the *initial* configurations and values of k for which the *k-circle formation* problem is deterministically unsolvable are characterized with the complete knowledge of the fixed points.

Result 2: A deterministic distributed algorithm is proposed that solves the *k-circle formation* problem within finite time with complete knowledge of fixed points.

Result 3: It is shown that the problem is deterministically *unsolvable* by *silent* and *oblivious* robots with the zero knowledge of the fixed points.

Result 4: A deterministic distributed algorithm is proposed considering one bit of persistent memory that solves the *k-circle formation* problem within finite time with zero knowledge of fixed points.

1.4.4 Uniform k-Circle Formation by Fat Robots

In Chapter 6, the *uniform k-circle formation* problem is investigated for a set of fat robots in the plane. To solve the *uniform k-circle formation* problem in addition to solving the *k-circle formation* problem, all the k robots on a circle must form a regular k-gon. The robots are represented by *transparent* unit disks. They are assumed to have an agreement on the direction and orientation of one of the axes. The robots are *silent* and *oblivious*. The following results are shown:

Result 1: All the *initial* configurations and values of k for which the *uniform k-circle formation* problem is deterministically unsolvable are characterized for fat robots.

Result 2: A deterministic distributed algorithm is proposed that solves the *uniform k-circle formation* problem within finite time.

1.5 Outline of the Thesis

Chapter 2 presents the literature survey of the existing works relevant to this thesis. The main contributions of this thesis are presented in Chapter 3 to Chapter 6. A summary of the results is shown in the following table (Table 1.1). Chapter 7 concludes the thesis by summarizing the research works done in this thesis and discussing the future directions of researches that come out of these studies.

Chapter 2

Related Works

Contents

2.1 Overview

A large volume of results have been reported in the literature which focus on the feasibility of a geometric problem under different sets of assumptions, namely agreement on the global axes, *chirality*, scheduler, dimension, visibility, etc. Finding minimal sets of capabilities to solve a given problem is the primary objective. The goal is to identify the minimal essential capabilities required for robots to perform a task, thereby reducing the cost of mass production. The most researched problem is the *gathering* problem, which can also be referred to as a point formation problem. The *pattern formation* problem has also been extensively studied in the literature. In this thesis, we investigate the *k-circle formation* problem, which is a special kind of *pattern formation* problem. In this chapter, we present a brief literature survey focusing on some fundamental geometric problems related to the *k-circle formation* problem.

2.2 Partitioning Problem

The *partitioning* problem asks the robots to partition themselves into multiple groups, with specified number of robots in each group. The robots in each group also need to converge in a small area. Efrima and Peleg [69] studied the *partitioning* problem in the Euclidean plane. They presented crash-fault-tolerant *partitioning* algorithms for various levels of common orientation and different timing models. Liu et al. [44] investigated the *team assembling* problem for heterogeneous robots. The robots need to form multiple teams, each containing a pre-fixed number of robots of different kinds. The *k-circle formation* problem can be viewed as a variant of the *partitioning* problem [69] and the *team assembling* problem [44].

2.3 Gathering in the Continuous Domain

Gathering is a fundamental coordination problem for a swarm of mobile robots. To solve the *gathering* problem, the robots need to gather at a point which is not fixed a priori. The *gathering* problem has been extensively studied both in the continuous domain [8, 12–15, 17–19, 21, 22, 48, 70–81] and discrete domain [16, 20, 82–91].

2.3.1 Gathering for Two Robots

Suzuki and Yamashita [8] proved that under SSYNC scheduler, the *gathering* problem for two robots, also known as the *rendezvous* problem, is deterministically unsolvable. Izumi et al. [19] investigated the magnitude of consistency between the local coordinate systems, which is necessary and sufficient to solve the *gathering* problem for two oblivious robots under SSYNC and ASYNC models. They considered two families of unreliable compasses: the static compass with (possibly incorrect) constant bearings and the dynamic compass, whose bearings can change arbitrarily (immediately before a new look-compute-move cycle starts and after the last cycle ends). The deviation (ϕ) is measured by the largest angle formed between the x-axis of a compass and the reference direction of the global coordinate system. For each of the combinations of robot and compass models, the condition on deviation ϕ that allows an algorithm to solve the *gathering* problem is established:

1. for SSYNC and ASYNC robots with static compasses $\phi < \frac{\pi}{2}$,
2. for SSYNC robots with dynamic compasses $\phi < \frac{\pi}{4}$, and
3. for ASYNC robots with dynamic compasses $\phi < \frac{\pi}{6}$.

In the first two cases, the above mentioned sufficient conditions are also necessary.

Flocchini et al. [79] proved that with rigid motions, *rendezvous* is solvable by $\mathcal{F}$-STATE robots under SSYNC scheduler and by $\mathcal{F}$-COMM robots even under ASYNC scheduler. Okumara et al. [80] showed that under ASYNC scheduler, *rendezvous* can be solved with two light colors in non-rigid movement if robots know the value of the minimum distance δ. Viglietta [22] gave a complete characterization of the number of light colors that are necessary to solve the *rendezvous* problem in different models, ranging from FSYNC to SSYNC to ASYNC, rigid and non-rigid, with preset or arbitrary *initial* configuration. Bramas et al. [48] showed that if the robots disagree on the unit distance of their coordinate system, it becomes possible to solve *rendezvous* and agree on a final common location without additional assumptions.

2.3.2 Gathering for more than Two Robots

Non-faulty Robots: Cohen and Peleg [14] proved the correctness of the gravitational algorithm for the *convergence* problem in the fully ASYNC model. Cohen and Peleg [77] studied the *convergence* problem, focusing on the ability of robot systems with inaccurate sensors, movements, and calculations to carry out the task of convergence. Cieliebak et al. [13] proved that the *gathering* problem for $n > 2$ robots under ASYNC scheduler is solvable for disoriented and oblivious robots starting from arbitrary *initial* configuration.

Prencipe [21] proved that in both the ASYNC and SSYNC settings, there does not exist any deterministic oblivious algorithm that solves the *gathering* problem within a finite time for $n \geq 2$ disoriented robots if they does not have multiplicity detection capability.

Flochhini et al. [78] studied the *gathering* problem for robots with limited visibility under ASYNC scheduler when the robots have full-axis agreement. Di Luna et al. [92] studied the *gathering* problem on a circle, in which all robots with limited visibility are initially in distinct locations on the circle, and their goal is to reach the same point on the circle within a finite time. Poudel et al. [81] proposed an $O(D_E)$ time algorithm for the *gathering* problem with limited visibility under ASYNC scheduler with the assumption of one axis agreement, where D_E is the Euclidean distance between the farthest-pair of robots in the *initial* configuration.

Czyzowicz et al. [15] were the first to study the *gathering* problem for fat robots. The authors solved the *gathering* problem for at most four robots. Honorat et al. [18] considered the *gathering* problem for four fat robots equipped with slim omnidirectional cameras and provided an algorithm to solve the problem in a fully ASYNC setting. For $n \geq 5$ transparent fat robots, Gan Chaudhuri et al. [11] studied the *gathering* problem. Agathangelou et al. [70] considered the *gathering* problem for opaque fat robots under ASYNC scheduler. The proposed algorithm works for any number of robots, starting from any *initial* configuration, with the assumption of a common *chirality*. A distributed algorithm is presented to solve the *gathering* problem in the three dimensional Euclidean space for a set of ASYNC robots under *obstructed* visibility by Bhagat et al. [93].

Bhagat et al. [74] studied the *gathering* problem by minimizing the maximum distance traveled by a single robot. They proved that a set of oblivious robots cannot solve the constrained *gathering* problem under FSYNC scheduler, even with multiplicity detection capability. They proposed a distributed algorithm for the constrained *gathering* problem for $n \geq 5$ robots using two bits of persistent memory. The min-max *gathering* of oblivious robots under ASYNC scheduler with non-rigid motion was considered by Bhagat et al. [72]. Cicerone et al. [12] investigated a variant of the *gathering* problem, considering *meeting* points in the plane. To solve the *gathering on meeting* points problem, the robots are required to gather at one of the pre-fixed *meeting* points. They fully characterized when the *gathering on meeting* points can be accomplished. They also studied when *gathering on meeting* points can be accomplished with respect to two objective functions: minimizing the total traveled distance by all robots and minimizing the maximum traveled distance performed by a single robot. Bhagat et al. [73] considered the *gathering* problem in the presence of obstacles. They proposed a distributed algorithm for *gathering* which works even if the configuration contains multiplicity points in the presence of non-intersecting transparent convex polygonal obstacles.

Faulty Robots: Cohen and Peleg [14] investigated the *convergence* problem in the presence of crash-fault robots. Agmon and Peleg [71] considered the *gathering* problem in the presence of both crash-fault and byzantine-fault. They observed that most of the existing algorithms would fail to operate correctly in a crash-fault setting. They proposed a single crash-fault-tolerant algorithm for $n \geq 3$. They showed that under SSYNC scheduler, the *gathering* for $n = 3$ robots is impossible even if at most one byzantine-fault robot is present. Next, a byzantine-fault-tolerant algorithm was proposed under FSYNC scheduler that solves the *gathering* problem in an n robot system with up to f faults, where $n \geq 3f + 1$. Bouzid et al. [75] studied the *gathering* for n robots with f crash-fault robots for any $f < n$. They provided a wait-free algorithm to gather all the non-faulty robots, assuming strong multiplicity detection and *chirality*. Défago et al. [17] investigated the feasibility of the *gathering* problem in a deterministic manner in terms of different synchrony modes and presence of faults (crash or byzantine). A deterministic *gathering* algorithm that admits an arbitrary number of crashes and gathers all the correct robots even if they do not have a common *chirality* was presented by Bramas et al. [76].

Bhagat et al. [94] investigated the *gathering* problem for $n \geq 2$ robots in the presence of f crash-fault robots under one axis agreement. They proposed two deterministic algorithms which solve the *gathering* problem starting from any *initial* configuration, one for *unlimited* visibility and another for *obstructed* visibility. Bhagat et al. [10] also addressed the *gathering* problem under SSYNC scheduler in the presence of crash-fault robots. First, a distributed algorithm is proposed which can tolerate at most ($\lfloor n/2 - 1 \rfloor$) crash faults for $n \geq 7$ robots with weak multiplicity detection. Next, a distributed algorithm was presented with knowledge of δ, which can tolerate at most $(n - 6)$ crash faults for $n \geq 7$ robots. In 3D space, the *gathering* problem under crash-fault model for a set of SSYNC *opaque* robots was studied by Bhagat et al. [93].

2.4 Gathering in the Discrete Domain

For an odd number of robots, Klasing et al. [20] proved that the *gathering* is feasible if and only if the *initial* configuration is not periodic and provided a *gathering* algorithm for any such configurations. For an even number of robots, they established the feasibility of *gathering* except for one type of symmetric configurations, and proposed *gathering* algorithms for *initial* configurations that proved to be gatherable. Klasing et al. [91] studied the influence of symmetries of the configuration on the feasibility of *gathering* on a ring under ASYNC scheduler. Izumi et al. [89] proposed a deterministic algorithm for the *gathering* problem on rings assuming weak multiplicity. The proposed algorithm is time optimal, i.e., the time complexity is $O(n)$, where n is the number of nodes. Kamei et al. [90] proposed a *gathering* protocol for an even number of robots in a ring that allows symmetric but not periodic configurations as *initial* configurations, using only local weak multiplicity detection. In their proposed protocol, the number of robots $k \geq 8$ and the number of nodes n on a network must be odd and greater than $k+3$. D'Angelo et al. [86] studied the *gathering* of six oblivious robots on anonymous symmetric rings. Bonnet et al. [84] investigated the *gathering* on a ring for four ASYNC robots. Das et al. [87] considered *gathering* on a ring in the presence of an adversarial mobile entity called the malicious agent. The *gathering* problem has also been studied in dynamic rings [85, 88].

D'Angelo et al. [16] studied the *gathering* problem in grid and tree networks. They provided a full characterization about gatherable configurations for grids and trees. They showed that on these topologies, the multiplicity detection is not required. Di Stefano [95] proposed an optimal algorithm in terms of the total number of moves for the *gathering* problem in infinite grids. They fully characterized the cases when optimal *gathering* is achievable by providing a distributed algorithm. Bhagat et al. [82, 83] considered the *gathering on meeting* nodes problem in an infinite grid.

2.5 Arbitrary Pattern Formation

The *arbitrary pattern formation* (*APF*) problem asks the robots to form an arbitrary pattern P which is given as an input.

Deterministic Algorithms: Suzuki and Yamashita [8, 96] were the first to study the APF problem in the Euclidean plane. They completely characterized the class of formable patterns under FSYNC and SSYNC schedulers for autonomous as well as anonymous robots when they have an unbounded amount of memory. The symmetricity $\rho(C)$ of a configuration C is the order of the rotational symmetry of the configuration. The characterizations are based on the symmetricity of a configuration. They showed that under SSYNC scheduler, the *gathering* problem for two oblivious robots is deterministically unsolvable, while it is trivially solvable for non-oblivious robots. The families of patterns formable by oblivious robots were characterized by Yamashita and Suzuki [37] under FSYNC and SSYNC schedulers. The results from the papers [8, 37, 96] can be summarized as follows:

1. a pattern P is formable from an *initial* configuration I by non-oblivious FSYNC robots if and only if $\rho(I)$ divides $\rho(P)$;
2. P is formable from I by oblivious FSYNC robots if and only if $\rho(I)$ divides $\rho(P)$;
3. P is formable from I by oblivious SSYNC robots if and only if P is not a point with two robots and $\rho(I)$ divides $\rho(P)$.

Fujinaga et al. [36] proved that for an *initial* configuration I without any multiplicity point, pattern P is formable from I by oblivious ASYNC robots if and only if P is not a point of multiplicity 2 and $\rho(I)$ divides $\rho(P)$. Flochhini et al. [35] showed that the patterns that can be formed depend heavily on the level of a priori agreement, the robots have about the orientation and direction of the axes in their local coordinate system. They showed the following:

1. If the robots are *disoriented*, then the robots cannot form an arbitrary pattern.
2. If the robots have one axis agreement, then any odd number of robots can form any arbitrary pattern. However, an even number of robots cannot form certain patterns in the worst case.
3. If the robots have full axis agreement, then any pattern can be formed by any number of robots.

They also proved that if it is possible to solve the *pattern formation* problem for $n \geq 3$ robots, then the *leader election* problem is also solvable. The relationship between the APF and *leader election* problem was studied by Dieudonné et al. [97] under ASYNC scheduler. They have proposed an algorithm that solves the APF problem starting from an *initial* configuration in which *leader election* is possible. They proved that for $n \geq 4$, the APF problem and *leader election* problem are equivalent if the robots have a common *chirality*. Bramas and Tixeuil [98] presented an algorithm that deterministically solves the APF problem for $n = 4$ robots under ASYNC scheduler. Cicerone et al. [30] investigated the APF problem without any assumption of a common *chirality*. They proved that for a given *initial* configuration I with any number of robots, the APF problem is solvable if and only if the *leader election* is solvable. In infinite grid, Bose et al. [99] studied the APF problem under a fully ASYNC scheduler. The APF problem was considered in the regular tesselation graphs (triangular and hexagonal grids) by Cicerone et al. [100]. The formation of a series of geometric patterns instead of a single pattern was investigated by Das et al. [101].

Yamauchi et al. [57] first considered *pattern formation* in three dimensional space. They presented a necessary and sufficient condition for FSYNC robots to solve the *plane formation* problem that does not depend on obliviousness. They assumed that the robots have a common *chirality*. Yusaku et al. [55] investigated the *plane formation* problem without the assumption of a common *chirality* for FSYNC robots. Uehara et al. [56] considered the *plane formation* problem for SSYNC robots with non-rigid movement.

Yamauchi et al. [102] were the first to study the APF problem under limited visibility. They showed that even if $\rho(I)$ divides $\rho(P)$, FSYNC oblivious robots with *limited* visibility may not be able to form any arbitrary pattern P. Next, they considered non-oblivious robots, each of which can record the history of local views and outputs during execution. They showed that SSYNC robots with rigid moves, and FSYNC robots with non-rigid moves have the same formation power as robots with unlimited visibility. Bose et al. [45] provided a full characterization of the *initial* configurations for which the APF problem is solvable by opaque robots in the settings where (a) robots have full axis agreement and (b) robots have one axis agreement. Bose et al. [103] also investigated the APF problem for fat robots under obstructed visibility. In this setting, the authors completely characterized all the *initial* configurations from which any arbitrary pattern can be formed in a deterministic distributed manner. In an infinite grid, Lukovszki et al. [104] studied the *pattern formation* problem under limited visibility.

Randomized Algorithms: All the works discussed above limit themselves to the solvability of the *APF* problem in a deterministic manner. Yamauchi et al. [105] proposed a randomized algorithm for the *APF* problem. They assumed that the robots have a common *chirality*. The proposed algorithm [105] consists of two phases. In the first phase, given an *initial* configuration I, if the symmetricity $\rho(I) > 1$, then the proposed algorithm translates I into another configuration I' such that $\rho(I') = 1$ with probability 1. In the second phase, a deterministic algorithm (e.g., [97]) can be used to form any pattern P starting from I' as $\rho(I') = 1$. Bramas and Tixeuil [29] proposed a new probabilistic algorithm to solve the *APF* problem without the assumption of a common *chirality*. The proposed algorithm consists of two phases: a probabilistic leader election phase, and a deterministic *pattern formation* phase. Also, the arbitrary pattern P can contain multiplicity points (except in the case of *gathering*, which is a special pattern defined by a unique point of multiplicity that remains impossible to solve [21]).

Vaidyanathan et al. [106] proposed randomized algorithms considering both oblivious and light models for the robots. They have proved runtime bounds for solving the *APF* problem in terms of the time required to solve the *leader election* problem. Hector et al. [107] presented two randomized algorithms for the *APF* problem under ASYNC scheduler, one under the classical oblivious model and another under the light model. Both the proposed algorithms run in $O(\max\{D^i, D^p\})$ time with $O(\max\{D^i, D^p\})$ moves by each robot, where D^i and D^p, respectively, are the diameters of the *initial* and pattern configurations. The algorithm for the light model uses $O(1)$ colors. They also proved a lower bound of $\Omega(\max\{D^i, D^p\})$ for time for any *APF* algorithm if scaling is not allowed on the target pattern.

2.5.1 Circle Formation

The *circle formation* problem asks the robots to position themselveson the circumference of a circle within a finite time; the center of the circle is not known a priori. Défago et al. [108] investigated the *circle formation* problem in a setting where the robots have no common origin, unit distance, or sense of direction. They proposed a distributed algorithm by which the robots would eventually form a circle. A new approach for the *circle formation* problem based on concentric circles formed by the robots was presented by Dieudonné et al. [32]. A distributed algorithm was proposed by Défago et al. [50], which ensured that the robots would deterministically form a non-uniform circle within a finite number of steps and would converge towards a solution to the *uniform circle formation*. Flocchini et al. [34] studied the *uniform circle formation* problem. They proved that the problem is solvable for any *initial* configuration with distinct robot positions. An optimum distributed algorithm that minimizes the maximum distance traveled by any robot to solve the *circle formation* problem was proposed by Bhagat et al. [28]. Datta et al. [49] proposed a distributed algorithm for the *circle formation* by a system of transparent fat robots. For fat robots with limited visibility, the *circle formation* problem was studied by Dutta et al. [51]. The *uniform circle formation* problem was considered for fat robots with limited visibility by Mondal et al. [52]. Felleti et al. [33] studied the *uniform circle formation* for opaque robots with lights.

2.5.2 Embedded Pattern Formation

Given a set of pre-fixed pattern points, the *embedded pattern formation* problem [31, 109] asks the robots to reach a *final* configuration in which each pattern point contains exactly one robot position. The pre-fixed points are assumed to be visible to all the robots, like landmarks. Fujinaga et al. [109] investigated the *embedded pattern formation* problem in a setting where the robots have a common *chirality*. They have shown that the *embedded pattern formation* problem is solvable by oblivious robots through the optimum matching between the robots and the pattern points under ASYNC scheduler. Later, Cicerone et al. [31] have studied the *embedded pattern formation* problem in a setting where the robots do not have a common *chirality*. They have fully characterized all the *initial* configurations for which the *embedded pattern formation* is unsolvable.

2.6 Mutual Visibility

A fundamental problem under *obstructed visibility* model is the *mutual visibility* problem: starting from an *initial* configuration, the robots must reach a configuration within finite time and without collision in which they can all see each other (i.e., no three robots are collinear). The *mutual visibility* problem is important as it gives a basis for any subsequent task requiring complete visibility.

Continuous Domain: Di Luna et al. [61] presented the first algorithm for the *mutual visibility* problem for oblivious robots under SSYNC scheduler. The proposed algorithm assumes that the robots have knowledge of the total number of robots and solves the *mutual visibility* problem by forming a convex n-gon. Without the knowledge of n, Di Luna et al. [62] proposed a deterministic algorithm that solves *mutual visibility* with six colors in the SSYNC setting and with ten colors in the ASYNC setting. For rigid motions, Di Luna et al. [46] proved the following:

1. if the robots have knowledge of n, then *mutual visibility* is solvable with no colors under SSYNC scheduler;
2. the *mutual visibility* is always solvable with two colors under SSYNC scheduler;
3. the *mutual visibility* is always solvable with three colors under ASYNC scheduler.

In case of non-rigid movements, Di Luna et al. [46] proved the following:

1. if the robots know δ and n, then the *mutual visibility* is solvable with no colors under SSYNC scheduler;
2. if the robots know δ, the *mutual visibility* is solvable with two colors under SSYNC scheduler;
3. the *mutual visibility* is always solvable with three colors under SSYNC scheduler;
4. if the robots agree on the direction of one coordinate axis, then the *mutual visibility* is solvable with three colors under ASYNC scheduler.

Sharma et al. [65] presented an improved algorithm which requires only two colors and works for both SSYNC and ASYNC schedulers under both rigid and non-rigid moves. The proposed algorithm is optimal in terms of persistent memory since any algorithm for *mutual visibility* requires at least two colors when n is not known. Bhagat et al. [110] solved the *mutual visibility* problem by assuming one bit of persistent memory and the knowledge of n under ASYNC scheduler. Without the knowledge of n, Bhagat et al. [111] investigated the *mutual visibility* problem under SSYNC scheduler using only one bit of persistent memory.

Sharma et al. [64] studied the runtime bounds for the proposed algorithms by Di Luna et al. [46] under FSYNC scheduler. They also proposed a new deterministic algorithm that solves the *mutual visibility* problem in $O(n \log n)$ rounds under FSYNC scheduler. They studied the runtime bounds of these algorithms under FSYNC scheduler. Vaidyanathan et al. [68] presented a sublinear time algorithm for *complete visibility* under FSYNC scheduler. The proposed algorithm runs in $O(\log n)$ time using twelve light colors. Sharma et al. [112] presented the first algorithm for *complete visibility* with $O(1)$ runtime under SSYNC scheduler. Later, Sharma et al. [66, 67] proposed algorithms with runtimes $O(\log n)$ and $O(1)$ using 25 and 47 light colors, respectively. Bhagat [113] presented a deterministic distributed algorithm to solve the *mutual visibility* problem for a set of synchronous robots using only one bit of persistent memory. The proposed algorithm solves the *mutual visibility* problem in two rounds and ensures collision-free movements for the robots. Sharma et al. [114] studied the *complete visibility* problem for fat robots. They proposed an algorithm for unit disc robots that solves *complete visibility* in $O(n)$ time using nine colors under FSYNC scheduler.

Bhagat et al. [60] proposed an optimum algorithm to solve the *mutual visibility* problem under ASYNC scheduler. The proposed solution minimizes the maximum distance travelled by a single robot using seven light colors. Aljohani et al. [59] proposed an algorithm that solves *complete visibility* tolerating one crash fault robot for $n \geq 3$ robots. They also presented an impossibility result for solving complete visibility if there is a byzantine fault single robot for $n = 3$ robots. Poudel et al. [63] provided the first algorithm for complete visibility that tolerates $f \leq n$ crash-fault robots in the ASYNC setting under one-axis agreement.

Discrete Domain: Adhikary et al. [58] first studied the *mutual visibility* problem in an infinite grid. They provided an algorithm that solves the problem starting from any *initial* configuration using nine colors under ASYNC scheduler. Poudel et al. [115] studied the mutual visibility problem for fat robots in an infinite grid. In this study, the robots were not restricted to move along grid lines or to move by one hop, i.e., a robot can directly move to any visible grid point in one step. They proposed a deterministic algorithm for $n \geq 4$ robots, positioned on the distinct nodes in $\sqrt{n} \times \sqrt{n}$ sub-grid under SSYNC scheduler, that solves the mutual visibility in $O(\sqrt{n})$ time.

Sharma et al. [116] primarily focused on minimizing (or providing a trade-off between) two fundamental performance metrics: (i) time to solve complete visibility and (ii) area occupied by the solution. They proved that mutual visibility can be optimally solved in $O(\max\{D, n\})$ time (where D is the diameter of the *initial* configuration), and with a final optimal area of $O(n^2)$. The proposed algorithm solves the *mutual visibility* problem under ASYNC scheduler through: (i) a deterministic algorithm using 17 colors if leader election is not required; (ii) a randomized algorithm using 32 colors that terminates in $O(\max\{D, n\})$ time with probability at least $1 - \frac{1}{2^{\max(D,n)}}$, if leader election is required. Hector et al. [117] studied the *convex hull formation* problem where all the robots are placed on the convex hull (solving the *mutual visibility* problem). They presented two randomized algorithms: an $O(\max\{n^2, D\})$ time algorithm using 50 colors that creates an $O(n^2)$ perimeter convex hull and an $O(\max\{n^{\frac{3}{2}}, D\})$ time algorithm using 55 colors that creates an $O(n^{\frac{3}{2}})$ perimeter convex hull.

Chapter 3

k-Circle Formation and k-EPF Problem

Contents

3.1 Overview of the Problem

In this chapter, we investigate the solvability of the k-*circle formation* problem under one axis agreement. Also, the relationship of the k-*circle formation* problem with the k-*EPF* problem (a generalized version of the *embedded pattern formation* problem) is studied. The theoretical motivation for studying the k-*circle formation* problem is twofold. First, we believe that the problem is theoretically interesting as it is a hybrid problem in between the *partitioning* problem [44, 69] and the *circle formation* problem [28, 34, 49–52]. Both the problems individually differs from the k-circle formation problem w.r.t. the following points:

1. The *partitioning* problem asks the robots to divide themselves into m groups, each having k robots. In addition, the robots in each group are asked to converge in a small area. Unlike the k-*circle formation* problem, the robots do not need to form circles containing exactly k robots, centered at one of the pre-fixed points.

2. The *circle formation* problem asks the robots to place themselves at distinct locations on a circle (not defined a priori), within finite amount of time. In this problem, all the robots participate in forming one single circle, whereas, in the k-*circle formation* problem, the robots need to form m circles each containing exactly k robots and centered at one of the fixed points.

To the best of our knowledge, we believe that this is the first work that aims at connecting the two well-known problems in the literature, namely the *partitioning* problem and the *circle formation* problem. Both the *partitioning* and *circle formation* problems do not consider the fixed points as well as symmetries related to the fixed points whereas the k-*circle formation* problem must address the symmetries related to the fixed points.

Secondly, if the robots could solve the k-circle formation problem, then all the k robots which lie on the same circle can gather at their respective center, which is a fixed point, within finite number of moves. Thus, studying the solvability of the k-*circle formation* problem includes investigating the solvability of the k-EPF problem where k robots need to reach and remain at each fixed point.

In addition, we believe that the k-*circle formation* problem would have the following applications in the field of swarm robotics:

1. The set of fixed points can be considered as emergency points, which need to be surrounded. By solving the k-*circle formation* problem, a swarm of robots can divide themselves into groups, containing k robots each and build a perimeter, surrounding the emergency points.

2. The set of fixed points can also be considered as charging stations, with some given permitted capacity. The robots need to be charged after a certain amount of time to continue working. By solving the *k-circle formation* problem, the robots can reach the charging stations without violating the permitted capacity.

3.2 Model and Definitions

The robots are assumed to be dimensionless, oblivious, anonymous, autonomous, and homogeneous. They are represented by points in the Euclidean plane. They have unlimited visibility range and have no explicit way of communication. The movements of robots are non-rigid. They execute Look-Compute-Move (LCM) cycle when they become active. We have considered a fair ASYNC scheduler. We assume that they have an agreement on the y-axis. The following notations are used in the proposed algorithms.

- **Configuration:** Let $R = \{r_1, r_2, \ldots, r_n\}$ be the set of robots. Let $r_i(t)$ denote the position of the robot r_i at time t. $R(t) = \{r_1(t), r_2(t), ..., r_n(t)\}$ is the set of robot positions at time t. We are given a set of fixed points denoted by $F = \{f_1, f_2, \ldots, f_m\}$. It is assumed that $n = km$ for some positive integer k. Let F_c be the center of gravity of the set of fixed points F. We assume that the y-axis passes through F_c and F_c is the origin. Let F_y and $R_y(t)$ denote the set of fixed points and robot positions, respectively, on the y-axis at time t. Suppose $\mathcal{H}_1$ and $\mathcal{H}_2$ denote the two half-planes delimited by the y-axis. Let $d(r, f)$ denote the Euclidean distance between r and f. The pair $C(t) = (R(t), F)$ represents the configuration at time t. In an *initial* configuration $C(0)$, it is assumed that all the robots are stationary and are placed at distinct positions. A configuration is said to be *balanced* at time t if the number of robots in both the open half-planes delimited by the y-axis is equal. Otherwise, the configuration is said to be *unbalanced*.

- **Circles and radii of circles:** We consider that all the circles formed by the robots would have the same radius. Let ρ denote the radius of the circles. Also, let $C(f, \rho)$ denote the circle centered at $f \in F$ with radius ρ. We have used the following notations to formulate the radius ρ of the circles:

1. ρ_1 = minimum distance between two fixed points.
2. ρ_2 = minimum distance between a fixed point $f \in (F \setminus F_y)$ and the y-axis.

The radius ρ is defined as $\rho = \frac{1}{3} \min(\rho_1, \rho_2)$.

- A fixed point and its respective circle $C(f_j, \rho)$ are said to be *unsaturated*, if $C(f_j, \rho)$ contains less than k robots on it. Let $D_j(t)$ denote the deficit in the number of robots in order to have exactly k robots on the $C(f_j, \rho)$. A fixed point and its respective circle $C(f_j, \rho)$ are said to be *saturated*, if $C(f_j, \rho)$ contains exactly k robots on it. In case $C(f_j, \rho)$ contains more than k robots, then $C(f_j, \rho)$ and f_j are called *oversaturated*.

- **Configuration Rank.** Let $y(s_i)$ denote the y-coordinate of a point s_i. Note that the robots do not have an agreement on the positive direction of the x-axis. In case, the robots could have an agreement on the positive direction of the x-axis, $\beta(s_i)$ denotes the x-coordinate of s_i. Otherwise, $\beta(s_i)$ denotes the distance of s_i from the y-axis. The pair $\gamma(s_i) = (\beta(s_i), y(s_i))$ is the configuration rank of the point s_i. Between the two points s_i and s_j, s_i is said to have higher configuration rank than s_j, if $y(s_i) > y(s_j)$ or $y(s_i) = y(s_j)$ and $\beta(s_i) > \beta(s_j)$. Since the robots have *unlimited* visibility, they can compute the configuration rank of each point $s_i \in F \cup R(t)$.

- **Symmetry about the y-axis.** If the robots r_i and r_j for $i \neq j$, have the same configuration rank, i.e., $\gamma(r_i(t)) = \gamma(r_j(t))$, they are said to be symmetric about the y-axis. Let $\phi(r)$ denote the symmetric image of r about the y-axis. If robots r_i and r_j are symmetric about the y-axis, then $r_i = \phi(r_j)$ and $r_j = \phi(r_i)$. Similarly, two fixed points are said to be symmetric about the y-axis, if they have the same configuration rank. An active robot in its *look* phase identifies the set $R(t)$ to be symmetric about the y-axis, if each robot position $r \in R(t)$ has a symmetric image $\phi(r) \in R(t)$. Similarly, a robot can identify whether the set F is symmetric about the y-axis or not. An active robot in its *look* phase identifies the configuration to be symmetric about the y-axis if both the sets F and $R(t)$ are symmetric about the y-axis. Since the robots have an agreement on the direction and orientation of the y-axis, the configuration can not admit translational or rotational symmety.

- **Partitioning of configurations:** When the robots have an agreement on the y-axis, all the configurations can be partitioned into the following disjoint classes-

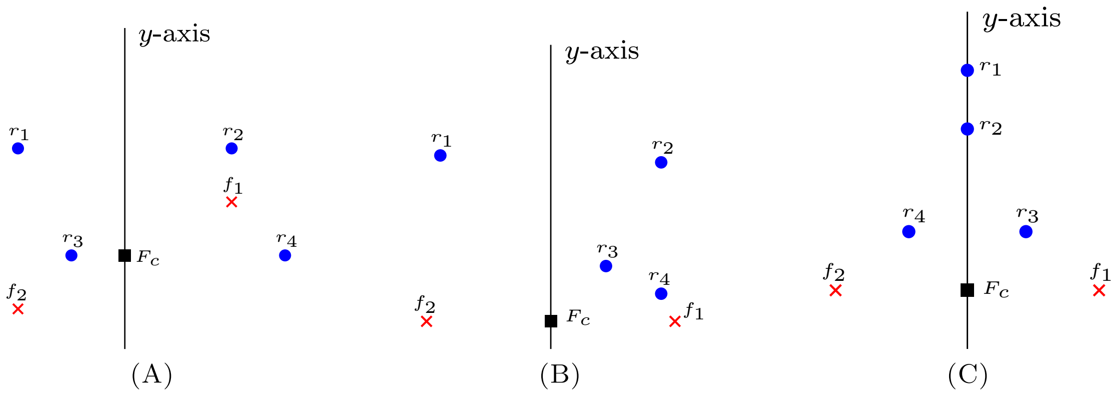


FIGURE 3.1: Black square represents the center of gravity, blue circles represent robot positions, and red crosses represent fixed points. (**A**) $\mathcal{I}_1$-configuration. (**B**) $\mathcal{I}_2$-configuration. (**C**) $\mathcal{I}_3$-configuration.

1. $\boldsymbol{\mathcal{I}_1}$**–** All configurations for which the y-axis is not a line of symmetry for F (Figure 3.1(A)).
2. $\boldsymbol{\mathcal{I}_2}$**–** All configurations for which the y-axis is a line of symmetry for F, but it is not a line of symmetry for $R(t)$ (Figure 3.1(B)).
3. $\boldsymbol{\mathcal{I}_3}$**–** All configurations for which the y-axis is a line of symmetry for $F \cup R(t)$ and $R_y(t) \neq \emptyset$, i.e., there exists a robot position on the y-axis (Figure 3.1(C)).
4. $\boldsymbol{\mathcal{I}_4}$**–** All configurations for which the y-axis is a line of symmetry for $F \cup R(t)$. Also, $F_y = \emptyset$ and $R_y(t) = \emptyset$, i.e., there are no robot positions and fixed points on the y-axis (Figure 3.2(A)).
5. $\boldsymbol{\mathcal{I}_5}$**–** All configurations for which the y-axis is a line of symmetry for $F \cup R(t)$. Also, $F_y \neq \emptyset$ and $R_y(t) = \emptyset$, i.e., there are no robot positions on the y-axis, but there are fixed points on the y-axis (Figure 3.2(B)).

Note that the classification of the configuration depends only on the y-axis and F_c. Since the y-axis and F_c are the same for all the robots, they can easily classify a configuration without conflict.

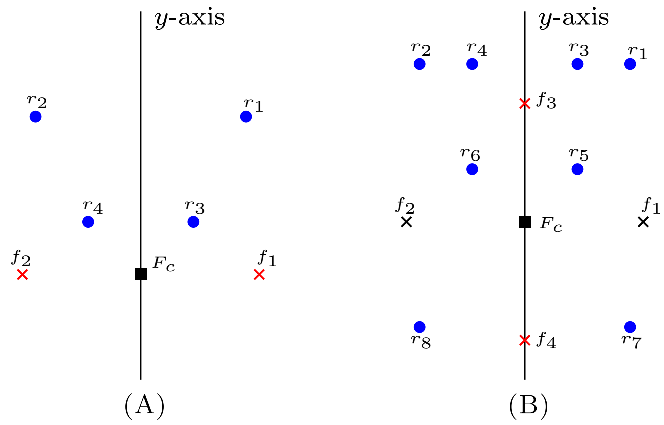


FIGURE 3.2: (**A**) $\mathcal{I}_4$-configuration. (**B**) $\mathcal{I}_5$-configuration.

3.2.1 Problem Definition

We call a configuration $C(t)$ *final* if the following conditions hold:

1. Every robot r_i is on a circle $C(f_j, \rho)$ for some $f_j \in F$,
2. $C(f_i, \rho) \cap C(f_j, \rho) = \emptyset$ for $f_i \neq f_j$,
3. Each circle contains exactly k robots at distinct positions.

The k-*circle formation* problem asks the robots to reach and remain in the *final* configuration, starting from an *initial* configuration.

The problem definition requires distinct robot positions in a *final* configuration. If a collision occurs among the robots, the result is a matter of assumptions. Under the assumption that a point of multiplicity will be created, the robots on a multiplicity point cannot be deterministically separated. Thus, collision avoidance is a fundamental requirement for solving the k-*circle formation* problem.

3.3 Impossibility Results

In this section, we characterize the *initial* configurations for which the k-*circle formation* problem cannot be solved deterministically. If k is an odd integer and the *initial* configuration $C(0) \in \mathcal{I}_5$, then $|F|$ must be even. For an *initial* configuration $C(0)$ which is symmetric about the y-axis, if both the values of k and $|F|$ are odd, then $R_y(0) \neq \emptyset$. As a result, $C(0)$ can not possibly belong to $\mathcal{I}_5$.

Theorem 3.3.1. *If the initial configuration $C(0) \in \mathcal{I}_5$ and k is an odd integer, then the k-circle formation problem is deterministically unsolvable.*

Proof. If possible, let algorithm $\mathcal{A}$ solve the *k-circle formation* problem starting from the given *initial* configuration $C(0) \in \mathcal{I}_5$ when k is odd. Consider the scheduler to be semi-synchronous with the additional property that whenever a robot r is activated, $\phi(r)$ is also activated. We assume that all the robots move with constant speed (which is the same for all robots) without transient stops. We also assume that the distance traveled by r is the same as that by $\phi(r)$. First, consider that both r and $\phi(r)$ have opposite notions of positive x-axis direction. As a result, their views would be identical. Since they run the same algorithm, their destinations and the corresponding paths would be mirror images. Even with non-rigid motion, if they travel the same distance, their final positions would be mirror images of each other. Since we started with a symmetric configuration, no algorithm can break the symmetry under this setup. Let $f \in F_y$. Since the overall configuration is symmetric, the robot positions on $C(f, \rho)$ must be symmetric around the y-axis. As k is odd, $C(f, \rho)$ must contain a robot position on the y-axis. Since the *initial* configuration did not have any robot position on the y-axis and all the robots move in pairs, having a robot r moved to the y-axis would mean moving $\phi(r)$ to the same point. As a result, a point of multiplicity will be created, from which it is deterministically impossible to separate r and $\phi(r)$. Hence, the *k-circle formation* problem is deterministically unsolvable. □

Notice that the unsolvability criterion (Theorem 3.3.1) for the *k-circle formation* problem would never be satisfied when k is an even integer. Even for odd values of k and the symmetric configurations in $\mathcal{I}_3 \cup \mathcal{I}_4$, the unsolvability criterion (Theorem 3.3.1) for the *k-circle formation* problem would never be satisfied.

3.4 *AlgorithmOneAxis*

In this section, we propose a deterministic distributed algorithm that solves the k-circle formation problem for the remaining configurations. Each active robot will execute the proposed algorithm $AlgorithmOneAxis(C(t))$ unless $C(t)$ is a *final* configuration. Each robot will follow the following steps during an execution of $AlgorithmOneAxis(C(t))$:

1. The robots identify the current configuration. The robots agree upon the positive direction of the x-axis in some configurations.

2. One or two *unsaturated* fixed points are selected for the circle formation, referred to as *target* fixed points.

3. The robots identify one or two robots for each *target* fixed point, referred to as *candidate* robots.

4. Each *candidate* robot moves towards the k-circle centered at its *target* fixed point.

Definition 3.4.1. *Let f_i be the unsaturated fixed point, which has the highest rank in $\mathcal{H}_1$ at time $t \geq 0$. Similarly, suppose $f_j \in \mathcal{H}_2$ is the the unsaturated fixed point, which has the highest rank at time $t \geq 0$. We say that there has been more progress in $\mathcal{H}_1$ than $\mathcal{H}_2$ at time t if one of the following conditions holds:*

1. $\gamma(f_i) < \gamma(f_j)$ *or*

2. $\gamma(f_i) = \gamma(f_j)$ *and* $D_i(t) < D_j(t)$ *or*

3. $\gamma(f_i) = \gamma(f_j)$ *and* $D_i(t) = D_j(t)$ *and* $d(f_i, r_1(t)) < d(f_j, r_2(t))$ *where* r_1 *and* r_2 *are candidate robots for* f_i *and* f_j*, respectively.*

Otherwise, we say that there has been the same progress in both the half-planes.

3.4.1 *AgreementOneAxis*

Since the robots have an agreement on the direction and orientation of the y-axis, they also have an agreement on the orientation of the x-axis without direction. This is the subprocedure by which the robots identify the configurations in which they could have an agreement on the direction of the x-axis. The robots make an agreement on the direction of the x-axis in such configurations. We have the following cases:

1. $\boldsymbol{C(t)} \in \boldsymbol{\mathcal{I}_1}$, i.e., F is asymmetric about the y-axis. Let $hline_1, \ldots, hline_s$ denote all the horizontal lines, each one of which passes through at least one fixed point, listed according to their increasing y-coordinates. Since the fixed points are asymmetric about the y-axis, at least one of these lines must contain asymmetric fixed points. Let $hline_v$ be the topmost among such horizontal lines which contains an asymmetric fixed point. Consider the fixed point closest to the y-axis and not having a symmetric image on $hline_v$. The direction from the y-axis towards the half-plane containing this fixed point is considered as the positive x-direction. All the robots agree upon this agreement.

2. $\boldsymbol{C(t)} \in \boldsymbol{\mathcal{I}_2}$, i.e., F is symmetric about the y-axis, but $R(t)$ is asymmetric about the y-axis. The robots consider the following cases:

   (a) The configuration is *unbalanced*. The direction from the y-axis, towards the half-plane containing the maximum number of robots, is considered as the positive x-direction. All the robots agree upon this agreement.

   (b) The configuration is *balanced* and all the fixed points in one of the half-planes are either *saturated* or *oversaturated*. In this case the robots consider the positive x-direction towards the half-plane in which all the fixed points are either *saturated* or *oversaturated*.

   (c) The configuration is *balanced* with at least one *unsaturated* fixed point in both the half-planes and $R_y(t) \neq \emptyset$. The robots do not make an agreement on the direction of positive x-axis. The robots decide to transform the configuration into an *unbalanced* configuration. Let r be the topmost robot on the y-axis. Define $\lambda = \max\limits_{f \in F,\, r_i \in R(t) \setminus \{r\}} d(r_i(t), f)$. Suppose p denotes the point on the y-axis, which is at 2λ distance above from topmost horizontal line $hline_s$. If the position of r is below p, then it moves towards p along the y-axis. Otherwise, r is moved to one of the half-planes to a point at $\frac{1}{3}\rho$ from the y-axis. This upward movement is required to avoid any collision, which might arise due to the inherent motion of r in a half-plane for some $t' \geq t$.

(d) The configuration is *balanced* with at least one *unsaturated* fixed point in both the half-planes and $R_y(t) = \emptyset$. Consider the following cases:

(i) k is odd and $F_y \neq \emptyset$. Note that in this case, the configuration has an even number of fixed points. The direction from the y-axis towards the half-plane in which there has been more *progress* is considered as the positive x-axis direction. It is possible that initially there has been the same *progress* in both the half-planes. Since $C(0)$ is asymmetric, there must be one asymmetric robot position about the y-axis. The positive x-direction is considered towards the half-plane that contains the asymmetric robot position, which has the highest configuration rank. All the robots agree upon this agreement.

(ii) Otherwise, the robots do not agree upon the direction of positive x-axis direction. This case includes the configurations in which (i) k is even and $F_y \neq \emptyset$, (ii) k is even and $F_y = \emptyset$, and (iii) k is odd and $F_y = \emptyset$. Notice that a configuration in this case might become symmetric with $R_y(t) = \emptyset$. Since the robots are oblivious, they would identify the configuration to be in $\mathcal{I}_4$ or $\mathcal{I}_5$, in which they can not make an agreement on the direction of positive x-axis. This decision of not to agree upon the direction of positive x-axis direction would ensure that the robots follow the same strategy in both symmetric and asymmetric cases.

3. $\boldsymbol{C(t) \in \mathcal{I}_3}$, i.e., $F \cup R(t)$ is symmetric about the y-axis and $R_y(t) \neq \emptyset$. Since $R(t)$ is symmetric about the y-axis, the configuration is *balanced*. The robots decide to transform the configuration into an *unbalanced* configuration. The robots follow the same strategy as described in the case of a *balanced* $\mathcal{I}_2$ configuration with at least one *unsaturated* fixed point in both the half-planes and $R_y(t) \neq \emptyset$ (case 2(c)).

4. $\boldsymbol{C(t) \in \mathcal{I}_4}$, i.e., $F \cup R(t)$ is symmetric about the y-axis, and $F_y = \emptyset$ and $R_y(t) = \emptyset$. Since $R(t)$ is symmetric about the y-axis, the configuration is *balanced*. As there are no robot positions on the y-axis, the configuration cannot be transformed into an *unbalanced* configuration. The robots can not have an agreement on the direction of positive x-axis direction in this case.

5. $\boldsymbol{C(t) \in \mathcal{I}_5}$, i.e., $F \cup R(t)$ is symmetric about the y-axis, and $F_y \neq \emptyset$ and $R_y(t) = \emptyset$. In this case, we have a *balanced* configuration. Since there are no robot positions on the y-axis, the configuration cannot be transformed into an *unbalanced* configuration. Note that k is an even integer in this case. Otherwise, the k-circle formation problem is unsolvable. The robots can not have an agreement on the direction of positive x-axis direction in this case.

3.4.2 *TargetFPSelection*

This is the subprocedure by which the robots select a *target* fixed point for the k-circle formation. The robots consider the following cases:

1. **Robots have an agreement on the positive direction of the x-axis**. Among the *unsaturated* fixed points, let f_j be the one, which has the highest configuration rank. The robots select f_j as the *target* fixed point.

2. **Robots do not have an agreement on the positive direction of the x-axis**. The robots consider the following cases:

   (a) All the fixed points in $F \setminus F_y$ are *saturated*. Among the *unsaturated* fixed points in F_y, let f_j be the topmost one. The robots select f_j as the *target* fixed point.

   (b) There exists an *unsaturated* fixed point in $F \setminus F_y$. If all the fixed points in one of the half-planes delimited by the y-axis are *saturated* or *oversaturated*, then the robots shall have an agreement on the positive direction of the x-axis. So assume that *unsaturated* fixed points are present in both the half-planes. In this case, the robots select two *target* fixed points, one from each of the half-planes. Let f_j and f_u be the *unsaturated* fixed points, which have the highest configuration rank in their respective half-planes. The robots select f_j and f_u as the *target* fixed points. Note that f_j and f_u may be symmetric images of each other.

3.4.3 $CandidateRSelection$

This is the subprocedure by which the robots select a *candidate* robot for a *target* fixed point. Let f_j be the *target* fixed point. Consider the following cases:

1. **There exists a robot position which lies within ρ distance from f_j.** Let $r_i \in R_\rho$ be the closest robot from $C(f_j, \rho)$. The robots select r_i as the *candidate* robot for f_j. If there are multiple such robots, then the robots select the one which has the highest configuration rank.

2. **There does not exist a robot position which lies within ρ distance from f_j.** Let r_i be the closest robot from f_j, which does not lie on a *saturated* circle. The robots select r_i as the *candidate* robot for f_j. If there are multiple such robots, then the robots select the one, which has the highest configuration rank. Note that r_i might lie on an *oversaturated* circle.

Note that, if f_j lies on the y-axis, and $C(t)$ does not have an agreement on the x-axis, then there may be two robots (say r_1 and r_2) having the same configuration rank, which are closest from f_j (case 2) or closest from $C(f_j, \rho)$ (case 1). In case, the configuration is asymmetric, let r_k be a robot position, which does not have a symmetric image about the y-axis. If there are multiple such robots, then the robots select the one, which has the highest configuration rank. The *candidate* robot is selected, from the half-plane, which contains r_k. Otherwise, both r_1 and r_2, are selected as the *candidate* robots. In case, f_j lies in a half-plane and $C(t)$ does not have an agreement on the x-axis, then the *candidate* robot is selected from the same half-plane in which it belongs.

3.4.4 $MovetoDestination$

This is the subprocedure by which a *candidate* robot r_i computes its *destination point* $q(t)$ on the circle centered at its *target* fixed point f_j and the *movement path* P along which it will move towards its *destination* point. The pseudocode of this subprocedure is given in Subprocedure 3.1. Let $p(t)$ denote the intersection point between $C(f_j, \rho)$ and $\overline{r_i(t)f_j}$. During its movement towards the circle centered at its *target* fixed point f_j, a

Subprocedure 3.1: $MovetoDestination(C(t), f_j, r_i)$

Input: $C(t), f_j, r_i$
Output: *Movement path* P and *destination point* $q(t)$

if $d(r_i(t), f_j) < \rho$ **then**
  Let $l_{ji}(t)$ be the line segment from f_j to $C(f_j, \rho)$, passing through r_i;
  Let q be the intersection point between $l_{ji}(t)$ and $C(f_j, \rho)$;
  if q *is not a robot position* **then**
    r_i selects $P = \overline{r_i q}$ and $q(t) = q$;
    r_i starts moving towards q along $\overline{r_i q}$;
  else
    if *there does not exist any robot positions on* $C(f_j, \rho)$ *other than being collinear with* r_i *and* f_j **then**
      Let B_1 be the ray starting from $r_i(t)$ such that $\measuredangle l_{ji}(t) r_i(t) B_1 = \frac{\pi}{4}$;
      Let q_1 be the intersection point between $C(f_j, \rho)$ and B_1;
      r_i selects $P = \overline{r_i q_1}$ and $q(t) = q_1$;
      r_i starts moving towards q_1 along $\overline{r_i q_1}$;
    else
      Let r_u be the robot on $C(f_j, \rho)$ such that $\measuredangle \overline{r_i(t)q} r_i(t) \overline{r_i(t) r_u(t)}$ is smallest;
      Let B_2 be the ray starting from $r_i(t)$ such that $\measuredangle \overline{r_i(t)q} r_i(t) B_2 = \frac{1}{2} min(\frac{\pi}{2}, \measuredangle \overline{r_i(t)q} r_i(t) \overline{r_i(t) r_u(t)})$;
      Let q_2 be the intersection point between $C(f_j, \rho)$ and B_2;
      r_i selects $P = \overline{r_i q_2}$ and $q(t) = q_2$;
      r_i starts moving towards q_2 along $\overline{r_i q_2}$;
    end
  end
else
  Let $p(t)$ be the intersection point between $C(f_j, \rho)$ and $\overline{r_i f_j}$;
  if $\overline{r_i f_j}$ *does not cut any saturated circle* **then**
    if $p(t)$ *is not a robot position* **then**
      r_i selects $P = \overline{r_i p(t)}$ and $q(t) = p(t)$;
      r_i starts moving towards $p(t)$ along $\overline{r_i p(t)}$;
    else if *there does not exist any robot positions on* $C(f_j, \rho)$ *other than being collinear with* r_i *and* f_j **then**
      Let t^a be one of the tangents from r_i to $C(f_j, \rho)$;
      Let t^a intersects $C(f_j, \rho)$ at q;
      r_i selects $P = \overline{r_i q}$ and $q(t) = q$;
      r_i starts moving towards q along $\overline{r_i q}$;
    else
      Let r_k be the robot position on $C(f_j, \rho)$ such that $\measuredangle \overline{r_i(t) r_k(t)} r_i(t) \overline{r_i(t) f_j}$ is the smallest;
      Let B_1 be the ray starting from $r_i(t)$ such that $\measuredangle \overline{r_i(t) r_k(t)} r_i(t) B_1 = \dfrac{1}{2} \measuredangle \overline{r_i(t) r_k(t)} r_i(t) \overline{r_i(t) f_j}$;
      Let q_1 be the intersection point between $C(f_j, \rho)$ and B_1;
      r_i selects $P = \overline{r_i q_1}$ and $q(t) = q_1$;
      r_i starts moving towards q_1 along $\overline{r_i q_1}$;
    end
  else
    Let $C(f_u, \rho)$ be the first *saturated* circle which r_i cuts while moving along $\overline{r_i f_j}$;
    Let q be the intersection point between $\overline{r_i f_j}$ and $C(f_u, \rho)$ which is at closest distance from r_i;
    if q *is not a robot position* **then**
      r_i selects $P = \overline{r_i q}$ and $q(t) = q$;
      r_i starts moving towards q along $\overline{r_i q}$;
    else
      Let r_k be the robot on $C(f_u, \rho)$ such that $\measuredangle \overline{r_i(t) f_j} r_i(t) \overline{r_i(t) r_k(t)}$ is the smallest;
      Let B_1 be the ray from $r_i(t)$ such that
      $$\measuredangle \overline{r_i(t) f_j} r_i(t) B_1 = \frac{1}{2} min(\measuredangle \overline{r_i(t) f_j} r_i(t) t^a, \measuredangle \overline{r_i(t) f_j} r_i(t) \overline{r_i(t) r_k(t)});$$
      Let q_1 be the intersection point between B_1 and $C(f_u, \rho)$ which is at closest distance from r_i;
      r_i selects $P = \overline{r_i q_1}$ and $q(t) = q_1$;
      r_i starts moving towards q_1 along $\overline{r_i q_1}$;
    end
  end
end

candidate robot must avoid collision with the other robots. In order to ensure collision-free movement, a *candidate* robot considers the following cases:

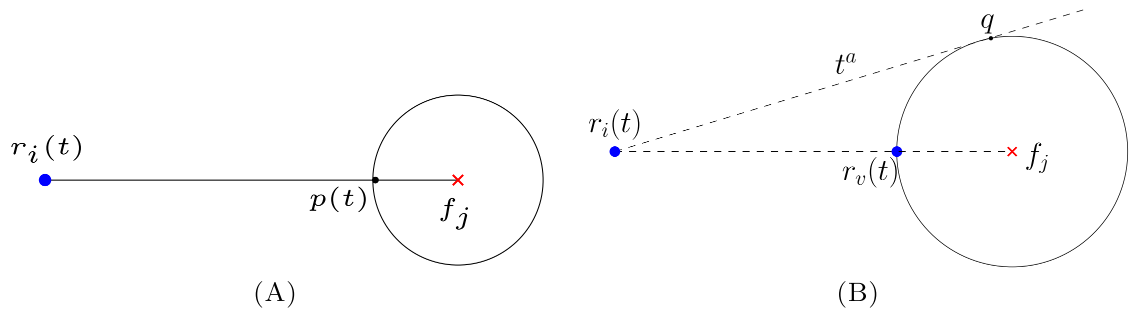


FIGURE 3.3: (**A**) $P = \overline{r_i(t)p(t)}$ and $q(t) = p(t)$. (**B**) $r_v(t)$ is the robot position on $p(t)$. $q(t) = q$ and $P = \overline{r_i(t)q}$, where q is the point of intersection between t^a and $C(f_j, \rho)$.

1. $\boldsymbol{d(r_i(t), f_j) > \rho}$ **and** $\overline{r_i(t)f_j}$ **does not cut any** ***saturated*** **circle**. If $p(t)$ is not a robot position, then r_i selects $q(t) = p(t)$ and $P = \overline{r_i(t)p(t)}$ (Figure 3.3(A)). Next, consider the case when $p(t)$ is a robot position and there are no other robot positions on $C(f_j, \rho)$ other than those collinear with r_i and f_j. In this case, r_i selects one of the tangent lines to $C(f_j, \rho)$ from its position (say t^a) as its *movement path*. Let t^a intersect $C(f_j, \rho)$ at q. In this case q can not be a robot position. Since r_i is a *candidate* robot, the line segement $\overline{r_i(t)q}$ can not possibly contain any robot positions other than $r_i(t)$. It selects $P = t^a$ and $q(t) = q$ (Figure 3.3(B)). Otherwise, among the robot positions on $C(f_j, \rho)$ which are not collinear with r_i and f_j, let r_k be the robot such that the angle $\measuredangle \overline{r_i(t)f_j} r_i(t) \overline{r_i(t)r_k(t)}$ is smallest. Let B_1 be the angle bisector such that $\measuredangle \overline{r_i(t)f_j} r_i(t) B_1 = \frac{1}{2} \measuredangle \overline{r_i(t)f_j} r_i(t) \overline{r_i(t)r_k(t)}$. Note that B_1 intersects $C(f_j, \rho)$ at exactly two points. Between these two points, let q_1 be the closest point from r_i. By the choice of r_k, q_1 can not be a robot position. Also, since r_i is a *candidate* robot, the line segment $\overline{r_i(t)q_1}$ can not possibly contain any robot positions other than $r_i(t)$. It selects $q(t) = q_1$ and $P = \overline{r_i(t)q_1}$ (Figure 3.4).

2. $\boldsymbol{d(r_i(t), f_j) > \rho}$ **and** $\overline{r_i(t)f_j}$ **cuts some** ***saturated*** **circle**. Let $C(f_u, \rho)$ be the first *saturated* circle, which r_i cuts while moving along $\overline{r_i(t)f_j}$. Notice that $\overline{r_if_j}$ would intersect $C(f_u, \rho)$ at two points. Consider q to be the intersection point between $C(f_u, \rho)$ and $\overline{r_i(t)f_j}$, which is at the closest distance from r_i. Since r_i is a *candidate* robot, the line segment $\overline{r_i(t)q}$ (excluding point q) can not possibly contain any robot positions other than $r_i(t)$. However, since q is a point on $C(f_u, \rho)$, it may be

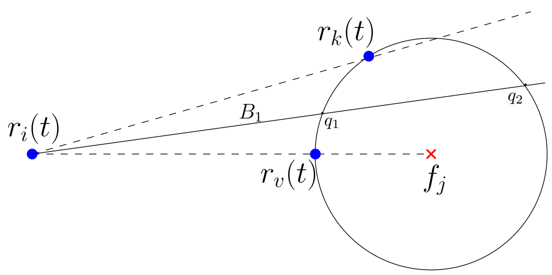


FIGURE 3.4: B_1 is the angle bisector of $\measuredangle \overline{r_i(t)f_j}r_i(t)\overline{r_i(t)r_k(t)}$. It intersects $C(f_j,\rho)$ at q_1 and q_2. In this case, r_i selects $P = \overline{r_i(t)q_1}$ and $q(t) = q_1$.

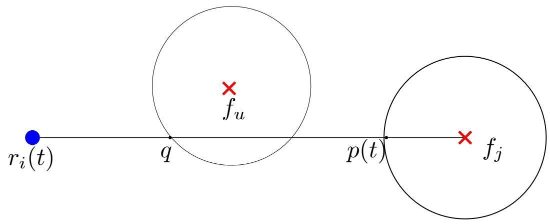


FIGURE 3.5: $P = \overline{r_i(t)q}$ and $q(t) = q$, where q is the point of intersection between $\overline{r_i(t)f_j}$ and $C(f_u,\rho)$.

a robot position. If q is not a robot position, then r_i selects $q(t) = q$ and $P = \overline{r_i(t)q}$ (Figure 3.5). Otherwise, let r_k (not collinear with r_i and f_j) be the robot on $C(f_u,\rho)$ such that angle between $\overline{r_i(t)f_j}$ and $\overline{r_i(t)r_k(t)}$ is the smallest. Since $C(f_u,\rho)$ is *saturated*, such a robot position always exists on it. Let B_1 be the angle bisector, such that $\measuredangle \overline{r_i(t)f_j}r_i(t)B_1 = \frac{1}{2}min(\measuredangle \overline{r_i(t)f_j}r_i(t)t^a, \measuredangle \overline{r_i(t)f_j}r_i(t)\overline{r_i(t)r_k(t)})$. Note that B_1 intersects $C(f_u,\rho)$ at exactly two points. Between these two points, let q_1 be the closest point from r_i. By the choice of r_k, q_1 can not be a robot position. Also, since r_i is a *candidate* robot $\overline{r_i(t)q_1}$ can not possibly contain any robot positions other than $r_i(t)$. Robot r_i selects $P = \overline{r_i(t)q_1}$ and $q(t) = q_1$ (Figure 3.6). Note that the choice of B_1 ensures that r_i always moves towards $C(f_j,\rho)$.

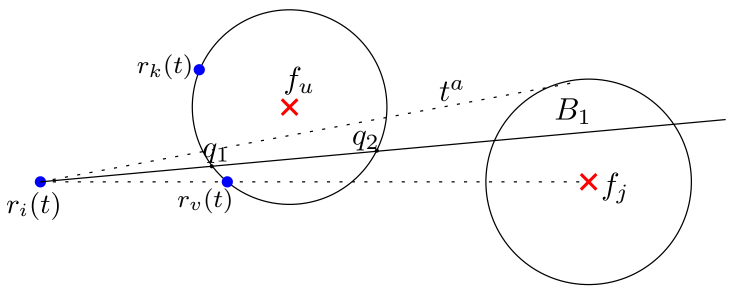


FIGURE 3.6: B_1 is the angle bisector of $\measuredangle \overline{r_i(t)f_j}r_i(t)t^a$. It intersects $C(f_u,\rho)$ at q_1 and q_2. In this case, r_i selects $P = \overline{r_i(t)q_1}$ and $q(t) = q_1$..

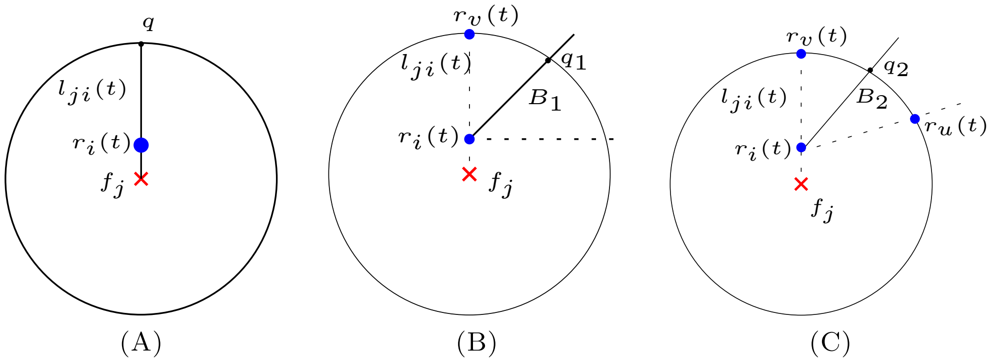


FIGURE 3.7: (**A**) $P = \overline{r_i(t)q}$ and $q(t) = q$, where q is the intersection point between $l_{ji}(t)$ and $C(f_j, \rho)$. (**B**) $q = r_v(t)$. B_1 is the ray starting from $r_i(t)$ such that $\measuredangle \overline{r_i(t)r_v(t)}r_i(t)B_1 = \frac{\pi}{4}$. $P = \overline{r_i(t)q_1}$ and $q(t) = q_1$, where q_1 is the intersection point between B_1 and $C(f_j, \rho)$. (**C**) $q = r_v(t)$. B_2 is the ray starting from $r_i(t)$ such that $\measuredangle \overline{r_i(t)r_v(t)}r_i(t)B_2 = \frac{1}{2}\measuredangle \overline{r_i(t)r_v(t)}r_i(t)\overline{r_i(t)r_u(t)}$. $P = \overline{r_i(t)q_2}$ and $q(t) = q_2$, where q_2 is the intersection point between B_2 and $C(f_j, \rho)$.

3. $\boldsymbol{d(r_i(t), f_j) < \rho}$. Let $l_{ji}(t)$ be the line segment from f_j to $C(f_j, \rho)$, passing through r_i. Let q be the intersection point between $l_{ji}(t)$ and $C(f_j, \rho)$. Since r_i is a *candidate* robot, the line segment $\overline{r_i(t)q}$ (excluding point q) can not possibly contain any robot positions other than $r_i(t)$. However, since q is a point on $C(f_j, \rho)$, it may be a robot position. If q is not a robot position, then r_i selects $q(t) = q$ and $P = \overline{r_i(t)q}$ (Figure 3.7(A)). Next, consider the case when q is a robot position and $C(f_j, \rho)$ does not contain any robot positions other than being collinear with r_i and f_j. Let B_1 be the ray starting from $r_i(t)$ such that $\measuredangle \overline{r_i(t)q}r_i(t)B_1 = \frac{\pi}{4}$ (Figure 3.7(B)). Suppose B_1 intersects $C(f_j, \rho)$ at q_1. The *candidate* robot r_i selects $q(t) = q_1$ and $P = \overline{r_i(t)q_1}$. Otherwise, let r_u (not collinear with r_i and f_j) be the robot position on $C(f_j, \rho)$ such that $\measuredangle \overline{r_i(t)q}r_i(t)\overline{r_i(t)r_u(t)}$ is the smallest. Let B_2 be the ray starting from $r_i(t)$ such that $\measuredangle \overline{r_i(t)q}r_i(t)B_2 = \dfrac{1}{2}min(\frac{\pi}{2}, \measuredangle \overline{r_i(t)q}r_i(t)\overline{r_i(t)r_u(t)})$. Suppose q_2 is the intersection point between B_2 and $C(f_j, \rho)$. The *candidate* robot selects $q(t) = q_2$ and $P = \overline{r_i(t)q_2}$ (Figure 3.7(C)).

In case there are exactly two *candidate* robots, which lie in different half-planes, each of them computes its *destination point* and *movement path* by ensuring that during its movement, it does not cross the y-axis. For example, consider the case when the *target* fixed point lies on the y-axis. A *candidate* robot will consider the tangent line and robot positions, which lie in its half-plane, while computing its *destination point* and *movement path*.

3.4.5 *AlgorithmOneAxis*

AlgorithmOneAxis is the proposed algorithm that solves the *k-circle formation* problem with one axis agreement. The pseudocode is given in Algorithm 3.2. Given $C(t)$, each active robot executes $AlgorithmOneAxis(C(t))$. During an execution of algorithm $AlgorithmOneAxis(C(t))$, if $C(t)$ is not a *final* configuration, then an active robot (say r_k) executes $AgreementOneAxis(C(t))$. Next, r_k considers the following cases:

ALGORITHM 3.2: *AlgorithmOneAxis*

```
Input: C(t) = (R(t), F)
Let r_k be an active robot at time t;
r_k executes AgreementOneAxis(C(t));
if the robots have an agreement on the positive direction of the x-axis then
    r_k executes TargetFPSelection(C(t));
    Let f_j be the target fixed point;
    r_k executes CandidateRSelection(C(t), f_j);
    Let r_i be the candidate robot;
    if r_k = r_i then
        r_k executes MovetoDestination(C(t), f_j, r_k);
    end
else
    if all the fixed points in F \ F_y are saturated then
        r_k executes TargetFPSelection(C(t));
        Let f_j be the target fixed point;
        r_k executes CandidateRSelection(C(t), f_j);
        if there is a unique candidate robot then
            Let r_i be the candidate robot;
            if r_k = r_i then
                r_k executes MovetoDestination(C(t), f_j, r_k);
            end
        else
            Let r_i be the candidate robot such that r_k and r_i lie in the same half-plane;
            if r_k = r_i then
                r_k executes MovetoDestination(C(t), f_j, r_k);
            end
        end
    else
        r_k executes TargetFPSelection(C(t));
        Let f_j and f_b be the target fixed points;
        r_k executes CandidateRSelection(C(t), f_j) and CandidateRSelection(C(t), f_b);
        Let r_i and r_a be the candidate robots of f_j and f_b, respectively;
        if r_k = r_i then
            r_k executes MovetoDestination(C(t), f_j, r_k);
        else if r_k = r_a then
            r_k executes MovetoDestination(C(t), f_b, r_k);
        end
    end
end
```

1. The robots have an agreement on the positive direction of the x-axis. Robot r_k executes $TargetFPSelection(C(t))$. In this case there is a unique *target* fixed point. Let f_j be the *target* fixed point. Next, r_k identifies the *candidate* robot by executing $CandidateRSelection(C(t), f_j)$. Let r_i be the *candidate* robot selected for f_j. If $r_k = r_i$, then the robot r_k executes $MovetoDestination(C(t), f_j, r_i)$.

2. The robots do not have any agreement on the positive direction of the x-axis. Robot r_k considers the following cases:

   (a) All the fixed points in $F \setminus F_y$ are *saturated.* The robot r_k executes sub-procedure $TargetFPSelection(C(t))$. In this case the unique *target* fixed point lies on the y-axis. Let f_j be the *target* fixed point. Robot r_k executes $CandidateRSelection(C(t), f_j)$. Let r_i be the *candidate* robot. Note that there may be two *candidate* robots for f_j. In that case, suppose r_i is the *candidate* robot, that lies in the same half-plane containing r_k. If $r_k = r_i$, then it executes $MovetoDestination(C(t), f_j, r_i)$.

   (b) There exists an *unsaturated* fixed point in $F \setminus F_y$. Note that such *unsaturated* fixed points are present in both the half-planes. Otherwise the robots would have an agreement on the positive direction of the x-axis. Robot r_k executes $TargetFPSelection(C(t))$. In this case there are two *target* fixed points, one from each of the half-planes. Let f_j and f_u be the two *target* fixed points. Without loss of generality, assume that r_k and f_j lie in the same half-plane. Next, r_k executes $CandidateRSelection(C(t), f_j)$. Let r_i be the *candidate* robot selected for f_j. If $r_k = r_i$, then the robot r_k executes sub-procedure $MovetoDestination(C(t), f_j, r_i)$.

3.5 Correctness of $AlgorithmOneAxis$

Lemma 3.5.1. *Given a configuration $C(t)$ for some $t \geq 0$, if the robots agree upon the positive direction of the x-axis, by the execution of $AgreementOneAxis(C(t))$, then the agreement remains invariant at any arbitrary point of time $t' > t$.*

Proof. Let the robots agree upon the positive direction of the x-axis, by the execution of $AgreementOneAxis(C(t))$. Consider the following cases:

Case 1. $C(t) \in \mathcal{I}_1$, i.e., F is asymmetric about the y-axis. Since this agreement is w.r.t. the fixed points, it remains invariant for any $t' > t$.

Case 2. $C(t) \in \mathcal{I}_2$ and $C(t)$ is *unbalanced.* In this case, the agreement on the direction of the positive x-axis is based upon robot positions. If the robots move across the y-axis from the negative side to the positive side, then the agreement does not change as the positive side of the y-axis would still contain the maximum number of robots. During an execution of $TargetFPSelection(C(t))$, the *unsaturated* fixed points with a higher configuration rank are given preference over the *unsaturated* fixed points with a lower configuration rank. As a result, the robots move across the y-axis from the positive side to the negative side, only when all the fixed points on the positive side of the y-axis are either *saturated* or *oversaturated.* Due to this movement, the configuration would transform into a *balanced* configuration. Next, case 3 would follow.

Case 3. $C(t) \in \mathcal{I}_2$ is a *balanced* configuration and all the fixed points in one of the half-planes are either *saturated* or *oversaturated.* Notice that a *candidate* robot, selected by the execution of $CandidateRSelection(C(t))$, would never lie on a *saturated* circle. As a result, once a circle becomes *saturated*, it would never become *unsaturated.* Thus, all the fixed points on the positive side of the y-axis would never become *unsaturated.* This implies that at any $t' > t$ the agreement on the positive direction of the x-axis remains invariant.

Case 4. $C(t) \in \mathcal{I}_2$ is a *balanced* configuration with at least one *unsaturated* fixed point in both the half-planes. Also, k is odd and $F_y \neq \emptyset$. In this case, the positive x-axis direction is considered towards the half-plane in which there has been more *progress* at time t. During an execution of $TargetFPSelection(C(t))$, the *unsaturated* fixed points with higher configuration rank are given preference over the *unsaturated* fixed points with lower configuration rank. As a result, it is guaranteed to have more *progress* in the positive side of the y-axis for any $t' > t$. Therefore, for any $t' > t$ the agreement on the positive direction of the x-axis remains invariant. In case $t = 0$, it might be possible that both the half-planes have the same *progress.* Since $C(0)$ is asymmetric about the y-axis in this case, there exists at least one robot asymmetric robot position. The positive x-axis direction is considered towards the half-plane, which contains the asymmetric robot with the highest configuration rank. For any $t' > t$, either $C(t') = C(0)$ or it is guaranteed to have more *progress* in the positive side of the y-axis. Therefore, the agreement on the positive direction of the x-axis remains invariant.

Hence, if the robots agree upon the positive direction of the x-axis by the execution of $AgreementOneAxis(C(t))$, then at any arbitrary point of time $t' > t$ the agreement remains invariant. □

Next, we consider the *balanced* configurations in which the robots make an agreement on the positive direction of the x-axis at some $t' > 0$. Lemma 3.5.1 ensures that the agreement remains invariant for any $t'' > t'$. Note that, at any arbitrary point of time $t \in [0, t')$, the robots have selected two *target* fixed points, one from each of the half-planes. Since the scheduler is assumed to be asynchronous, it is possible to have a *candidate* robot on the negative side of the y-axis, selected at some $t \in [0, t')$ and which has not reached its *destination point* at t'. We need to ensure that there would not be any collision due to the inherent motion of such a *candidate* robot.

Lemma 3.5.2. *Let $C(t')$ for some $t' > 0$, be the configuration in which the robots make an agreement on the positive direction of the x-axis. Let r_i be the candidate robot on the negative side of the y-axis, that was selected for some target fixed point f_j at $t \in [0, t')$. If t'' is the point of time at which it re-computes its destination point, then it would avoid collisions with any other candidate robots in the time interval $[t', t'']$.*

Proof. Let f_a be the *target* fixed point at some $t \in [t', t'']$. Since the robots have agreement on the positive direction of the x-axis, a unique *candidate* robot would be selected by the execution of $CandidateRSelection(C(t), f_a)$. Let r_b be the *candidate* robot. Note that, $f_a \geq f_j$ i.e., the configuration rank of f_j can not be higher than f_a. Otherwise, f_j would have been selected as the *target* fixed point. Consider the following cases:

Case 1. $f_a = f_j$. In this case $r_a = r_i$. This is because r_i is the *candidate* robot that was selected for f_j at $t \in [0, t')$ and has not reached $C(f_j, \rho)$. It would remain as the closest robot position from f_j, that does not lie on a *saturated* circle. Since r_i would be the unique robot which is in motion within $d(f_j, r_i)$ distance from f_j, there would not be any collision of robots.

Case 2. So we assume that $f_j \neq f_a$. The *movement paths* of r_i and r_b would not intersect. Otherwise, by triangle inequality r_i would have been at closer distance from f_a. So r_i would have selected as the *candidate* robot for f_a by the execution of subprocedure $CandidateRSelection(C(t), f_a)$. Since the *movement paths* do not intersect, r_i would

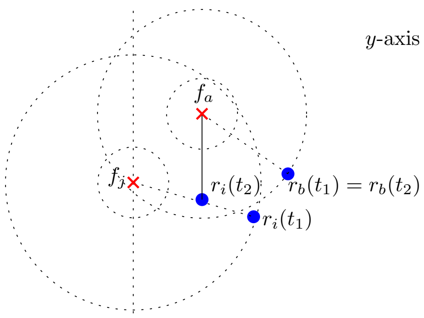


FIGURE 3.8: Robot r_i has moved from $r_i(t_1)$ to $r_i(t_2)$. It becomes a *candidate* robot for f_a at time t_2.

not collide with r_b during the time interval $[t', t'']$. Since the scheduler is assumed to be asynchronous, it is possible that r_i becomes the *candidate* robot for f_a as in Figure 3.8. As the *movement paths* do not intersect, r_i would continue its movement towards $C(f_j, \rho)$ without collision unless it stops and re-computes its *destination point*. If it stops it will execute $MovetoDestination(C(t), f_a, r_i)$. It computes its *movement path* towards $C(f_a, \rho)$ that does not intersect with the *movement path* of r_b. As a result, it would continue its movement towards $C(f_a, \rho)$ in subsequent time without any collision with r_b.

Hence, r_i would avoid collisions with any other *candidate* robots in the time interval $[t', t'']$. $\square$

Theorem 3.3.1 characterizes all the configurations and the values of k for which the k-*circle formation* problem is deterministically unsolvable. For some $k > 0$, if the k-*circle formation* problem is deterministically solvable for a given $C(0)$, the robots can identify it in its *look* phase. The robots must ensure that such configurations would not transform into an configuration that would satisfy the unsolvability criterion (Theorem 3.3.1) for any $t > 0$ during an execution of *AlgorithmOneAxis*.

Lemma 3.5.3. *Given $k > 0$ and $C(0)$, if the k-circle formation problem is deterministically solvable, then at any arbitrary point of time $t > 0$ the configuration would not satisfy the unsolvability criterion (Theorem 3.3.1).*

Proof. Since the k-*circle formation* problem is deterministically solvable for every even value of k, we assume that k is odd. Note that all the *initial* configurations, in which F is asymmetric about the y-axis or in which $F_y = \emptyset$, would never satisfy the unsolvability criterion stated in Theorem 3.3.1. So we only need to consider all the *initial* configurations in which F is symmetric about the y-axis and $F_y \neq \emptyset$. So, $C(0) \notin \mathcal{I}_1 \cup \mathcal{I}_4$. Also, $C(0) \notin \mathcal{I}_5$ (Otherwise, initially it would have been unsolvable). Therefore, $C(0) \in \mathcal{I}_2 \cup \mathcal{I}_3$. We have the following cases:

Case 1. The robots make an agreement on the positive direction of x-axis, which remains invariant for any $t > 0$ (Lemma 3.5.1). Since the agreement remains invariant, even if the configuration becomes symmetric about the y-axis, the configuration will not satisfy the unsolvability criterion stated in Theorem 3.3.1 for any $t > 0$.

Case 2. The robots decide to transform $C(0)$ into an *unbalanced* configuration, in order to make an agreement on the positive direction of x-axis. This includes the following configurations:

1. $C(0) \in \mathcal{I}_3$.
2. $C(0) \in \mathcal{I}_2$ and it is *balanced* with at least one *unsaturated* fixed point in both the half-planes and $R_y(t) \neq \emptyset$..

Let t' be earliest possible point of time at which it becomes *unbalanced*. In the time interval 0 to t', only the topmost robot on the y-axis would move along the y-axis. As a result, the configuration would not satisfy the unsolvability criterion (Theorem 3.3.1) for any $t \in [0, t')$. At t', the robots make an agreement on the positive direction of x-axis. Next, the proof follows from case 1.

Therefore, $C(0)$ would not transform into an unsolvable configuration at time $t > 0$. $\square$

Given a configuration $C(t)$, let $n_k(t)$ denote the number of *unsaturated* fixed points. The robots may select one or two *target* fixed points. First, consider the case when the *target* fixed point is unique. Suppose, f_j is the *target* fixed point and r_i its *candidate* robot selected by the robots. Let P and $q(t)$ be the *movement path* and *destination point*, respectively, computed by r_i at time t, by the execution of $MovetoDestination(C(t), f_j, r_i)$. Consider a straight line along P towards $C(f_j, \rho)$ intersecting the circle $C(f_j, \rho)$ first at

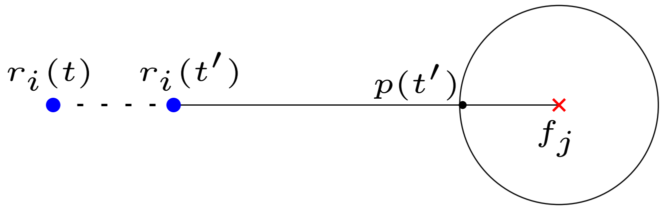


FIGURE 3.9: Robot r_i has moved from $r_i(t)$ to $r_i(t')$, along $P = \overline{r_i(t)p(t)}$ towards $q(t) = p(t)$ computed at time t. Robot r_i selects $P' = \overline{r_i(t')p(t')}$ and $q(t') = p(t')$ at time t'. In this case, $q(t') = q(t)$. Also, $q(t) = s(t)$ and $q(t') = s(t')$, i.e., the *destination point* lies on $C(f_j, \rho)$.

$s(t)$ (The line would always intersect $C(f_j, \rho)$) at time t. Suppose $d_j(t)$ denotes the distance between $r_i(t)$ and $s(t)$. Recall that $D_j(t)$ denote the deficit in the number of robots in order to make f_j a *saturated* fixed point. Let $V_j(t) = (n_k(t), D_j(t), d_j(t))$.

We say that there has been *significant progress* in the time interval t to t' if $V_j(t') < V_j(t)$, i.e., one of the following conditions holds:

1. $n_k(t') < n_k(t)$, or
2. $n_k(t') = n_k(t)$ and $D_j(t') < D_j(t)$, or
3. $n_k(t') = n_k(t)$ and $D_j(t') = D_j(t)$ and $d_j(t') + \delta \leq d_j(t)$.

Lemma 3.5.4. *Let t' be an arbitrary point of time before r_i reaches its destination computed at time t. During an execution of AlgorithmOneAxis$(C(t))$, execution of MovetoDestination$(C(t), f_j, r_i)$ ensures that $d_j(t') + \delta \leq d_j(t)$.*

Proof. Let P and P' be the selected *movement paths* for r_i at time t and t', respectively. We have $d_j(t) = d(r_i(t), s(t))$ and $d_j(t') = d(r_i(t'), s(t'))$. Note that $q(t) = s(t)$ implies that the *destination point* lies on $C(f_j, \rho)$. Consider the following cases:

Case 1. $q(t) = s(t)$ and $p(t)$ does not contain any robot position. This is the case where the robot moves straight towards f_j, i.e., $P = \overline{r_i(t)f_j}$ and the destination point $q(t)$ lies on $C(f_j, \rho)$ (Step 25 of Subprocedure 3.1). At time t' there would not be any robot on $q(t)$ and r_i would continue along the same path. Since δ is the minimum displacement in a round, $d_j(t') + \delta \leq d_j(t)$. Recall that $p(t)$ denotes the intersection point between $C(f_j, \rho)$ and $\overline{r_i f_j}$. The movements are shown in Figure 3.9.

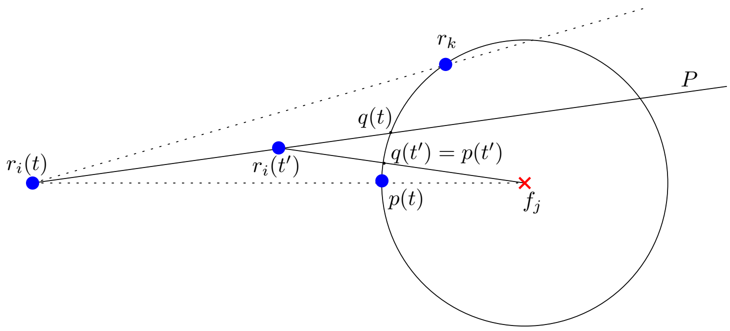


FIGURE 3.10: Robot r_i has moved from $r_i(t)$ to $r_i(t')$, along P towards $q(t)$ computed at time t. Robot r_i selects $P' = \overline{r_i(t')p(t')}$ and $q(t') = p(t')$ at time t'. Also, $q(t) = s(t)$ and $q(t') = s(t')$, i.e., the *destination point* lies on $C(f_j, \rho)$.

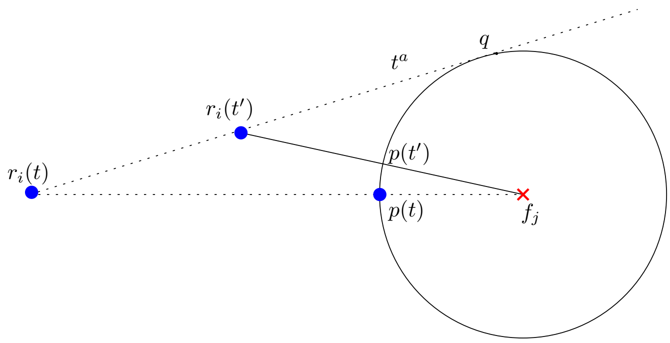


FIGURE 3.11: Robot r_i has moved from $r_i(t)$ to $r_i(t')$, along $P = \overline{r_i(t)q}$ towards $q(t) = q$ computed at time t (q is the point of intersection between $C(f_j, \rho)$ and t^a). Robot r_i selects $P' = \overline{r_i(t')p(t')}$ and $q(t') = p(t')$ at time t'. Also, $q(t) = s(t)$ and $q(t') = s(t')$, i.e., the *destination point* lies on $C(f_j, \rho)$.

Case 2. $q(t) = s(t)$ and $p(t)$ contains a robot position. There are robot positions on $C(f_j, \rho)$, that are not collinear with r_i and $p(t)$. By step 36 of Subprocedure 3.1 robot r_i computes the *movement path* P and destination point $q(t)$. It starts moving towards $q(t)$ along P. At time $t' > t$, let $s(t')$ be the intersection point between $C(f_j, \rho)$ and $\overline{r_i(t')f_j}$. Note that, $p(t')$ is not a robot position. Robot r_i selects $P' = \overline{r_i(t')f_j}$ and $q(t') = p(t')$. We have $d(r_i(t'), q(t)) > d(r_i(t'), q(t'))$ and $d(r_i(t), q(t)) - d(r_i(t'), q(t')) > d(r_i(t), q(t)) - d(r_i(t'), q(t)) \geq \delta$. This implies that $d_j(t') + \delta \leq d_j(t)$. The movements are shown in Figure 3.10.

Case 3. $q(t) = s(t)$ and $p(t)$ contains a robot position. There are no robots on $C(f_j, \rho)$, other than being collinear with r_i and f_j. By step 30 of Subprocedure 3.1 robot r_i computes the *movement path* P and destination point $q(t)$. This case is similar to case 2. The movements are shown in Figure 3.11.

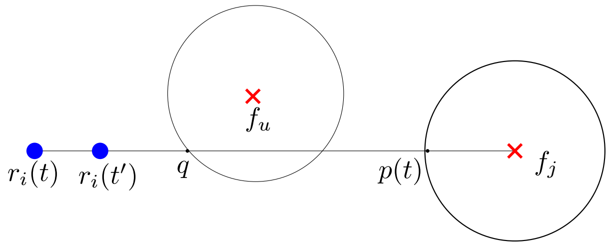


FIGURE 3.12: $C(f_u, \rho)$ is a *saturated* circle and q is the point of intersection between $C(f_u, \rho)$ and $\overline{r_i(t)f_j}$, which is at closest distance from r_i. Robot r_i has moved from $r_i(t)$ to $r_i(t')$, along $P = \overline{r_i(t)q}$ towards $q(t) = q$ computed at time t. Robot r_i selects $P' = \overline{r_i(t')q}$ and $q(t')$ on $C(f_u, \rho)$ at time t'. In this case, $q(t') = q(t)$. Also, $q(t) \neq s(t)$ and $q(t') \neq s(t')$, i.e., the *destination point* does not lie on $C(f_j, \rho)$.

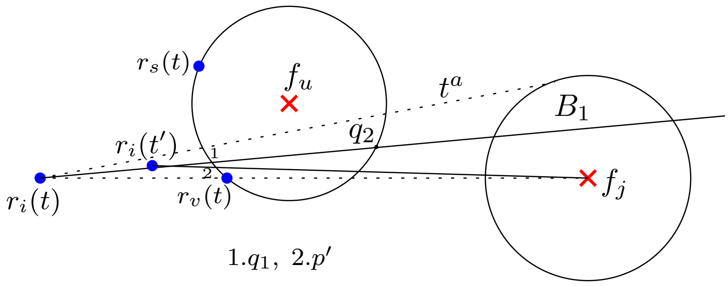


FIGURE 3.13: Robot r_i has moved from $r_i(t)$ to $r_i(t')$, along $P = \overline{r_i(t)q_1}$ towards $q(t) = q_1$ computed at time t. Robot r_i selects $P' = \overline{r_i(t)q}$ and $q(t') = p'$ (p' is the point of intersection between $C(f_u, \rho)$ and $\overline{r_i(t')f_j}$) on $C(f_u, \rho)$ at time t'

Case 4. $q(t) \neq s(t)$. In this case $q(t)$ lies on a *saturated* circle $C(f_u, \rho)$ for some $f_u \neq f_j$. Note that, $C(f_u, \rho)$ is the first circle, that r_i cuts while moving along $\overline{r_i(t)f_j}$. First, consider the case in which $P = \overline{r_i(t)q}$ and $q(t) = q$ (Step 43 of Subprocedure 3.1), where q is intersection point between $\overline{r_i(t)f_j}$ and $C(f_u, \rho)$, which is at closest distance from r_i. Since δ is the minimum displacement in a round, $d_j(t') + \delta \leq d_j(t)$. The movements are shown in (Figure 3.12). Next, consider the case in which r_i computes its *movement path* P by step 49 of Subprocedure 3.1. It starts moving towards $q(t)$ along path P. At time $t' > t$, let p' be the intersection point between $C(f_u, \rho)$ and $\overline{r_i(t')f_j}$. Note that p' is not a robot position. Robot r_i selects $P' = \overline{r_i(t')f_j}$ and $q(t') = p'$ (Figure 3.13). We have $d(r_i(t'), s(t)) > d(r_i(t'), s(t'))$ and $d(r_i(t), s(t)) - d(r_i(t'), s(t')) > d(r_i(t), s(t)) - d(r_i(t'), s(t)) \geq \delta$. This implies that $d_j(t') + \delta \leq d_j(t)$.

Case 5. $d(r_i, f_j) < \rho$. We have $q(t) = s(t)$. Let q be the intersection point between $C(f_j, \rho)$ and $l_{ji}(t)$. First, consider the case when r_i selects $P = \overline{r_i(t)q}$ and $q(t) = q$ (Step 5 of Subprocedure 3.1). At time t', there would not be any robot position on $q(t)$. Robot r_i selects $P' = \overline{r_i(t')q}$. Since δ is the minimum displacement in a round, $d_j(t') + \delta \leq d_j(t)$.

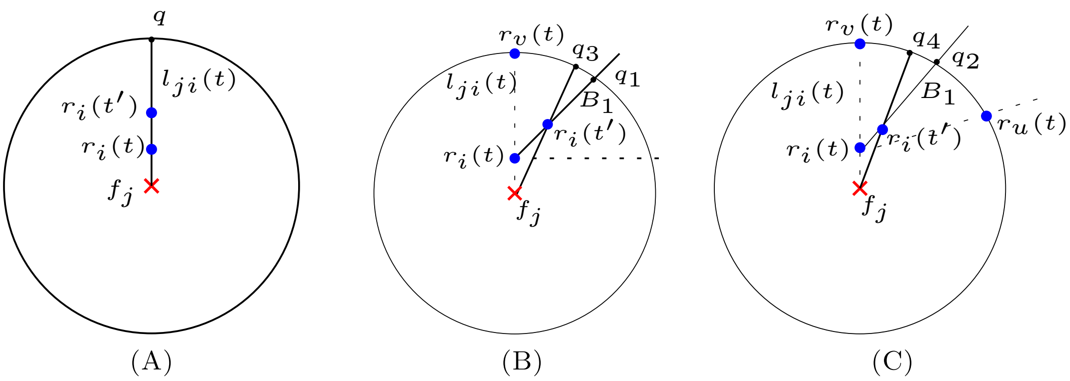


FIGURE 3.14: (**A**) Robot r_i has moved from $r_i(t)$ to $r_i(t')$ along $P = \overline{r_i(t)q}$ towards $q(t) = q$ (q is the point of intersection between $C(f_j, \rho)$ and $l_{ji}(t)$). It selects $P' = \overline{r_i(t')q}$ and $q(t') = q$. (**B**) At time t, r_i selects $P = \overline{r_i(t)q_1}$ and $q(t) = q_1$. It selects $P' = \overline{r_i(t)q_3}$ and $q(t') = q_3$. (**C**) At time t, r_i selects $P = \overline{r_i(t)q_2}$ and $q(t) = q_2$. It selects $\overline{r_i(t)q_4}$ and $q(t') = q_4$.

Movements are shown in Figure 3.14(A). Next, consider the case in which r_i selects its *movement path* P by step 11 or step 17 of Subprocedure 3.1. We have $d(r_i(t'), q(t)) > d(r_i(t'), q(t'))$ and $d(r_i(t), q(t)) - d(r_i(t'), q(t')) > d(r_i(t), q(t)) - d(r_i(t'), q(t)) \geq \delta$. Hence, $d_j(t') + \delta \leq d_j(t)$. Movements are shown in Figure 3.14(B) and 3.14(C).

Hence, execution of $MovetoDestination(C(t))$ ensures $d_j(t') + \delta \leq d_j(t)$. □

Lemma 3.5.5. *Let f_j be the target fixed point and r_i its candidate robot in the configuration $C(t)$. During an execution of $AlgorithmOneAxis(C(t))$, the execution of $MovetoDestination(C(t), f_j, r_i)$ ensures significant progress.*

Proof. Let r_i compute *movement path* P and *destination point* $q(t)$ by the execution of $MovetoDestination(C(t), f_j, r_i)$ at time t. Let $t' > t$ be an arbitrary point of time at which r_i has completed at least one LCM cycle. We need to show that there has been *significant* progress in between the time interval t to t'. We have the following cases:

Case 1. $r_i(t') = q(t)$ and r_i is on the $C(f_j, \rho)$. We have the following two sub-cases:

Subcase 1. If $C(f_j, \rho)$ has exactly k robots on it, then $n_k(t') = n_k(t) - 1$, ensuring *significant* progress.

Subcase 2. If $C(f_j, \rho)$ has less than k robots on it, then $D_j(t') = D_j(t) - 1$, ensuring *significant* progress.

Case 2. $r_i(t') \neq q(t)$ and r_i is not on any *oversaturated* $C(f_u, \rho)$. In this case $d_j(t') + \delta \leq d_j(t)$ by Lemma 3.5.4, which ensures *significant* progress.

Case 3. $r_i(t') \neq q(t)$ and r_i is on an *oversaturated* $C(f_u, \rho)$. Since at this stage, a *candidate* robot for f_j will be selected again, $CandidateRSelection(C(t'), f_j)$ will select a robot r_k such that $d(r_k(t'), f_j) \leq d(r_i(t'), f_j)$. Either $r_k = r_i$ or $r_k \neq r_i$. By Lemma 3.5.4, *significant* progress is ensured, in both the cases.

Hence, execution of $MovetoDestination(C(t), f_j, r_i)$ ensures *significant* progress. □

Lemma 3.5.6. *Let f_j be a target fixed point and r_i its unique selected candidate robot at time t. Until r_i reaches its destination point computed at time t, it remains the candidate robot for f_j.*

Proof. Let r_i compute its *movement path* P and *destination point* $q(t)$ by the execution of $MovetoDestination(C(t), f_j, r_i)$. Note that, $q(t)$ is either a point on the circle $C(f_j, \rho)$ or on some *saturated* circle $C(f_u, \rho)$. Let t' be an arbitrary point of time such that $r_i(t') \neq q(t)$. At time t', f_j remains an *unsaturated* fixed point. As a result, f_j remains a *target* fixed point at time t'. Lemma 3.5.4 guarantees that r_i has moved at least δ amount closer to $C(f_j, \rho)$. Therefore, it remains the *candidate* robot for f_j. □

Next, we consider the case when there are two *candidate* robots for a *target* fixed point. Since robots have an agreement on the directions and orientations of the y-axis, there can be at most two *candidate* robots at any point of time. Note that, in this case, the configuration would have a unique *target* fixed point, that lies on the y-axis.

Lemma 3.5.7. *Let f_j be the target fixed point and r_i and r_v are the two selected candidate robots for f_j at time t. Until at least one of them reaches its destination point computed at time t, no other robot becomes a candidate robot. If one of the candidate robots have reached its destination point and the other one has not, then the other robot either continues its inherent motion towards its destination point (computed at time t) without any collision or gets selected as a candidate robot only when $D_j(t)$ reduces by one.*

Proof. Let $t' > t$ be an arbitrary point of time when at least one of the *candidate* robots has completed its LCM cycle. Without loss of generality, assume that r_i has completed its

LCM cycle at t'. Let $q(t)$ be the *destination point* and P be the *movement path* computed for r_i by $MovetoDestination(C(t), f_j, r_i)$. Note that $q(t)$ is a point either on the $C(f_j, \rho)$ or on some *saturated* $C(f_u, \rho)$. We have the following cases:

Case 1. $q(t)$ is a point on the circle $C(f_j, \rho)$. We have the following subcases:

Subcase 1. $r_i(t') = q(t)$. Since r_i has reached its destination, the first part of the lemma follows. We have $D_j(t') = D_j(t) - 1$. At t', if r_v has also completed its LCM cycle and has not reached its destination point, then it becomes the next *candidate* robot for f_j. If r_v is in motion, then being the only robot in motion within the annulus region between $C(f_j, \rho)$ and $C(f_j, d(f_j, r_v(t')))$, it continues its motion without any collision. Note that, in this case, no other robot will be selected for movement until r_v reaches its destination.

Subcase 2. $r_i(t') \neq q(t)$. First consider that $|d(f_j, r_i(t')) - \rho| > |d(f_j, r_v(t')) - \rho|$, i.e., robot r_v is closer to $C(f_j, \rho)$ than r_i. At t', either r_v has also completed its LCM cycle and has not reached its destination point or r_v is in motion. In both the cases, r_v remains a *candidate* robot for f_j. The first part of the lemma follows for r_v. Robot r_i will be selected as a *candidate* robot when r_v will reach $C(f_j, \rho)$. Next consider that $|d(f_j, r_i(t')) - \rho| < |d(f_j, r_v(t')) - \rho|$, i.e., robot r_i is closer to $C(f_j, \rho)$ than r_v. Robot r_i will be selected as a *candidate* robot. At t', if r_v has also completed its LCM cycle, then it will become the *candidate* robot when r_i will reach $C(f_j, \rho)$. If r_v is in motion, then it continues its motion without any collision (As *destination point* and *movement path* computed by r_i and r_v respectively are separated by the y-axis and there are no other robots in the half-plane containing r_v, which is in motion within the annulus region between $C(f_j, \rho)$ and $C(f_j, d(f_j, r_v(t'))))$. We have two possible cases. First, r_v will also reach $C(f_j, \rho)$. Second, if it stops before reaching $C(f_j, \rho)$, then it will become a *candidate* robot only when r_i will reach $C(f_j, \rho)$.

Case 2. $q(t)$ is a point on some *saturated* circle $C(f_u, \rho)$. Consider the following cases:

Subcase 1. $r_i(t') = q(t)$. Since r_i has reached its destination, the first part of the lemma follows. At t', since $C(f_u, \rho)$ contains $k + 1$ robots, the next *candidate* robot for f_j will be selected from $C(f_u, \rho)$. Note that, this robot position would have higher y-coordinate than $q(t)$. If r_v has also completed its LCM cycle and has not reached its destination point, then it will become a *candidate* robot for f_j only when $D_j(t'') = D_j(t') - 1$ for some $t'' > t'$. If r_v is in motion, then it continues its motion without any collision (It is the only robot, which is in motion within the annulus region between $C(f_j, \rho)$ and $C(f_j, d(f_j, r_v(t')))$ and below $q(t)$).

Subcase 2. $r_i(t') \neq q(t)$. First consider that $|d(f_j, r_i(t')) - \rho| > |d(f_j, r_v(t')) - \rho|$, i.e., robot r_v is closer to $C(f_j, \rho)$ than r_i. At t', either r_v has also completed its LCM cycle and has not reached its destination point or r_v is in motion. In both cases, r_v remains a *candidate* robot for f_j. The first part of the lemma follows for r_v. Robot r_i will be selected as a *candidate* robot only when $D_j(t)$ reduces by one. Next consider that $|d(f_j, r_i(t')) - \rho| < |d(f_j, r_v(t')) - \rho|$, i.e., robot r_i is closer to $C(f_j, \rho)$ than r_v. Robot r_i will be selected as a *candidate* robot. At t', if r_v has also completed its LCM cycle and has not reached its destination point, then it will become a *candidate* robot only when $D_j(t)$ reduces by one. If r_v is in motion, then it continues its motion without any collision (As destination point and path computed by r_i and r_v, respectively, are separated by the y-axis and there are no other robots in motion within the annulus region between $C(f_j, \rho)$ and $C(f_j, d(f_j, r_v(t')))$ and below the point $q(t)$). We have two possible cases. First, r_v will also reach $C(f_u, \rho)$. Second, if it stops before reaching $C(f_u, \rho)$, then it will become a *candidate* robot only when $D_j(t)$ reduces by one. □

Next, we consider the case when there are two *target* fixed points, one from each half-plane. Let f_j and f_a be the *target* fixed points at time t. Let r_i and r_b be their respective *candidate* robots. We have $V_j(t) = (n_k(t), D_j(t), d_j(t))$ and $V_a(t) = (n_k(t), D_a(t), d_a(t))$.

Lemma 3.5.8. *Let a given configuration $C(t)$ admit two target fixed points during an execution of AlgorithmOneAxis$(C(t))$ and $t' > t$ be an arbitrary point of time when at least one candidate robot has completed its LCM cycle. For at least one target fixed point $f_i \in \{f_j, f_a\}$ and its candidate robot, $d_i(t') + \delta \leq d_i(t)$.*

Proof. Each *target* fixed point is unique in their respective half-planes. Execution of *AlgorithmOneAxis*$(C(t))$ ensures that for each *target* fixed point, its *candidate* robot is selected from its respective half-planes. The circle formation process continues independently in both the half-planes. This implies that for each $i \in \{j, a\}$, $V_i(t)$ is updated only due to the movement of f_i's *candidate* robot. Without loss of generality, suppose *candidate* robot r_i of the *target* fixed point f_j has completed its LCM cycle. By Lemma 3.5.4, $d_j(t') + \delta \leq d_j(t)$ is ensured. □

Lemma 3.5.9. *Let a given configuration $C(t)$ admit two target fixed points during an execution of $AlgorithmOneAxis(C(t))$ and $t' > t$ be an arbitrary point of time when at least one candidate robot has completed its LCM cycle. $AlgorithmOneAxis(C(t))$ ensures significant progress.*

Proof. Lemma 3.5.8 ensures that for at least one *target* fixed point $f_i \in \{f_j, f_a\}$ and its *candidate* robot, $d_i(t') + \delta \leq d_i(t)$ holds. Without loss of generality, assume that for the *target* fixed point f_j we have $d_j(t') + \delta \leq d_j(t)$ in the time interval t to t'. By Lemma 3.5.5, we have $V_j(t') < V_j(t)$, i.e., *significant* progress is ensured. □

Theorem 3.5.10. *If the initial configuration $C(0) \in \{\mathcal{I}_1 \cup \mathcal{I}_2 \cup \mathcal{I}_3 \cup \mathcal{I}_4 \cup \mathcal{I}_5\}$ and $C(0)$ does not satisfy the unsolvability criterion stated in Theorem 3.3.1, then the robots would eventually solve the k-circle formation problem under one axis agreement, by the execution of $AlgorithmOneAxis$.*

Proof. Lemma 3.5.3 guarantees that for any $t > 0$, the configuration $C(t)$ would not satisfy the unsolvability criterion stated in Theorem 3.3.1. We have the following cases:

Case 1. There is a unique *target* fixed point (say f_j) in the configuration. The Lemma 3.5.5 ensures that each time a *candidate* robot gets activated, *significant* progress is ensured. If there is a unique *candidate* robot for f_j, then Lemma 3.5.6 guarantees that until the *candidate* robot reaches its destination, it would remain the *candidate* robot. In case there are two *candidate* robots for f_j, then Lemma 3.5.7 guarantees that until one of the *candidate* robots reaches its destination point, no other robot will become a *candidate* robot. As a result, one of the *candidate* robots will reach its destination point eventually. If the other *candidate* robot does not reach its destination point, then it becomes a *candidate* robot for f_j when $D_j(t)$ reduces by one. Thus, the circle formation process around all the fixed points will be completed eventually.

Case 2. There are two *target* fixed points. Note that the *target* fixed points lie in different half-planes delimited by the y-axis. Lemma 3.5.9 ensures *significant* progress. Lemma 3.5.6 guarantees that until a *candidate* robot reaches its destination, it remains the *candidate* robot. Note that in this case for each of the *target* fixed points, always a unique *candidate* robot gets selected. Thus, the circle formation process around all the fixed points will be completed eventually.

Hence, the robots would eventually solve the *k-circle formation* problem with one axis agreement. □

From Theorem 3.5.10, it follows that the robots would solve the *k-circle formation* problem under one axis agreement within finite time. Since we have considered the scheduler to be ASYNC, the robots do not have any common notion of time. As a result, the actual time to solve the *k-circle formation* problem depends upon the scheduling of the robots. We use the notion of an *epoch* [118] to discuss the runtime complexity of our proposed algorithm. An epoch is the time interval in which all the robots in the configuration have performed their LCM cycles at least once. According to this definition, the time is divided into global *epochs*. We also assume that the robots have rigid motion, i.e., the robot is guaranteed to reach its destination whenever it moves. In such a setting, we have the following observations:

1. If a *candidate* robot does not have to pass through a *saturated* circle in order to reach the circle centered at its *target* fixed point, then it would reach the circle within one *epoch*.

2. If a *candidate* robot has to pass through a *saturated* circle in order to reach the circle centered at its *target* fixed point, then it would reach the circle in at most three *epochs*. This is because the *movement path* would intersect the *saturated* circle either one or two times.

From the above two observations, it follows that a *candidate* robot would reach the circle centered at its *target* fixed point within $2(m-1)+1 = 2m-1$ *epochs*. This is because, it might have to pass through $(m-1)$ number of *saturated* circles. Since *AlgorithmOneAxis* is sequential, each *target* fixed point would need at most $k(2m-1)$ *epochs* to become *saturated*. Therefore, the *k-circle formation* problem would be solved within $\mathcal{O}(m^2k)$ *epochs*. This is a loose upper bound on the running time of *AlgorithmOneAxis* in terms of *epochs*.

3.6 k-Circle Formation when $n > km$

In this section, we assume that there are $n > km$ robots in the Euclidean plane. As the definition of the *k-circle formation* problem requires k distinct robot positions on each circle, there will be $n - km$ surplus robots.

3.6.1 Impossibility Results when $n > km$

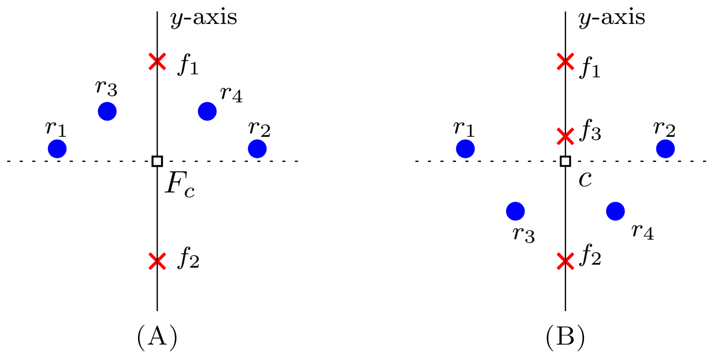


FIGURE 3.15: Examples of Impossibility Results when $n > km$. (A) $|F|$ is even, (B) $|F|$ is odd.

Theorem 3.6.1. *Given $C(0) \in \mathcal{I}_5$, if $R_y(0) = \emptyset$ and k is an odd integer, then the k-circle formation problem is unsolvable.*

Proof. The idea of proof is similar to the proof of Theorem 3.3.1. □

Figure 3.15 shows examples of configurations in which the *k-circle formation* problem is unsolvable. For both the configurations, $k = 1$ and $R_y(0) = \emptyset$. In Figure 3.15(A), $|F| = 2$ (even) and $n = 4 > 2 = km$, whereas in Figure 3.15(B), $|F| = 3$ (odd) and $n = 4 > 3 = km$.

3.6.2 Algorithm for the k-Circle Formation when $n > km$

The definition of a *final* configuration includes the criterion that each robot is located on a circle. However, there will be $n - km$ surplus robots present in the configuration. In this case, we define a configuration to be a *final with surplus* robots if the following conditions are satisfied:

1. $C(f_i, \rho) \cap C(f_j, \rho) = \emptyset$ for $f_i \neq f_j$,

2. Each circle contains exactly k robots at distinct positions.

Define algorithm *AlgoSurplus* as follows:

1. If the current configuration is not a *final with surplus* robots, then the robots will execute $AlgorithmOneAxis$.

2. Else terminate.

Theorem 3.6.2. *If $C(0) \in \{\mathcal{I}_1 \cup \mathcal{I}_2 \cup \mathcal{I}_3 \cup \mathcal{I}_4 \cup \mathcal{I}_5\}$ and $C(0)$ does not satisfy the unsolvability criterion stated in Theorem 3.6.1, then the robots would eventually solve the k-circle formation problem under one axis agreement, by the execution of AlgoSurplus.*

Proof. The idea of proof is similar to the proof of Theorem 3.5.10. □

3.7 k-Circle Formation when $n < km$

In this section, we assume that there are $n < km$ robots in the Euclidean plane. As the definition of the k-*circle formation* problem requires exactly k distinct robot positions on each circle and $n < km$, some fixed points will remain *unsaturated*. The objective is to maximize the number of *saturated* circles.

3.7.1 Impossibility Results when $n < km$

Let $m_1 = \dfrac{m - |F_y|}{2}$. If $C(t) \in \mathcal{I}_4$, then $m_1 = \dfrac{m}{2}$.

Theorem 3.7.1. *Let $C(0) \in \mathcal{I}_4$ be such that $R_y(0) = \emptyset$. If $k(p-1) < \dfrac{n}{2} < kp$ where $1 \leq p \leq m_1$, and $\dfrac{n}{2} - k(p-1) \geq \left\lceil \dfrac{k}{2} \right\rceil$, then the k-circle formation problem is unsolvable.*

Proof. The idea of proof is similar to the proof of Theorem 3.3.1. □

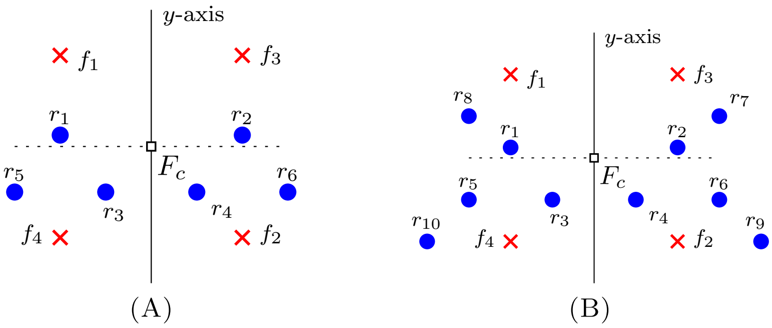


FIGURE 3.16: Examples of Impossibility Results when $n < km$ and $C(t) \in \mathcal{I}_4$. (A) k is even, (B) k is odd.

Figure 3.16 shows examples of configurations in which the k-*circle formation* problem is unsolvable as the unsolvability criterion stated in Theorem 3.7.1 is satisfied. For both the configurations, $C(t) \in \mathcal{I}_4$, $|F| = 4$ and $R_y(0) = \emptyset$. In Figure 3.16(A), $k = 2$ (even), $p = 2$ and $k(p-1) = 2 < \frac{n}{2} = 3 < kp = 4$. Also, $\frac{n}{2} - k(p-1) = 3 - 2 = 1 \geq \left\lceil \frac{k}{2} \right\rceil = 1$. In Figure 3.15(B), $k = 3$ (odd), $p = 2$ and $k(p-1) = 3 < \frac{n}{2} = 4 < kp = 6$. Also, $\frac{n}{2} - k(p-1) = 5 - 3 = 2 \geq \left\lceil \frac{k}{2} \right\rceil = 2$.

Theorem 3.7.2. *Let $C(0) \in \mathcal{I}_5$ such that $R_y(0) = \emptyset$ and k is an even integer. If the following conditions hold:*

1. $n > k|F_y|$,
2. $k(p-1) < \frac{n - k|F_y|}{2} < kp$ *where* $1 \leq p \leq m_1$, *and*
3. $\frac{n - k|F_y|}{2} - k(p-1) \geq \left\lceil \frac{k}{2} \right\rceil$,

then the k-circle formation problem is unsolvable.

Proof. The idea of proof is similar to the proof of Theorem 3.3.1. □

Theorem 3.7.3. *Let $C(0) \in \mathcal{I}_5$ be such that $R_y(0) = \emptyset$ and $k = 1$. If $n > 2m_1$, then the k-circle formation problem is unsolvable.*

Proof. The idea of proof is similar to the proof of Theorem 3.3.1. □

In the Figure 3.17(A), an example of a configuration $C(t) \in \mathcal{I}_5$ that satisfies the unsolvability criterion stated in Theorem 3.7.2. We have $k = 2$, $|F_y| = 2$, $n = 10 >$

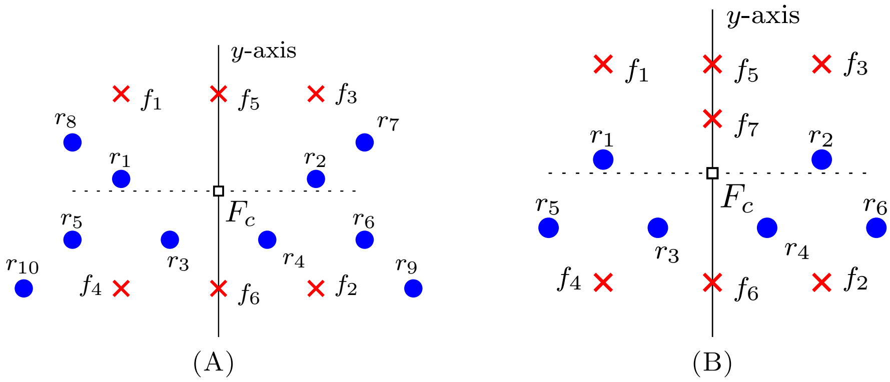


FIGURE 3.17: Examples of Impossibility Results when $n < km$ and $C(t) \in \mathcal{I}_5$. (A) $k = 2$, (B) $k = 1$.

$k|F_y| = 4$, $k(p-1) = 2(2-1) = 2 < \dfrac{n - k|F_y|}{2} = 3 < kp = 4$ where $p = 2$. Also, $\dfrac{n-k|F_y|}{2} - k(p-1) = 3 - 2 = 1 \geq \left\lceil \dfrac{2}{2} \right\rceil = 1$. Figure 3.17(B) shows an example of a configuration $C(t) \in \mathcal{I}_5$ in which $k = 1$ and $n = 6 > 2m_1 = 2.2 = 4$ satisfying the unsolvability criterion stated in Theorem 3.7.3.

Theorem 3.7.4. *Let $C(0) \in \mathcal{I}_5$ be such that $R_y(0) = \emptyset$. If $k > 1$ is an odd integer such that one of the following conditions holds:*

1. *$n > 2km_1$, or*
2. *$n < 2km_1$ and $k(p-1) < \dfrac{n}{2} < kp$ where $1 \leq p \leq m_1$, and $\dfrac{n}{2} - k(p-1) \geq \left\lceil \dfrac{k}{2} \right\rceil$,*

then the k-circle formation problem is unsolvable.

Proof. The idea of proof is similar to the proof of Theorem 3.3.1. □

Figure 3.18(A) shows an example of a configuration $C(t) \in \mathcal{I}_5$ with $n = 10 > 2km_1 = 6$, that satisfies the unsolvability criterion stated in Theorem 3.7.4. In the Figure 3.18(B), $C(t) \in \mathcal{I}_5$ with $n = 4 > 2km_1 = 12$. Also, $k(p-1) = 3.(1-1) = 0 < \dfrac{n}{2} = 2 < kp = 3$ where $p = 1$, and $\dfrac{n}{2} - k(p-1) = 3 - 0 = 3 \geq \left\lceil \dfrac{k}{2} \right\rceil = 2$. Figure 3.18(B) satisfies the unsolvability criterion stated in Theorem 3.7.4.

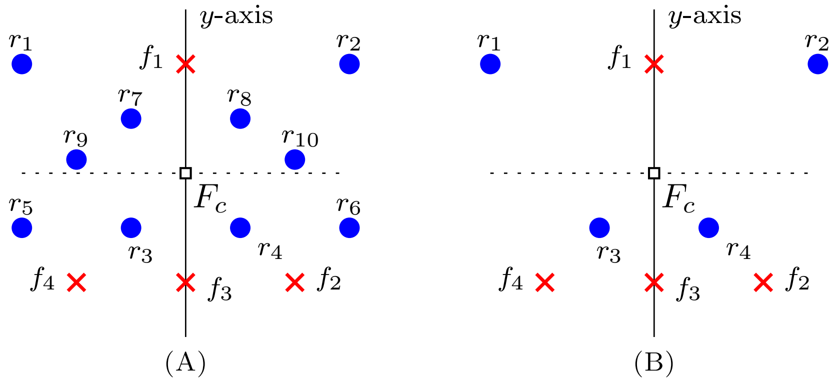


FIGURE 3.18: Examples of Impossibility Results when $n < km$, $k = 3$ and $C(t) \in \mathcal{I}_5$. (A) $n > 2km_1$, (B) $n < 2km_1$.

3.7.2 Algorithm for the k-Circle Formation when $n < km$

Suppose $n = kp_1 + p_2$ where $p_1 \geq 0$ and $0 \leq p_2 \leq k$. In this case, we define a configuration to be a *final with slack* robots if the following conditions are satisfied:

1. $C(f_i, \rho) \cap C(f_j, \rho) = \emptyset$ for $f_i \neq f_j$,
2. There are exactly p_1 number of *saturated* circles.

Define algorithm *AlgoSlack* as follows:

1. If the current configuration is not a *final with slack* robots, then execute algorithm *AlgorithmOneAxis*.
2. Else terminate.

Theorem 3.7.5. *If $C(0) \in \{\mathcal{I}_1 \cup \mathcal{I}_2 \cup \mathcal{I}_3 \cup \mathcal{I}_4 \cup \mathcal{I}_5\}$ and $C(0)$ does not satisfy the unsolvability criteria stated in Theorems 3.7.1, 3.7.3, 3.7.4 and 3.7.4 then the robots would eventually solve the k-circle formation problem under one axis agreement, by the execution of AlgoSlack.*

Proof. The idea of proof is similar to the proof of Theorem 3.5.10. □

3.8 Relationship between the k-Circle Formation problem and the k-EPF problem

Given $m > 0$ fixed points and $n = km$ robots for some positive integer k, the *k-EPF* problem asks exactly k robots to reach and remain in each fixed point. Since the definition of the *k-circle formation* problem asks for distinct robot positions, we only consider the *initial* configurations with distinct robot positions. We want to prove the following theorem.

Theorem 3.8.1. *For a given initial configuration with distinct robot positions and a positive integer k, if the k-circle formation problem is deterministically solvable then the k-EPF problem is also deterministically solvable.*

In order to prove the above theorem, we modify *AlgorithmOneAxis*, to solve the *k-EPF* problem deterministically within finite time.

3.8.1 Algorithm for the *k-EPF* problem

Let $C(0)$ be the given *initial* configuration. Suppose the *k-circle formation* problem has been solved in $C(t)$, for some $t \geq 0$, with radius ρ, by the execution of *AlgorithmOneAxis*. In order to solve the *k-EPF* problem, the robots must reach the fixed points. The robots can accomplish this by moving in a straight line towards the fixed point. Since the robots are oblivious, they do not remember any information about the past events. Therefore, if any robot stops before reaching the fixed point for some $t' > t$, it would not remember that the *k-circle formation* problem has already been solved. As a result, it will again start executing *AlgorithmOneAxis*. To resolve such a situation, consider the following definition. A configuration is said to satisfy *Property* 1, if the following conditions hold:

1. Each robot lies within ρ distance from some fixed point.

2. $\forall f_i \in F$, there are at most k robots, which lie within ρ distance from f_i.

Given a configuration which satisfies *Property* 1, let $\mathcal{A}$ be an algorithm as follows:

If there exists a robot r_i such that $0 < d(r_i, f_j) \leq \rho$ for some $f_j \in F$, then r_i moves along $\overline{r_i f_j}$ towards f_j.

Define algorithm $AlgokEPF$ as follows:

If the current configuration satisfies *Property* 1, then execute $\mathcal{A}$.

Else the robots execute $AlgorithmOneAxis$.

During an execution of $\mathcal{A}$, it must be ensured that none of the robots have any inherent motion, which is not directed towards the fixed point. Since all the robots are stationary in the *initial* configuration, if $C(0)$ satisfies *Property* 1, then none of the robots would have any inherent motion.

Lemma 3.8.2. *During an execution of* $AlgorithmOneAxis$ *if* $t > 0$ *is the earliest possible point of time at which the configuration* $C(t)$ *satisfies* Property 1*, then none of the robots would have any inherent motion in* $C(t)$*.*

Proof. Since $C(t)$ satisfies *Property* 1, each robot lies within ρ distance from some fixed point. Also, notice that there are no *oversaturated* circles in $C(t)$. Let f_j be the *target* fixed point which became *saturated* at time t due to the movement of a *candidate* robot (say r_i). Notice that if f_j lies on the y-axis and the configuration is symmetric, there would be two such *candidate* robots. In that case, we assume that both of them reached $C(f_j, \rho)$ at time t. Otherwise, the configuration $C(t)$ can not possibly satisfy *Property 1.* Suppose r_i became a *candidate* robot at some time $t_1 < t$ by the execution of $CandidateRSelection$. Note that in the time interval $[t_1, t)$, the distance of r_i from f_j was greater than ρ. Otherwise, the choice of t is wrong. If there were two *candidate* robots for f_j, then this is true for both the *candidate* robots. Also, at time t_1 there were no robot position (say r_a) such that $d(f_j, r_a(t)) < \rho$. Otherwise, r_a would have been selected as a *candidate* robot. Notice that the *candidate* robot(s) was the only robot which was moving towards $C(f_j, \rho)$. Therefore, all the robots on $C(f_j, \rho)$ are static at t. Next, consider a fixed point $f_l \in F$ such that f_l has higher configuration rank than f_j. All the robots within ρ distance from f_l must lie on $C(f_l, \rho)$. This is because, during an execution of $CandidateRSelection$ for a fixed point, a robot within ρ distance from that fixed point is given higher preference than any robot at greater than ρ distance from that fixed point. Since $C(f_l, \rho)$ is not *oversaturated*, all the robots are static at time t. Next, consider a fixed point $f_b \in F$ such that f_b has lower configuration rank than f_j. By the choice of f_j and r_i, none of the robots within ρ distance from f_b were selected as a *candidate* robot. Therefore, all the robots within ρ distance from f_b are static at time t. Hence, if the configuration $C(t)$ satisfies Property 1, then none of the robots have any inherent motion in $C(t)$. □

Theorem 3.8.3. *If the initial configuration $C(0) \in \{\mathcal{I}_1 \cup \mathcal{I}_2 \cup \mathcal{I}_3 \cup \mathcal{I}_4 \cup \mathcal{I}_5\}$ and $C(0)$ does not satisfy the unsolvability criterion stated in Theorem 3.3.1, then the robots would eventually solve the k-EPF problem under one axis agreement, by the execution of algorithm $AlgokEPF$.*

Proof. First, consider the case when the configuration does not satisfy *Property 1.* The robots would start executing *AlgorithmOneAxis*. From Theorem 3.5.10, it follows that the configuration would eventually satisfy *Property 1.* Next, consider the case when the configuration satisfies *Property 1.* From Lemma 3.8.2, it follows that all the robots would be static in such a configuration. The robots would start executing $\mathcal{A}$. During an execution of $\mathcal{A}$, each robot moves in a straight line by at least δ distance, towards the fixed point from which it is at the closest distance. Since ρ is finite and there are finitely many robots, eventually each of the fixed points will contain exactly k robots.

Hence, the robots would eventually solve the *k-EPF* problem by the execution of algorithm $AlgokEPF$. □

The above theorem provides an evidence that a deterministic distributed algorithm to solve the *k-circle formation* problem can be modified to solve the *k-EPF* problem and proves Theorem 3.8.1. Notice that during an execution of algorithm $AlgokEPF$, the robots are only allowed to create a multiplicity on the fixed points. Therefore, the existence of a deterministic distributed algorithm which solves the *k-EPF* problem, without allowing a robot multiplicity point outside the fixed points, is guaranteed by Theorem 3.8.1.

3.9 Conclusion

This chapter studies the *k-circle formation* problem by asynchronous, autonomous, anonymous and oblivious robots in the Euclidean plane. The problem is investigated in a setting where the robots have an agreement on the direction and orientation of the y-axis. The following three main results have been proved:

1. If the *initial* configuration $C(0)$ is symmetric about the y-axis such that $F_y \neq \emptyset$ (there are fixed points on the y-axis) and $R_y(0) = \emptyset$ (there are no robot positions on the the y-axis), then the *k-circle formation* problem is deterministically unsolvable for odd values of k. This is the complete set of the *initial* configurations and values of k for which the *k-circle formation* problem is deterministically unsolvable under this setting.

2. For the rest of the configurations and the values of k, a deterministic distributed algorithm has been proposed under one axis agreement.

3. All the *initial* configurations and values of k for which the problem is deterministically unsolvable are characterized when $n > km$.

4. All the initial configurations and values of k for which the problem is deterministically unsolvable are characterized when $n < km$.

5. It has also been shown that if the *k-circle formation* problem is deterministically solvable then the *k-EPF* problem is also deterministically solvable. This has been established by modifying *AlgorithmOneAxis*; the modified algorithm *Algokepf* deterministically solves the *k-EPF* problem.

Chapter 4

k-Circle Formation by Disoriented Robots

Contents

4.1 Overview

In this chapter, the *k-circle formation* problem is studied for completely *disoriented* robots. In other words, the robots neither have any agreement on a global coordinate system nor have any agreement on a common *chirality*. When the robots have an agreement on one axis, all the robots and fixed points can be ordered with respect to the axis of agreement. As a result, the presence of rotational symmetries can be managed. In this new setting, rotational symmetries must be considered in addition to the reflectional symmetry. The number of unsolvable cases would also increase significantly in this current setting. Due to rotational symmetry, there can be multiple numbers of moving robots at any particular point of time. To solve the problem in this setting, it must be ensured that the problem remains solvable throughout the execution of the algorithm. The assumption of an asynchronous scheduler adds more challenges in designing a distributed algorithm in order to solve the *k-circle formation* problem. In this setting, all the *initial* configurations and values of k for which the *k-circle formation* problem is deterministically unsolvable are characterized. A deterministic distributed algorithm is proposed that deterministically solves the *k-circle formation* problem for the remaining configurations and values of k.

4.2 Model and Definitions

The robots are *autonomous*, *anonymous*, *oblivious*, *homogeneous*, and *silent*. They operate in *Look-Compute-Move* cycles under a fair ASYNC scheduler. They are represented by points in the Euclidean plane. The robots are completely *disoriented*. While any value of the radius is acceptable, we take $\rho = \frac{1}{3}\rho_1$ as the common radii of the circles. Recall that ρ_1 denotes the minimum distance between any two fixed points.

4.2.1 Configuration View

Given $C(t) = (R(t), F)$, let $S = R(t) \cup F$ and $d_i = d(F_c, s_i)$ where $s_i \in S$. Let $Ray(F_c, s_i)$ denote the ray that starts from F_c and passes through $s_i \in S$. Let $S_i^+ = (s_1, s_2, \ldots, s_n)$ denote the list, in the order by which the points in S would be encountered if $Ray(F_c, s_i)$ is rotated by an angle of 2π in the clockwise direction. If multiple points are encountered simultaneously by the sweep line, then the point closest to F_c is considered at first. In case, a robot lies on a fixed point, then the robot position is given preference over the fixed point. Define a function $x : S \to \{r, f\}$ as follows:

$$x(s_j) = \begin{cases} r & \text{if } s_j \text{ is a robot position} \\ f & \text{if } s_j \text{ is a fixed point} \end{cases}$$

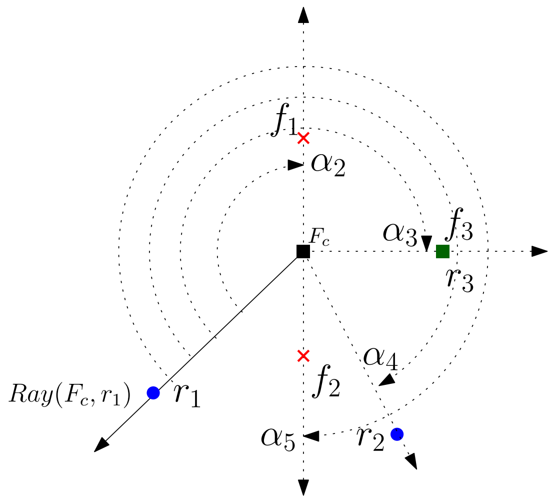


FIGURE 4.1: Green square represents robot positions on a fixed point. Illustration of configuration view of r_1.

Let α_j denote the angle by which $Ray(F_c, s_i)$ has been rotated when the j^{th} point in S_i^+ is being encountered. Define the clockwise view of s_i as

$$\mathscr{V}^+(s_i) = (\alpha_1, d_1, x(s_1), \alpha_2, d_2, x(s_2), \ldots, \alpha_n, d_n, x(s_n))$$

Similarly, the counter-clockwise view of s_i can be defined. For example, consider the view of r_1 in Figure 4.1. $S_1^+ = (r_1, f_1, r_3, f_3, r_2, f_2)$ is the list of points encountered while rotating $Ray(F_c, r_1)$ in the clockwise direction, starting from r_1.

$$\mathscr{V}^+(r_1) = (\alpha_1 = 0, d(F_c, r_1), r, \alpha_2, d(F_c, f_1), f, \alpha_3, d(F_c, r_3), r,$$
$$\alpha_3, d(F_c, f_3), f, \alpha_4, d(F_c, r_2), r, \alpha_5, d(F_c, f_2), f)$$

By defining $r < f$, the configuration views of all the points in S can be lexicographically ordered. The view of a point $p \in R(t) \cup F$ is given by $\mathscr{V}(p) = \min(\mathscr{V}^+(p), \mathscr{V}^-(p))$ and the view of a configuation is given by $\mathscr{V}(C(t)) = \cup_{p \in R(t) \cup F} \mathscr{V}(p)$. These definitions are similar to the configuration view defined in Cicerone et al. [12]. Note that, even though the robots do not have a common *chirality*, they get the same information about the configuration by computing $\mathscr{V}(C(t))$. The view of a set (say S) is defined as $\mathscr{V}(S) = \min_{s_i \in S}(\min(\mathscr{V}^+(s_i), \mathscr{V}^-(s_i)))$. A robot can determine whether a given configuration is symmetric or not by the following two results, proved in Cicerone et al. [12].

Lemma 4.2.1. *[12] Let $C(t) = (R(t), F)$ be a given configuration. The configuration*

$C(t)$ admits a line of symmetry if and only if there exists two points $p, q \in R(t) \cup F$, not necessarily distinct, such that $\mathscr{V}^+(p) = \mathscr{V}^-(q)$.

Lemma 4.2.2. *[12] Let $C(t) = (R(t), F)$ be a given configuration. The configuration $C(t)$ admits rotational symmetry if and only if there exists two distinct points $p, q \in R(t) \cup F$, such that $\mathscr{V}^+(p) = \mathscr{V}^+(q)$.*

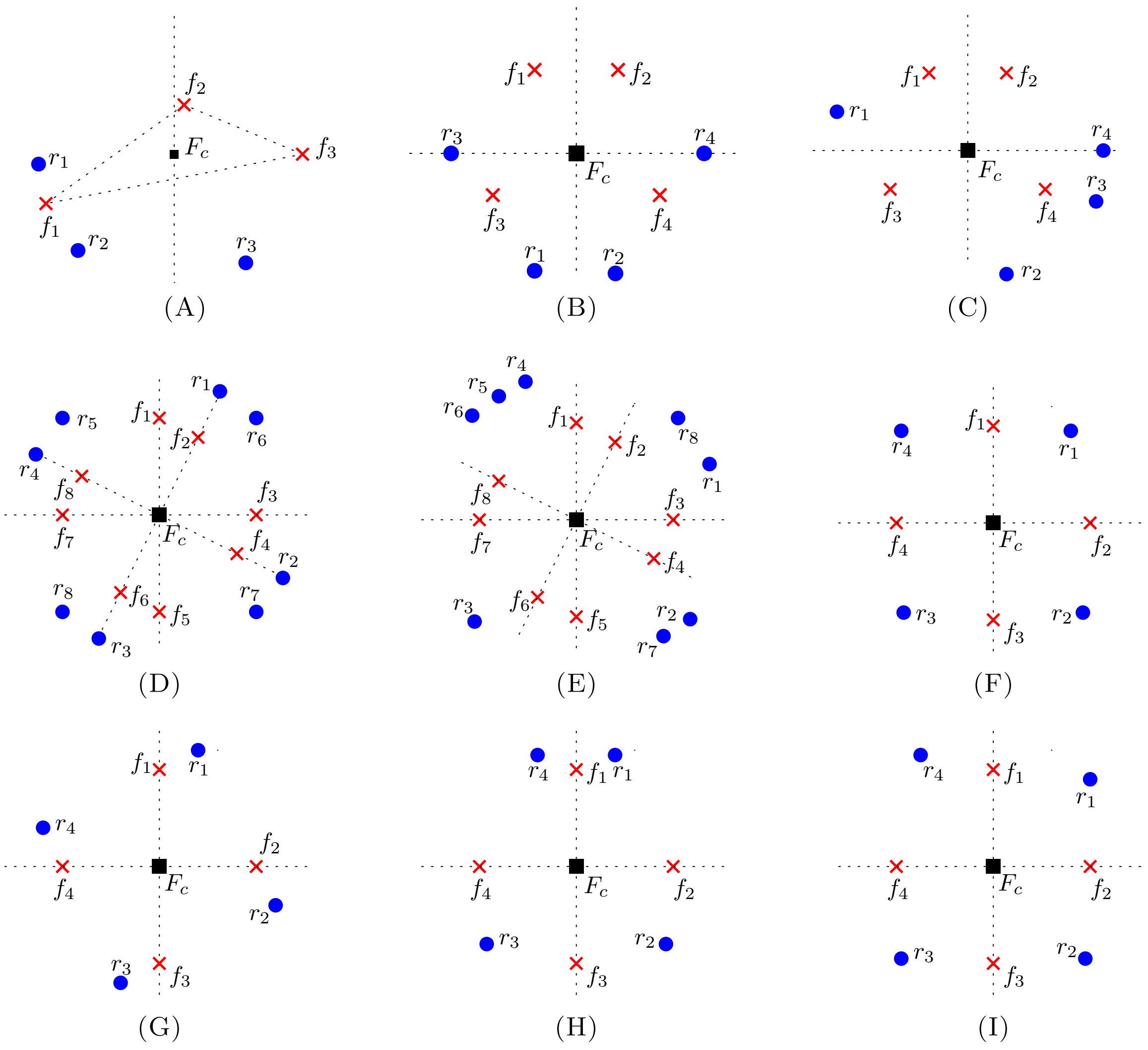


FIGURE 4.2: **(A)** $C(t) \in \mathcal{F}ASYM$, **(B)**-**(C)** $C(t) \in \mathcal{F}REFL$, **(D)**-**(E)** $C(t) \in \mathcal{F}CHIR$, **(F)**-**(I)** $C(t) \in \mathcal{F}MULT$.

Automorphisms and ***orbits*** **[12]:** Given an automorphism $\phi \in Aut(C(t))$, the cyclic subgroup of order k generated by ϕ is given by $\{\phi^0, \phi^1 = \phi, \phi^2 = \phi \circ \phi, \ldots, \phi^{k-1}\}$ where ϕ^0 is the identity. For example, a reflection ϕ generates a cyclic subgroup $H = \{\phi^0, \phi\}$ of order two. If H is a cyclic subgroup of $Aut(C(t))$, the *orbit* of a point $p \in R(t) \cup F$ is given by $Hp = \{\phi(p) \mid \phi \in H\}$. Note that the *orbits* Hp, for each $p \in R(t) \cup F$ form a partition of $R(t) \cup F$. The associated equivalence relation is defined by saying that p and q are equivalent if and only if their *orbits* are the same, that is $Hp = Hq$. Equivalent robots are indistinguishable by any algorithm.

Symmetry of a Configuration [12]**:** A function $\phi : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$ is called an *isometry* or distance preserving map if for any $p, q \in \mathbb{R}^2$, $d(\phi(p), \phi(q)) = d(p, q)$. Examples of *isometries* in the plane are translations, rotations and reflections. An automorphism of $C(t)$ is an *isometry* from $\mathbb{R}^2$ to $\mathbb{R}^2$ that maps $R(t)$ to $R(t)$ and F to F. The set of all automorphisms of $C(t)$ forms a group with respect to the composition called automorphism group of $C(t)$ and it is denoted by $Aut(C(t))$. If $|Aut(C(t))| = 1$, then $C(t)$ is said to be asymmetric (Figures 4.2(A), 4.2(C), 4.2(E) and 4.2(I)). If $|Aut(C(t))| > 1$, then $C(t)$ is said to be symmetric, i.e., it admits either rotations (Figures 4.2(D), 4.2(F) and 4.2(G)) or reflections (Figures 4.2(B), 4.2(F) and 4.2(H)). Since $|F \cup R(t)|$ is finite, translations are not possible.

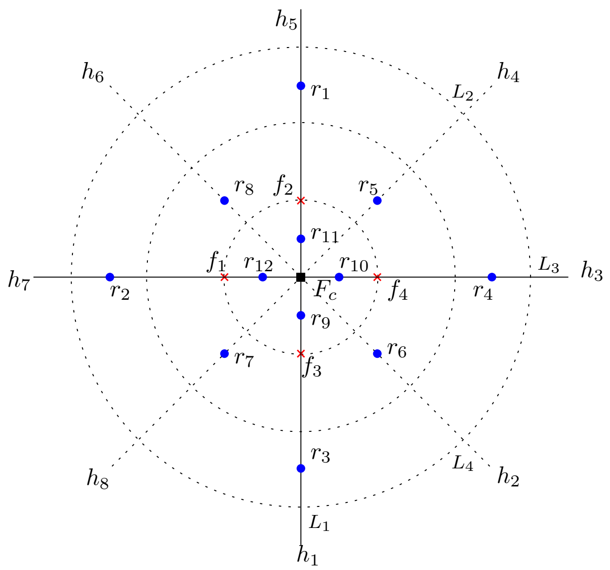


FIGURE 4.3: $\mathcal{L} = \{L_1, L_2, L_3, L_4\}$. $\mathcal{L}' = \{L_1, L_3\}$. $\mathcal{Z} = \{r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8\}$. $\mathcal{L}_R = \{r_1, r_2, r_3, r_4\}$.

4.2.2 Partitioning of the Configurations

All the configurations can be partitioned into the following disjoint classes:

1. $\mathcal{F}ASYM$ – F is asymmetric (Figure 4.2(A)).

2. $\mathcal{F}REFL - F$ has a single line of symmetry (Figure 4.2(B) and 4.2(C)).

3. $\mathcal{F}CHIR - F$ admits rotational symmetry without any line of symmetry (Figure 4.2(D) and 4.2(E)).

4. $\mathcal{F}MULT - F$ admits multiple lines of symmetry (Figure 4.2(F), 4.2(G), 4.2(H) and 4.2(I)).

Since the partition of the set of all the configurations depends only on F, the robots can easily identify the class to which a configuration belongs to without any conflicts. In Figure 4.2(B), $C(t)$ admits a single line of symmetry whereas $C(t)$ is asymmetric in Figure 4.2(C). $C(t)$ admits rotational symmetry without any line of symmetry (Figure 4.2(D)). $C(t)$ is asymmetric (Figure 4.2(E)). $C(t)$ admits multiple lines of symmetry (Figure 4.2(F)). $C(t)$ admits rotational symmetry without any line of symmetry (Figure 4.2(G)). $C(t)$ admits a single line of symmetry (Figure 4.2(H)). $C(t)$ is asymmetric (Figure 4.2(I)).

4.2.3 Additional Notations

Given a configuration $C(t)$, let $\mathcal{L}$ denote the set of all the lines of symmetry for F (Figure 4.3). Define $\mathcal{L}' = \{L_i \mid L_i \in \mathcal{L} \; and \; L_i \cap F \neq \emptyset\}$ (Figure 4.3). Let h_j denote a half-line starting from F_c and passing along some $L_i \in \mathcal{L}$. In case $|\mathcal{L}| > 0$, define

$$\mathcal{Z} = \{r \mid r \in h_j \text{ along some } L_i \in \mathcal{L} \text{ and } d(F_c, r) = \max_{r_i \in h_j} d(F_c, r_i)\} \; (Figure\ 4.3)$$

$\mathcal{D}$ denotes the radius of the minimum enclosing circle for $R(t) \setminus \mathcal{Z}$. Define

$$\mathcal{L}_R = \{r \mid r \in \mathcal{Z} \text{ and } d(F_c, r) = \max_{r_i \in \mathcal{Z}} d(F_c, r_i)\} \text{ (Figure 4.3)}$$

.

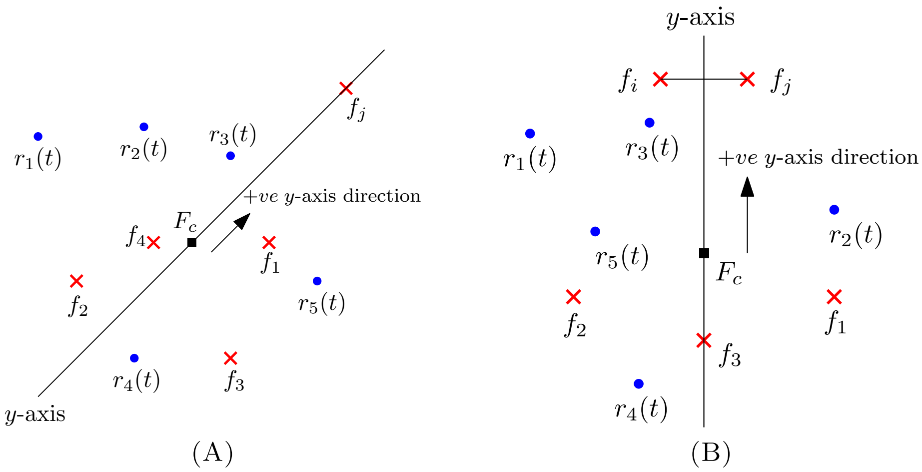


FIGURE 4.4: y-axis agreement. (**A**) $C(t) \in \mathcal{F}ASYM$ (**B**) $C(t) \in \mathcal{F}REFL$.

4.2.4 Global and Local Agreements

An active robot identifies the class of the current configuration and agrees on the following agreements accordingly:

1. $\boldsymbol{C(t) \in \mathcal{F}ASYM}$. Let f_j be the farthest fixed point from F_c. In case of a tie, choose the one having the minimum view. F_c is considered as the origin. The straight line passing through F_c and f_j is considered as the y-axis. The direction from F_c to f_j is considered as the positive y-axis direction (Figure 4.4(A)).

2. $\boldsymbol{C(t) \in \mathcal{F}REFL}$. Let L be the line of symmetry for F. The y-axis is assumed to pass along L. Consider all the symmetric pairs of fixed points, which are not collinear with F_c. Among all such pairs, choose the pair (say f_i and f_j), which is farthest from F_c. In the case of a tie, select the pair(s) closest to the y-axis. In case there are two such pairs, choose the one having the minimum view. F_c is considered as the origin. The direction from F_c towards $\overline{f_i f_j}$ is considered as the positive y-axis direction (Figure 4.4(B)).

3. $\boldsymbol{C(t) \in \mathcal{F}CHIR}$. $C(t)$ admits a rotation $\phi \in Aut(C(t))$. $C(t)$ satisfies Lemma 4.2.2 but does not satisfy Lemma 4.2.1. The cyclic subgroup generated by ϕ of order l is given by $H = \{\phi^0, \phi, \ldots, \phi^{l-1}\}$. Suppose Hf denotes the *orbit* of a fixed point $f \in F$ such that f has the minimum view. Since $C(t)$ does not admit any lines

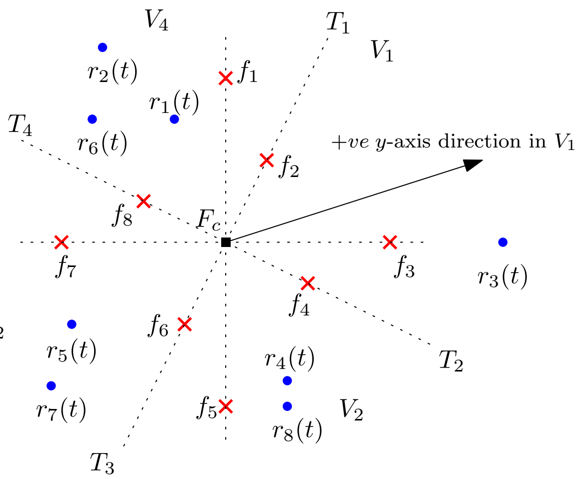


FIGURE 4.5: $C(t) \in \mathcal{F}CHIR$.

of symmetry, $\forall p, q \in R(t) \cup F$, not necessarily distinct, $\mathscr{V}^+(p) \neq \mathscr{V}^-(q)$. The direction of $\mathscr{V}(f)$ is globally considered as the clockwise direction. Let T_i be the half-line from F_c that passes through an $f_i \in Hf$. Each such half-line is considered as a wedge boundary. Let V_i denote the wedge in between T_i and T_{i+1}. Let $W_1 = \{V_1, V_2, \ldots, V_l\}$ for some $l > 0$ denotes the set of all wedges. Without loss of generality, assume that V_i is in the clockwise direction from T_i. The direction away from F_c and along the wedge bisector of V_i is considered as the positive y-axis direction in $V_i \cup T_i$ (Figure 4.5). The robots form an agreement on a common *chirality*.

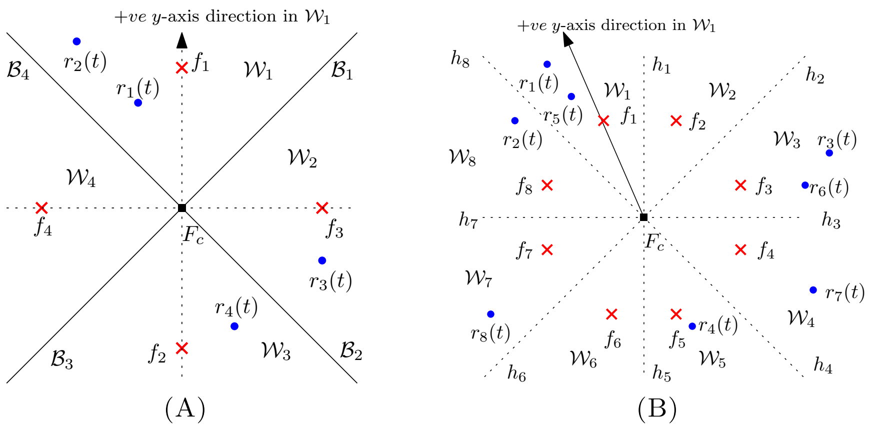


FIGURE 4.6: (**A**) $C(t) \in \mathcal{F}MULT$ and $\mathcal{L}' \neq \emptyset$ (**B**) $C(t) \in \mathcal{F}MULT$ and $\mathcal{L}' = \emptyset$.

4. $\boldsymbol{C(t) \in \mathcal{F}MULT}$. First, consider the case when $\mathcal{L}' \neq \emptyset$ (set of all the lines of symmetry for F containing fixed points). For each $L_i \in \mathcal{L}'$, consider the two half-lines, starting from F_c and along L_i. Suppose $H = \{h_1, h_2, \ldots, h_v\}$ denotes

the set of all such half-lines. Let $\mathcal{B}_i$ denote the angle bisector of $\measuredangle h_i F_c h_{i+1}$ where $h_i, h_{i+1} \in H$. Let $\mathcal{W}_i$ denote the wedge between $\mathcal{B}_{i-1}$ and $\mathcal{B}_i$ (Figure 4.6(A)). Next, consider the case when $\mathcal{L}' = \emptyset$. Each half-line along some $L_i \in \mathcal{L}$ is considered as a wedge boundary (Figure 4.6(B)). Let $W_2 = \{\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_p\}$ for some $p > 0$ denote the set of all wedges. The direction away from F_c and along the wedge bisector is considered as the positive y-axis direction in a wedge $\mathcal{W}_i \in W_2$. The robots do not have agreement on a common *chirality* in this case.

The robots agree upon two different sets of wedges, namely W_1 (if $C(t) \in \mathcal{F}CHIR$) and W_2 (if $C(t) \in \mathcal{F}MULT$). Note that, there are local y-axes one per each wedge. Since the definition of wedges is based on the partitioning of the configurations, the robots would identify a type of wedge without any conflict.

Definition 4.2.3. *A point p is said to be a virtual robot position at time t, if $\exists\, r_k \in R(t)$ such that p and r_k are symmetric about a line of symmetry $L \in \mathcal{L}'$.*

4.2.5 Problem Definition

$C(t)$ is said to be a *final* configuration if the following conditions hold:

i) Each robot position $r_i(t)$ is on a circle $C(f_j, \rho)$, for some $f_j \in F$,

ii) $\forall f_i \in F,\ D_i(t) = 0$ and $|C(f_i, \rho) \cap R(t)| = k$.

To solve the *k-circle formation* problem, starting from an *initial* configuration, the robots are required to reach and remain in a *final* configuration. The definition of ρ ensures that all the circles are disjoint in any *final* configuration.

4.3 Impossibility Result

All the *initial* configurations and values of k, for which the *k-circle formation* problem is deterministically unsolvable in this setting are characterized.

Theorem 4.3.1. *Let $C(0) \in \mathcal{F}REFL \cup \mathcal{F}MULT$ be such that there exists a line of symmetry for $R(0) \cup F$ (say L), and the following conditions hold:*

i) $L \cap F \neq \emptyset$.

ii) $L \cap R(0) = \emptyset$.

If k is an odd integer, then the k-circle formation problem is deterministically unsolvable.

Proof. Let $\mathcal{A}$ be a deterministic distributed algorithm that solves the *k-circle formation* problem, for some odd integer $k > 0$. Let the symmetric image of r with respect to L is denoted by $\phi(r)$. Consider the following setting:

(i) The scheduler is considered to be SSYNC. In addition, assume that both r and $\phi(r)$ are activated simultaneously.

(ii) All the robots are assumed to move with the same constant speed without any transient stops. Also, assume that both r and $\phi(r)$ would travel the same amount of distance.

The robots would run the same algorithm. According to Lemma 4.2.1, the robots r and $\phi(r)$ would have the same configuration view. Thus, their computed destination points and the paths for movement would be symmetric images with respect to L. Since the *initial* configuration was symmetric, the robots would not be able to deterministically break the symmetry under this setting. Let f be a fixed point on L. As the configuration would remain symmetric, all the distinct k robot positions on $C(f, \rho)$ must be symmetric about L. Since k is odd, $C(f, \rho)$ must contain a robot position on L. As $L \cap R(0) = \emptyset$, one of the robots must reach L. Since all the robots move in pairs, if a robot r moves to L, then $\phi(r)$ would move to the same point. As a result, a point of robot multiplicity will be created on L. The robots on a multiplicity point can not be separated deterministically. Hence, the *k-circle formation* problem is deterministically unsolvable. $\square$

Definition 4.3.2. *A configuration $C(t)$ for some $t \geq 0$ is said to be a solvable configuration, if it does not satisfy the unsolvability criterion stated in Theorem 4.3.1.*

4.4 Algorithm

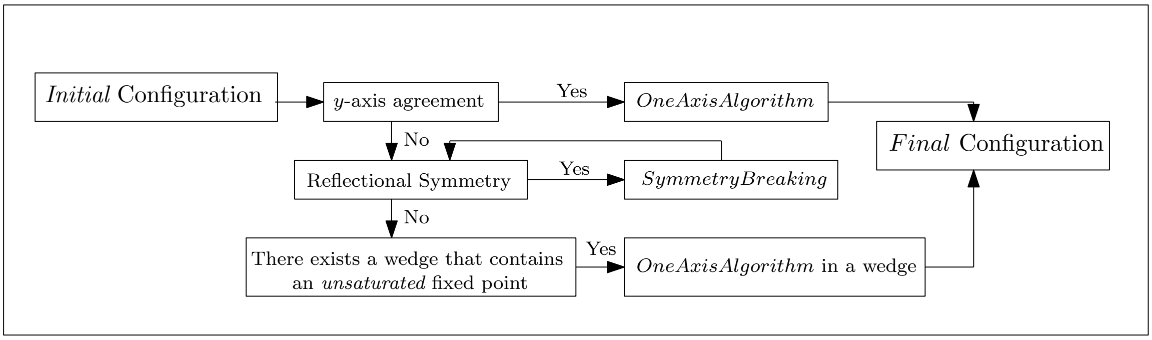


FIGURE 4.7: *AlgorithmNoAxis*

In this section, a deterministic distributed algorithm is proposed that solves the k-*circle formation* problem for completely disoriented robots. *AlgorithmNoAxis* would be discussed in details in subsection 4.4.3. Figure 4.7 shows a diagramatic representation of *AlgorithmNoAxis*. An overview of *AlgorithmNoAxis* is discussed as follows:

1. **The robots have a global y-axis agreement.** This includes the configurations in $\mathcal{F}ASYM \cup \mathcal{F}REFL$. The robots solve the k-*circle formation* problem by *AlgorithmOneAxis* discussed in Chapter 3.

2. **The robots do not have a global y-axis agreement.** They agree on the set of wedges W_1 or W_2. This includes the configurations in $\mathcal{F}CHIR \cup \mathcal{F}MULT$. In each such wedges, the robots make a local y-axis agreement. To break the reflectional symmetry about a line $L \in \mathcal{L}$, *SymmetryBreaking* (Subsection 4.4.1) is executed. The robots execute *AlgorithmOneAxis* locally in each wedge. However, the distribution of robot positions among the wedges may not be uniform. In such a case, the robots move from one wedge to another by the execution of *MovetoLine* (Subsection 4.4.2).

4.4.1 *SymmetryBreaking*

SymmetryBreaking is the procedure by which the robots would break the reflectional symmetry of a configuration $C(t) \in \mathcal{F}MULT$ for $t \geq 0$.

Definition 4.4.1. *Let h_j be a half-line along some $L \in \mathcal{L}$. Suppose h_j^+ denotes the half-line, that makes an angle $\frac{\alpha}{p}$ from h_j, measured in the clockwise direction from h_j, where p is the smallest positive integer for which there are no fixed points within $\frac{\alpha}{p}$ from h_j (excluding h_j). Similarly, assume that h_j^- denotes such a half-line in the counter-clockwise direction from h_j. Define $Region(h_j)$ as the closed region bounded by the half-lines h_j^+ and h_j^- (including h_j^+ and h_j^-) that contains h_j. Define $D_{j1} = d(F_c, r_i)$, where r_i is one of the farthest robot from F_c in $Region(h_j)$. Also, define $D_{j2} = d(F_c, r_k)$, where r_k is one of the second farthest robot from F_c in $Region(h_j)$.*

4.4.1.1 Phases during *SymmetryBreaking*

We define the following phases at any arbitrary point of time $t \geq 0$:

1. $\boldsymbol{P_1}$: $\exists L \in \mathcal{L}$ such that $C(t)$ is symmetric about L, $L \cap R(t) \neq \emptyset$ and $\exists r \in \mathcal{L}_R$ such that $d(F_c, r) < \mathcal{D} + 2$.

2. $\boldsymbol{P_2}$: $\exists L \in \mathcal{L}$ such that $C(t)$ is symmetric about L, $L \cap R(t) \neq \emptyset$ and $\forall r \in \mathcal{L}_R$ such that $d(F_c, r) \geq \mathcal{D} + 2$.

3. $\boldsymbol{P_3}$: $\exists r_i \in Region(h_j)$ for some h_j along some $L \in \mathcal{L}$ such that $D_{j1} - D_{j2} > 2$.

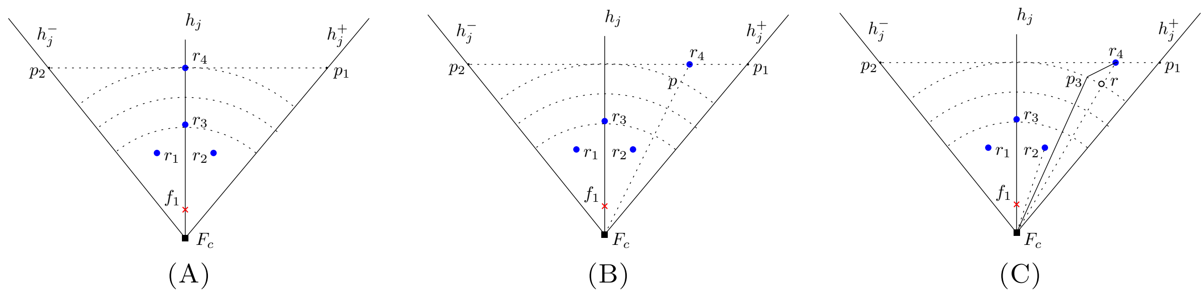


FIGURE 4.8: Empty circle represents a *virtual* robot position. (**A**) phase $\neg P_1 \wedge P_2$, (**B**)-(**C**) phase $\neg P_1 \wedge \neg P_2 \wedge P_3$.

4.4.1.2 Movements during *SymmetryBreaking*

Different types of movements during any execution of *SymmetryBreaking* are as follows:

1. $\boldsymbol{m_1}$: This movement is executed when the configuration is in phase $\boldsymbol{P_1}$. Suppose $r_i \in h_j$ such that $r_i \in \mathcal{L}_R$ and $d(F_c, r_i) < \mathcal{D}+2$. Let $p \in h_j$ such that $d(F_c, p) - \mathcal{D} = 2$. Robot r_i would be selected as a *candidate* robot. r_i moves along the half-line h_j towards p.

2. $\boldsymbol{m_2}$: This movement is executed when the configuration is in phase $\neg\boldsymbol{P_1} \wedge \boldsymbol{P_2}$. Suppose $r_i \in h_j$ such that $r_i \in \mathcal{L}_R$ and $d(F_c, r_i) \geq \mathcal{D} + 2$. Robot r_i would be selected as a *candidate* robot. Let T be the tangent to the circle $C(F_c, d(F_c, r_i))$ at $r_i(t)$. Suppose it intersects h_j^+ and h_j^- at the points p_1 and p_2, respectively. Robot r_i would select its destination point arbitrarily between p_1 and p_2 (Figure 4.8(A)). Without loss of generality, assume that p_1 is selected as the destination point. r_k moves towards p_1 along $\overline{r_k(t)p_1}$.

3. $\boldsymbol{m_3}$: This movement is executed when the configuration is in phase $\neg\boldsymbol{P_1} \wedge \neg\boldsymbol{P_2} \wedge \boldsymbol{P_3}$. Suppose $r_i \in Region(h_j)$ for some h_j along some $L \in \mathcal{L}$ such that $D_{j1} - D_{j2} > 2$. Let p be the point on $\overline{r_i(t)F_c}$ such that $d(F_c, p) - D_{j2} = 2$. Since r_i is the unique robot position in $Region(h_j)$ such that $D_{j1} - D_{j2} > 2$, there can not be any robot positions on $\overline{r_i(t)F_c}$. There are two cases:

   (*a*) $\overline{r_i(t)F_c}$ does not contain any *virtual* robot positions. Robot r_i would be selected as a *candidate* robot. r_i selects p as its destination point and it moves along $\overline{r_i(t)F_c}$ (Figure 4.8(B)).

   (*b*) $\overline{r_i(t)F_c}$ contains a *virtual* robot position. Let r_v be a robot or virtual robot position in $Region(h_j)$ such that $\measuredangle \overline{r_i(t)F_c} F_c \overline{r_v(t)F_c}$ is minimum and which does not lie on $\overline{r_i(t)F_c}$. Let B be the ray starting from F_c such that $\measuredangle \overline{r_i(t)F_c} F_c B = \frac{1}{2} min(\measuredangle \overline{r_i(t)F_c} F_c \overline{r_v(t)F_c}, \measuredangle \overline{r_i(t)F_c} F_c h_j)$. Suppose p_3 is the point on B such that $d(F_c, p_3) - D_{j2} = 2$. Robot r_i would be selected as a *candidate* robot. The *candidate* robot selects p_3 as its destination point and it moves along $\overline{r_i(t)p_3}$ (Figure 4.8(C)).

In Figure 4.8(A), r_4 lies on h_j and it would arbitrarily select its destination point between p_1 and p_2. In Figure 4.8(B), r_4 does not lie on h_j and it would select p as its destination point. In Figure 4.8(C), r_4 does not lie on h_j and p contains a virtual robot position r.

Let r_2 be the robot position in $Clear(h_j)$ such that the angle $\measuredangle \overline{F_c r_2} F_c \overline{F_c r_4}$ is minimum and $\measuredangle \overline{F_c p_3} F_c \overline{F_c r_4} = \frac{1}{2} \measuredangle \overline{F_c r_2} F_c \overline{F_c r_4}$. It would select p_3 as its destination point.

At time $t \geq 0$, if the configuration is in phase $P_1 \vee P_2 \vee P_3$, then any active robot will execute *SymmetryBreaking*. Execution of *Symmetrybreaking* is terminated when the configuration is in $\neg(P_1 \vee P_2 \vee P_3)$ (Figure 4.9). The detailed description of *SymmetryBreaking* is presented in the following table 4.1.

Phases	Movements	Phases after the Movements
P_1	m_1	P_1 or $\neg P_1 \wedge P_2$
$\neg P_1 \wedge P_2$	m_2	$\neg P_1 \wedge P_2$ or $\neg P_1 \wedge \neg P_2 \wedge P_3$
$\neg P_1 \wedge \neg P_2 \wedge P_3$	m_3	$\neg P_1 \wedge \neg P_2 \wedge P_3$ or $\neg(P_1 \vee P_2 \vee P_3)$

TABLE 4.1: Phase Transitions during *SymmetryBreaking*

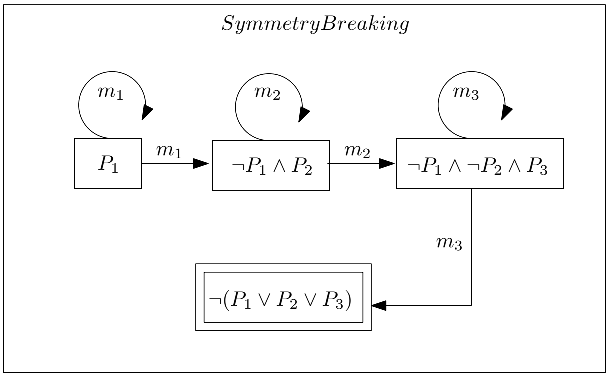


FIGURE 4.9: Phase transitions during *SymmetryBreaking*.

4.4.1.3 Progress during *SymmetryBreaking*

Lemma 4.4.2. *If the configuration $C(t)$ is in phase $P_1 \vee P_2 \vee P_3$, then by the execution of SymmetryBreaking the configuration would eventually be in phase $\neg(P_1 \vee P_2 \vee P_3)$.*

Proof. Let $t' > t$ be an arbitrary point of time at which r_i has completed at least one LCM cycle. We have the following cases:

Case 1. $C(t)$ is in $\boldsymbol{P_1}$. Let $L \in \mathcal{L}$ be such that it is a line of symmetry for $C(t)$ and $L \cap R(t) \neq \emptyset$. Let r_i be the farthest robot position on the half-line h_j along L. Since the configuration is in $\boldsymbol{P_1}$, $\exists r \in \mathcal{L}_R$ such that $d(F_c, r) < \mathcal{D} + 2$. Without loss of genrality, assume that $d(F_c, r_i) < \mathcal{D} + 2$. Robot r_i performs movement $\boldsymbol{m_1}$, i.e., it moves along

h_j to a point p, such that $d(F_c, p) - \mathcal{D} = 2$. Since r_i moves in a straight line towards p by at least δ amount, it will eventually reach p. Therefore, the configuration will be in $\neg \boldsymbol{P_1} \wedge \boldsymbol{P_2}$ within finite time.

Case 2. $C(t)$ is in $\neg \boldsymbol{P_1} \wedge \boldsymbol{P_2}$. Let $r_i(t) \in h_j$ be such that $d(F_c, r_i(t)) \geq \mathcal{D}+2$. Movement $\boldsymbol{m_2}$ will be performed by a *candidate* robot (say r_i). Since r_i would move by at least δ amount away from h_j, $r_i(t') \notin h_j$. Either $r_i(t') = p$ or $r_i(t') \neq p$. At t', $D_{j1} - D_{j2} > 2$ would be satisfied. Since $|\mathcal{L}_R|$ is finite within finite time the configuration will be in $\neg \boldsymbol{P_1} \wedge \neg \boldsymbol{P_2} \wedge \boldsymbol{P_3}$.

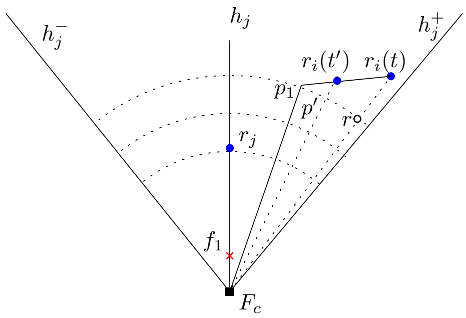


FIGURE 4.10: r_i selects p_1 as its destination point at time t. At t', r_i selects p' as its destination point.

Case 3. $C(t)$ is in $\neg \boldsymbol{P_1} \wedge \neg \boldsymbol{P_2} \wedge \boldsymbol{P_3}$. Let $r_i \in Region(h_j)$ be a *candidate* robot. Let p_1 be the destination point computed by r_i for performing movement $\boldsymbol{m_3}$. Let $d = d(r_i(t), p_1)$. At time t', let p' be the point on $\overline{r_i(t')F_c}$ such that $d(F_c, p) - D_{j2} = 2$. Since $\overline{r_i(t')F_c}$ would not contain any *virtual* robot positions (ensured by the selection of the destination point), r_i would select p' as its destination point and $\overline{r_i(t')F_c}$ as the path for movement (Figure 4.10). Since $d' = d(r_i(t'), p') < d(r_i(t'), p_1)$, $d - d' > d - d(r_i(t'), p_1) \geq \delta$. Thus, r_i would eventually reach a point such that $D_{j1} - D_{j2} = 2$ in $Region(h_j)$. Since $|\mathcal{L}|$ is finite there are only finite number of *candidate* robots. Therefore, within finite time the configuration will be in $\neg \boldsymbol{P_1} \wedge \neg \boldsymbol{P_2} \wedge \neg \boldsymbol{P_3}$.

Hence, if $C(t)$ is in phase $\boldsymbol{P_1} \vee \boldsymbol{P_2} \vee \boldsymbol{P_3}$, then by the execution of *SymmetryBreaking* the configuration would eventually be in phase $\neg(\boldsymbol{P_1} \vee \boldsymbol{P_2} \vee \boldsymbol{P_3})$. □

4.4.1.4 Solvability during *SymmetryBreaking*

In order to satisfy the unsolvability criterion (Theorem 4.3.1), k must be odd. We have the following observation.

Observation 1. *The k-circle formation problem is deterministically solvable for all the even values of k.*

Consider that $|F|$ is odd. If k is even, from observation 1 the *k-circle formation* problem is *solvable*. If k is odd, then the configuration would contain an odd number of robots. As a result, the configuration can not admit a line of symmetry without any robot positions on it. In order to satisfy the unsolvability criterion (Theorem 4.3.1), the line of symmetry should not contain any robot positions.

Observation 2. *All the configurations containing an odd number of fixed points are always solvable.*

To satisfy the unsolvability criterion (Theorem 4.3.1), the configuration must have a line of symmetry containing fixed points.

Observation 3. *If $\mathcal{L}' = \emptyset$, then the configuration would remain solvable.*

Lemma 4.4.3. *If $C(0) \in \mathcal{F}MULT$ and it is in $\boldsymbol{P_1} \vee \boldsymbol{P_2}$, then during any execution of SymmetryBreaking, $C(t)$ for some $t > 0$ would remain solvable.*

Proof. Let $L \in \mathcal{L}$ be such that it is a line of symmetry for $C(0)$ and $L \cap R(0) \neq \emptyset$. According to Observation 3, $C(t)$ would always remain *solvable* if $L \in \mathcal{L} \setminus \mathcal{L}'$. We assume that $L \in \mathcal{L}'$. Suppose h_i and h_j are the half-lines starting from F_c and passing along L. Assume that $r_a \in h_i$ and $r_b \in h_j$ are the farthest robots from F_c. Without loss of generality, assume that either $r_a \in \mathcal{L}_R$ or $r_b \in \mathcal{L}_R$. Otherwise, we can always select some $L' \in \mathcal{L} \setminus \{L\}$ can be selected. We have the following cases:

Case 1. $C(0)$ is in $\boldsymbol{P_1}$. Movement $\boldsymbol{m_1}$ will be performed by the *candidate* robots. Since $L \cap R(t) \neq \emptyset$ is preserved during movement $\boldsymbol{m_1}$ along L, the configuration would remain *solvable*.

Case 2. $C(0)$ is in $\neg \boldsymbol{P_1} \wedge \boldsymbol{P_2}$. Movement $\boldsymbol{m_2}$ will be performed by the *candidate* robots. We must show the following points:

Subcase 1. $C(t)$ will become asymmetric about L. First, consider that either $r_a \in \mathcal{L}_R$ or $r_b \in \mathcal{L}_R$. Without loss of generality, assume that $r_a \in \mathcal{L}_R$. Robot r_a would perform movement $\boldsymbol{m}_2$. Since r_a would be the unique robot position in $Region(h_j)$ such that $\mathcal{D}_{j1} - \mathcal{D}_{j2} > 2$, $C(t)$ would remain asymmetric about L. Next, consider that $r_a \in \mathcal{L}_R$ and $r_b \in \mathcal{L}_R$. Both r_a and r_b would perform movement $\boldsymbol{m}_2$. The following two scenarios are possible:

1. The *candidate* robots have moved to the same half-plane delimited by L.

2. The *candidate* robots have moved to different half-planes delimited by L.

In both the above two scenarios, $C(t)$ will become asymmetric about L.

Subcase 2. $C(t)$ will become asymmetric about $L_b \in \mathcal{L}' \setminus \{L\}$ or symmetric about L_b with $L_b \cap R(t) \neq \emptyset$. If k is even, then $C(t)$ would always remain *solvable* (Observation 1). We assume that k is odd. If possible, let L_b become a line of symmetry for $C(t)$. Let r be the symmetric image of r_a about L_b. Similarly, r_b would also have a symmetric image about L_b. According to the definition of $\mathcal{D}$, r must be on some $L_c \in \mathcal{L}'$ at $t = 0$. Otherwise, it cannot be a symmetric image of r_a at time t. This is because r_a would have avoided the *virtual* position of r during its movement. Also, the definition of the set $\mathcal{L}_R$ implies that $d(F_c, r(0)) = d(F_c, r_a(0))$. So, r_a and r are symmetric images of each other about L_b in $C(0)$. This is true for all such robot positions which have performed movement $\boldsymbol{m_2}$ in the time interval $[0, t]$. The following two scenarios are possible:

1. $C(0)$ is symmetric about L_b. Since $C(0)$ is *solvable* and $L_b \in \mathcal{L}'$, we must have $L_b \cap R(0) \neq \emptyset$. If $C(t)$ is symmetric about L_b, then we must have $\cap R(t) \neq \emptyset$. Otherwise, from subcase 1 of case 2 $C(t)$ is guaranteed to become asymmetric about L_b.

2. $C(0)$ is asymmetric about L_b. All the robots which have performed movement $\boldsymbol{m_2}$ would avoid all the virtual robot positions and other robot positions during their movement. Therefore, $C(t)$ would remain asymmetric about L_b.

Hence, if $C(0) \in \mathcal{F}MULT$ be such that it is in $\boldsymbol{P_1} \vee \boldsymbol{P_2}$, then during any execution of *SymmetryBreaking*, $C(t)$ for some $t > 0$ would remain *solvable*. □

Lemma 4.4.4. *If $C(0) \in \mathcal{F}MULT$ be such that it is in $\boldsymbol{P_3}$, then during any execution of SymmetryBreaking, $C(t)$ for some $t > 0$ would remain solvable.*

Proof. Let r_i be a *candidate* robot such that $r_i \in Region(h_j)$, where h_j is a half-line along some $L_i \in \mathcal{L}$ and $D_{j1} - D_{j2} > 2$. Robot r_i would perform movement $\boldsymbol{m_3}$. According to Observation 3, $C(t)$ would always remain *solvable* if $L_i \in \mathcal{L} \setminus \mathcal{L}'$. We assume that $L_i \in \mathcal{L}'$. During any execution of *SymmetryBreaking*, r_i would remain the unique robot in $Region(h_j)$ such that $D_{j1} - D_{j2} > 2$. As a result, $C(t)$ would remain asymmetric about L_i. Let h_a be a half-line along some $L_b \in \mathcal{L}' \setminus \{L_i\}$. Consider the following cases:

Case 1. $\exists\, r_j \in Region(h_a)$ such that $D_{a1} - D_{a2} \geq 2$. Since r_j would remain the unique robot in $Region(h_a)$, the configuration would remain asymmetric about L_b.

Case 2. $\forall r \in Region(h_a)$, $D_{a1} - D_{a2} < 2$. Let h_b be a half-line along some $L_p \in \mathcal{L}'$ such that $Region(h_b)$ and $Region(h_j)$ are mirror images about L_b. Consider the following cases:

Subcase 1. $\forall\, r \in Region(h_b)$, $D_{b1} - D_{b2} < 2$. Since r_i would not have any symmetric image in $Region(h_b)$ about L_b, $C(t)$ would remain asymmetric about L_b.

Subcase 2. $\exists\, r \in Region(h_b)$ such that $D_{b1} - D_{b2} \geq 2$. If r_i and r were not symmetric images of each other about L_b in $C(0)$, then $C(t)$ is guaranteed to be asymmetric about L_b. This is because each robot would avoid the *virtual* robot positions during its movement $\boldsymbol{m_3}$. Next, consider that r_i and r were symmetric images of each other in $C(0)$. Since $C(0)$ was solvable either the *initial* configuration was asymmetric about L_b or symmetric about L_b with $L_b \cap R(0) \neq \emptyset$. First, consider that was asymmetric about L_b. Since all the *candidate* robots have performed their movement $\boldsymbol{m_3}$ by avoiding *virtual* robot positions, $C(t)$ would remain asymmetric about L_b. Next, consider that $C(0)$ was symmetric about L_b with $L_b \cap R(0) \neq \emptyset$. Either $L_b \cap R(t) \neq \emptyset$ or $C(t)$ is asymmetric about L_b (follows from Lemma 4.4.3). $C(t)$ would remain *solvable*.

Hence, if $C(0) \in \mathcal{F}MULT$ be such that it is in $\boldsymbol{P_3}$, then during any execution of *SymmetryBreaking*, $C(t)$ for some $t > 0$ would remain *solvable*. □

4.4.2 *MovetoLine*

Before we discuss the procedure *MovetoLine*, we consider the following definitions.

Definition 4.4.5. *Consider a wedge $\mathcal{W}_i$ with the wedge boundaries $\mathcal{B}_i$ and $\mathcal{B}_{i+1}$. Suppose $\mathcal{W}_{i+1}$ be the adjacent wedge of $\mathcal{W}_i$, which is separated by the wedge boundary $\mathcal{B}_{i+1}$. Similarly, $\mathcal{W}_{i-1}$ denotes the adjacent wedge of $\mathcal{W}_i$, which is separated by the wedge boundary $\mathcal{B}_i$. Let M_1 be the half-line from F_c, that lies in the adjacent wedge $\mathcal{W}_{i+1}$, such that it makes an angle $\frac{\alpha}{p}$ from $\mathcal{B}_{i+1}$, where p is the smallest positive integer, for which there are no fixed points within angle $\frac{\alpha}{p}$ from $\mathcal{B}_{i+1}$. Similarly, define M_2, that lies in the adjacent wedge $\mathcal{W}_{i-1}$. Let $Area(\mathcal{W}_i)$ denote the open region (excluding M_1 and M_2 but including F_c) bounded by the half-lines M_1 and M_2, that includes $\mathcal{W}_i$.*

Definition 4.4.6. *$V_i \in W_1$ is said to contain a surplus robot if the following conditions hold:*

(i) There are no unsaturated fixed points in $V_i \cup T_i$ and

(ii) there exists either an oversaturated fixed point in $V_i \cup T_i$ or a robot that has not reached any circle yet in $V_i \cup T_i$.

Definition 4.4.7. *$\mathcal{W}_i \in W_2$ is said to contain a surplus robot if the following conditions hold:*

(i) There are no unsaturated fixed points in $Area(\mathcal{W}_i)$ and

(ii) there exists either an oversaturated fixed point in $Area(\mathcal{W}_i)$ or a robot that has not reached any circle yet in $Area(\mathcal{W}_i)$.

MovetoLine is the procedure by which a *surplus* robot moves towards a wedge in which there exists an *unsaturated* fixed point.

4.4.2.1 Phases during *MovetoLine*

We define the following phases at any arbitrary point of time $t \geq 0$:

1. $\boldsymbol{P_4}$: $C(t) \in \mathcal{F}CHIR$ and $\exists\, V_i \in W_1$ such that V_i contains a *surplus* robot.

2. $\boldsymbol{P_5}$: $C(t) \in \mathcal{F}MULT$ and $\exists\, \mathcal{W}_i \in W_2$ such that $\mathcal{W}_i$ contains a *surplus* robot.

4.4.2.2 Candidate Robot and its Destination Line

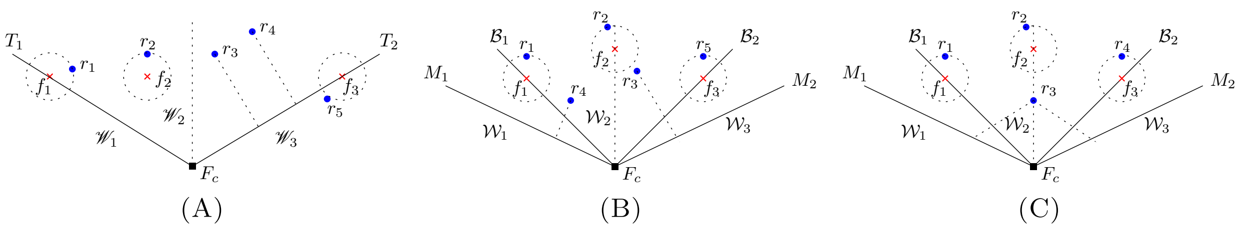


FIGURE 4.11: Selection of *destination* lines and *candidate* robots.

We have the following cases at any arbitrary point of time $t \geq 0$:

1. **$\boldsymbol{C(t)}$ is in phase $\boldsymbol{P_4}$.** Let $V_i \in W_1$ be such that V_i contains a *surplus* robot. The wedge boundary T_{i+1} that lies in the clock-wise direction from V_i is selected as the *destination line* (Figure 4.11(A)). Let r_j be the surplus robot that lies at a closest distance from T_{i+1}. In case, there are multiple such robots, choose the one that is farthest from F_c. Robot r_j is selected as the *candidate* robot (Figure 4.11(A)).

2. **$\boldsymbol{C(t)}$ is in phase $\boldsymbol{P_5}$.** Let $\mathcal{W}_i \in W_2$ be such that $\mathcal{W}_i$ contains a *surplus* robot. Consider the folowing definition:

   Definition 4.4.8. *The wedge boundary $\mathcal{B}_i$ is said to be closer to $\mathcal{W}_j$ than $\mathcal{W}_k$ if the number of wedges between $\mathcal{B}_i$ and $\mathcal{W}_j$ is less than the number of wedges between $\mathcal{B}_i$ and $\mathcal{W}_k$.*

   We have the following cases:

   (a) **Both the adjacent wedges of $\boldsymbol{\mathcal{W}_i}$ do not contain any** *unsaturated* **fixed points.** Without loss of generality, assume that between the two wedge boundaries, $\mathcal{B}_{i+1}$ is the wedge boundary that is closest to a wedge, that contains an *unsaturated* fixed point. M_1 is selected as the *destination line*. Let r_i be the *surplus* robot that lies at the closest distance from M_1. If there are multiple such robots, choose the one that is farthest from F_c. Robot r_i is selected as the *candidate* robot. Next, consider the case when both $\mathcal{B}_i$ and $\mathcal{B}_{i+1}$ are respectively closest to some wedge, which contains an *unsaturated* fixed point. In this case, both M_1 and M_2 are selected as a *destination line*. For each *destination line*, a *candidate* robot will be selected, similar to the above case. In this case, there may be two *candidate* robots in $\mathcal{W}_i$ (Figure 4.11(B)). If a robot lies at an equal distance from both the *destination lines*, then the robot would arbitrarily select its *destination line* (Figure 4.11(C)).

(b) **One of the adjacent wedges of $\mathcal{W}_i$ contains an *unsaturated* fixed point.** Without loss of generality, assume that $\mathcal{W}_{i+1}$ is the wedge that contains an *unsaturated* fixed point. M_1 is selected as the *destination line*. Next, a *candidate* robot will be selected for M_1 similarly to the above case.

Suppose $k = 1$. In Figure 4.11(A), r_3 and r_4 are the *surplus* robots in $V_2 \cup T_2$. Assume that V_3 lies in the clockwise direction from V_2. Wedge boundary T_2 is selected as the destination line. Both r_3 and r_4 are at equal distance from T_2. Since $d(c, r_4) > d(c, r_3)$, r_4 is selected as the *candidate* robot. In Figure 4.11(B), r_3 and r_4 are *surplus* robots in $Sur(\mathcal{W}_i)$. Without loss of generality, assume that both the adjacent wedges of $\mathcal{W}_2$ do not contain any *unsaturated* fixed points. In addition, assume that both $\mathcal{B}_1$ and $\mathcal{B}_2$ are individually closest to some wedge which contains an *unsaturated* fixed point. Both M_1 and M_2 are selected as *destination lines*. Both r_3 and r_4 are selected as *candidate* robots. In Figure 4.11(C), r_3 is the only *surplus* robot in $Sur(\mathcal{W}_i)$. Since it is at equidistant from both M_1 and M_2, it selects its *destination line* arbitrarily.

Let r_i be a *candidate* robot and L its *destination line* during $MovetoLine$. Let $x \in L$ be the point such that $\overline{r_i x}$ is perpendicular to L.

4.4.2.3 Conditions during $MovetoLine$

We define the following conditions during $MovetoLine$:

1. $\boldsymbol{c_1}$: $\overline{r_i x}$ passes through a *saturated* circle.
2. $\boldsymbol{c_2}$: $\overline{r_i x}$ does not pass through any *saturated* circle but there exists a robot position or a *virtual* robot position on $\overline{r_i x}$ (line segment excluding x).

3. $\boldsymbol{c_3}$: $\overline{r_i x}$ neither passes through any *saturated* circle nor contains any robot positions nor any *virtual* robot positions on $\overline{r_i x}$ (line segment excluding x).

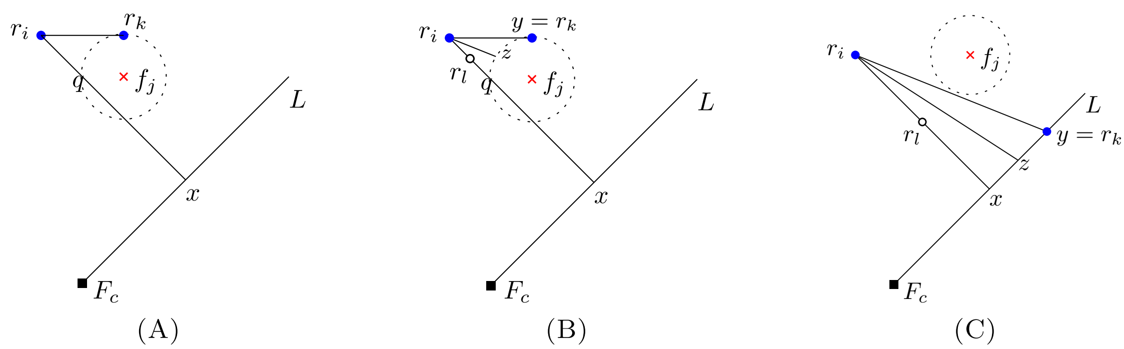


FIGURE 4.12: Movements (**A**)-(**B**)m_4, (**C**) m_5

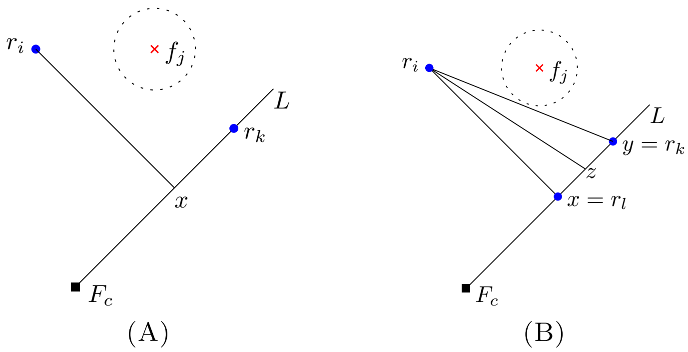


FIGURE 4.13: (**A**)-(**B**) Movement m_6.

4.4.2.4 Movements during *MovetoLine*

Different types of movements are as follows:

1. $\boldsymbol{m_4}$ **:** This movement is executed when r_i satisfies $\boldsymbol{c_1}$. Let $C(f_j, \rho)$ be the first circle to which r_i intersects while moving along $\overline{r_i x}$ towards x. Suppose, q is the intersection point between $\overline{r_i x}$ and $C(f_j, \rho)$, which is at closest distance from r_i. The *candidate* robot has the following cases:

   (i) There neither exists a *virtual* robot position on $\overline{r_i q}$ nor a robot position on q. The robot r_i moves along $\overline{r_i q}$, towards q (Figure 4.12(A)).

   (ii) Otherwise, let y be the closest robot position or the *virtual* robot position from q (such that q, y and x are not collinear), which lies on $C(f_j, \rho)$. Let z be the point on $C(f_j, \rho)$ such that $\measuredangle \overline{r_i q} r_i \overline{r_i z} = \frac{1}{2^p}(\measuredangle \overline{r_i q} r_i \overline{r_i y})$, where p is the smallest positive integer for which $\overline{r_i z}$ does not contain any *virtual* robot positions. Due to the choice of y and *candidate* robot r_i, $\overline{r_i z}$ possibly can not contain any robot positions. Robot r_i moves along $\overline{r_i z}$ towards z (Figure 4.12(B)).

2. $\boldsymbol{m_5}$ **:** This movement is executed when r_i satisfies $\boldsymbol{c_2}$. Let y be the closest robot position or the *virtual* robot position from x, which lies on L. In case there are no such robots, take $y = F_c$. Let z be the point on $\overline{xy}$ such that $\measuredangle \overline{r_i x} r_i \overline{r_i z} = \frac{1}{2^p}(\measuredangle \overline{r_i x} r_i \overline{r_i y})$, where p is the smallest positive integer for which $\overline{r_i z}$ does not contain any robot positions or any *virtual* robot positions or does not intersects any *saturated* circle. Robot r_i moves along $\overline{r_i z}$ towards z (Figure 4.12(C)).

3. $\boldsymbol{m_6}$ **:** This movement is executed when r_i satisfies $\boldsymbol{c_3}$. If x is not a robot position, then r_i moves along $\overline{r_i x}$ towards x (Figure 4.13(A)). Otherwise, the actions are similar to the case 2 for the *candidate* robot r_i (Figure 4.13(B)).

In Figure 4.12(A) $\overline{r_i q}$ does not contain any robot positions and *virtual* robot positions, r_i selects q as its destination point and moves along $\overline{r_i q}$. In Figure 4.12(B) $\overline{r_i q}$ contains a *virtual* robot position r_l, r_i selects z as its destination point and moves along $\overline{r_i z}$. In Figure 4.12(C) $\overline{r_i x}$ contains a *virtual* robot position r_l, r_i selects z as its destination point and moves along $\overline{r_i z}$. In Figure 4.13(A) $\overline{r_i x}$ does not contain any robot positions or virtual robot positions. Since x is neither a robot position nor a *virtual* robot position, r_i selects x as its destination point and moves along $\overline{r_i x}$. In Figure 4.13(B) Since x is a robot position, r_i selects z as its destination point and moves along $\overline{r_i z}$.

At time $t \geq 0$, if the configuration is in either phase P_4 or P_5, then any active robot will execute $MovetoLine$. Execution of $MovetoLine$ is terminated when the configuration is neither in P_4 nor in P_5 (Figure 4.14). The detailed description of $MovetoLine$ is presented in the following table 4.2.

4.4.2.5 Solvability during $MovetoLine$

Lemma 4.4.9. *If $C(0) \in \mathcal{F}MULT$ be such that it is in $\boldsymbol{P_4}$ or $\boldsymbol{P_5}$, then during any execution of $MovetoLine$, $C(t)$ at $t > 0$ would remain solvable.*

Phases	Conditions	Movements	Phases after the Movements
P_4	c_1	m_4	P_4 or $\neg P_4$
P_4	c_2	m_5	P_4 or $\neg P_4$
P_4	c_3	m_6	P_4 or $\neg P_4$
P_5	c_1	m_4	P_5 or $\neg P_5$
P_5	c_2	m_5	P_5 or $\neg P_5$
P_5	c_3	m_6	P_5 or $\neg P_5$

TABLE 4.2: Phase Transitions during $MovetoLine$

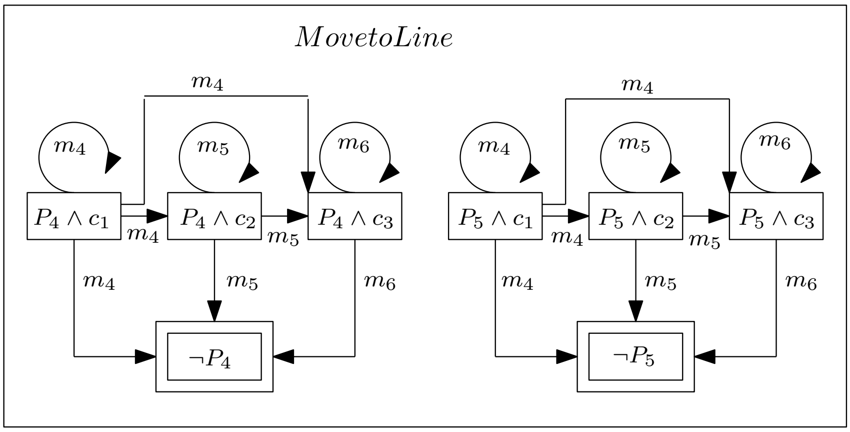


FIGURE 4.14: Phase transitions by a *candidate* robot during $MovetoLine$.

Proof. Let r_i be a *candidate* robot during an execution of $MovetoLine$. If $C(0)$ is asymmetric, execution of $MovetoLine$ would be started in a unique wedge. In case, $C(0)$ admits rotational symmetry, execution of $MovetoLine$ would be started in multiple wedges. A *candidate* robot would perform either $\boldsymbol{m_4}$ or $\boldsymbol{m_5}$ during any execution of $MovetoLine$. During such movements r_i would select its path for movement by ensuring that it does not contain any robot positions or *virtual* robot positions. Robot r_i would avoid all the points at which $C(t)$ might become symmetric about some $L_i \in \mathcal{L}'$. Therefore, $C(t)$ would remain asymmetric about each $L_i \in \mathcal{L}'$. Hence, if $C(0) \in \mathcal{F}MULT$ be such that it is in $\boldsymbol{P_4}$ or $\boldsymbol{P_5}$, then during any execution of $MovetoLine$, $C(t)$ at $t > 0$ would remain $solvable$. □

4.4.2.6 Progress during $MovetoLine$

Let $C(t)$ be in phase P_4 and $V_k \in W_1$ be a wedge that contains a *surplus* robot. In the wedge V_k, both the destination line (say L) and the *candidate* robot (say r_i) are unique. Let $q_i(t)$ be its destination point computed at time t. Let $Ray(r_i(t), q_i(t))$ denotes the ray starting from r_i and passing through $q_i(t)$. Suppose $p_1(t)$ denotes the point at which $Ray(r_i(t), q_i(t))$ intersects L. Define $g_i(t) = d(r_i(t), p_1(t))$. Let $x(t)$ be the point on L, such that $\overline{r_i x(t)}$ is perpendicular to L. Assume that there are $\mathcal{N}_1(t)$ number of *surplus* robots in V_k. Define $\mathcal{V}_k(t) = (\mathcal{N}_1(t), g_i(t))$.

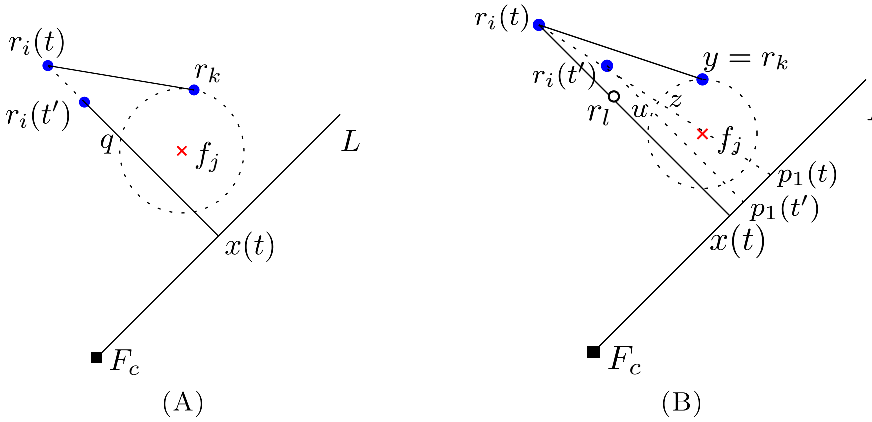


FIGURE 4.15: (**A**)-(**B**) Progress during movement m_4

In Figure 4.15(A), r_i selects q as its destination point at time t. It moves straight towards q and selects q as its destination point at time t'. In Figure 4.15(B), r_i selects z as its destination point at time t. At time t', r_i selects u as its destination point. In

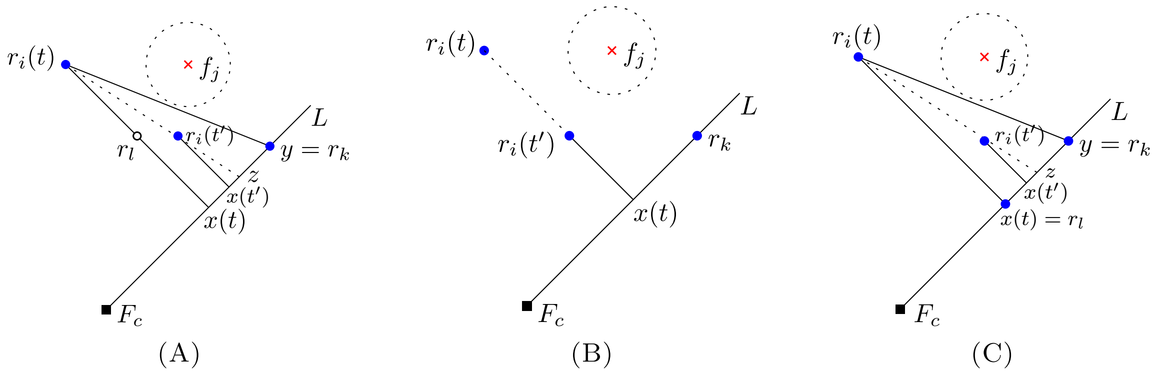


FIGURE 4.16: Progress during movements (**A**) m_5, (**B**)-(**C**) m_6.

Figure 4.16(A), $\overline{r_i(t)x(t)}$ contains a *virtual* robot position. r_i selects z as its destination point at time t. At time t', $\overline{r_i(t')x(t')}$ does not contain any robot positions or *virtual* robot positions. It selects $x(t')$ as its destination point. In Figure 4.16(B), r_i selects $x(t)$ as its destination point at time t. It moves straight towards $x(t)$ and selects $x(t)$ as its destination point at time t'. In Figure 4.16(C), $x(t)$ is a robot position. At time t, r_i

selects z as its destination point. At time t', $x(t')$ does not contain any robot positions. It selects $x(t')$ as its destination point.

Definition 4.4.10. *Let $C(t)$ be in phase P_4. During any execution of MovetoLine, we say that there has been progress in a wedge $V_k \in W_1$ in the time interval t to t' if $\mathcal{V}_k(t') < \mathcal{V}_k(t)$, i.e., $\mathcal{V}_k(t')$ is lexicographically smaller than $\mathcal{V}_k(t)$.*

Lemma 4.4.11. *Let $C(t)$ be in phase P_4. Let r_i be a candidate robot and $t' > t$ be an arbitrary point of time at which r_i has completed its at least one LCM cycle. Execution of MovetoLine ensures that $g_i(t') + \delta \leq g_i(t)$.*

Proof. The following cases are to be considered:

Case 1. $C(t)$ satisfies c_1. Let $C(f_j, \rho)$ be the first circle to which r_i intersects while moving along $\overline{r_i x}$ towards x. Suppose, q is the intersection point between $\overline{r_i x}$ and $C(f_j, \rho)$.

Subcase 1. $\overline{r_i q}$ does not contain any *virtual* robot positions and q does not contain any robot position. In this case, r_i executes case (i) of movement $\boldsymbol{m_4}$ (Figure 4.15(A)). Since r_i moves at least δ distance, $g_i(t') + \delta \leq g_i(t)$.

Subcase 2. Either q is a robot position or $\overline{r_i q}$ contains a *virtual* robot position. The *candidate* robot computes a destination point on $C(f_j, \rho)$ according to case (ii) of movement $\boldsymbol{m_4}$ (Figure 4.15(B)). At time t', $d(r_i(t'), p_1(t')) < d(r_i(t'), p_1(t))$ and $d(r_i(t), p_1(t)) - d(r_i(t'), p_1(t')) > d(r_i(t), p_1(t)) - d(r_i(t'), p_1(t)) \geq \delta$ (Figure 4.15(B)). This implies that $g_i(t') + \delta \leq g_i(t)$.

Case 2. $C(t)$ satisfies c_2. Robot r_i executes movement $\boldsymbol{m_5}$(Figure 4.16(A)). This case is similar to Subcase 2 of Case 1.

Case 3. $C(t)$ satisfies c_3. Robot r_i executes movement $\boldsymbol{m_6}$. If $x(t)$ is not a robot position, the case is similar to Subcase 1 of Case 1 (Figure 4.16(B)). In case $x(t)$ is a robot position, the case is similar to Subcase 2 of Case 1 (Figure 4.16(C)).

Hence, an execution of *MovetoLine* ensures that $g_i(t') + \delta \leq g_i(t)$. □

Lemma 4.4.12. *During an execution of MovetoLine in $V_k \in W_1$, let $t' > t$ be an arbitrary point of time at which a candidate robot r_i has completed its at least one LCM cycle. An execution of MovetoLine ensures progress in V_k in the time interval t to t'.*

Proof. The following cases are to be considered:

Case 1. $x(t) = r_i(t')$. *Then* $\mathcal{N}_1(t') = \mathcal{N}_1(t) - 1$, *which implies* $\mathcal{V}_2(t') < \mathcal{V}_2(t)$.

Case 2. $x(t) \neq r_i(t')$. *Lemma 4.4.11 ensures that* $g_i(t') < g_i(t)$. *Thus,* $\mathcal{V}_2(t') < \mathcal{V}_2(t)$.

Hence, an execution of *MovetoLine* ensures *progress in* V_k in the time interval t to t'. □

Next, assume that $C(t)$ is in $\boldsymbol{P_5}$. Let $\mathcal{W}_k \in W_2$ be a wedge that contains a *surplus* robot. *Progress in a wedge* $\mathcal{W}_k \in W_2$ can be defined similarly to the Definition 4.4.10. There are two possible cases: (i) single destination line and (ii) two destination lines. If there is a single destination line in $\mathcal{W}_k$, then there is a unique *candidate* robot. Thus, *progress in* $\mathcal{W}_k$ is ensured by (Lemma 4.4.12). If there are two destination lines, then there are two *candidate* robots (say r_1 and r_2) in $\mathcal{W}_k$. In this case, both the *candidate* robots must have different destination lines. Otherwise, both of them can not be selected as a *candidate* robot. As a result, each of them will continue their movements by ensuring *progress in* $\mathcal{W}_k$ (Lemma 4.4.12) without any conflict.

Lemma 4.4.13. *If* $C(t)$ *for* $t \geq 0$ *is in* $\boldsymbol{P_4}$ *or* $\boldsymbol{P_5}$, *then by the execution of MovetoLine within finite time the configuration would eventually be neither in* $\boldsymbol{P_4}$ *nor in* $\boldsymbol{P_5}$

Proof. Lemma 4.4.12 guarantees *progress in a wedge*. Since there is only a finite number of wedges, the number of wedges containing *surplus* robots is also finite. Therefore, within finite time the configuration would eventually be neither in $\boldsymbol{P_4}$ nor in $\boldsymbol{P_5}$ by the execution of *MovetoLine*. □

4.4.3 *AlgorithmNoAxis*

Definition 4.4.14. Let $\mathcal{W}_i$ and $\mathcal{W}_j$ be two wedges that contain a *surplus* robot. Let r_a and r_b be the *candidate* robots in $\mathcal{W}_i$ and $\mathcal{W}_j$, respectively. If there are two *candidate* robots in a wedge, then select the one that lies closest to its destination line. If there is a tie, select the one with the minimum view. If both the *candidate* robots have the same view, then select one of them arbitrarily. At time t, $\mathcal{W}_i$ is said to have more *progress* than $\mathcal{W}_j$ during *MovetoLine*, if $\mathcal{V}_i(t)$ is lexicographically smaller than $\mathcal{V}_j(t)$.

Definition 4.4.15. *Progress* **in a wedge $\boldsymbol{\mathcal{W}_i} \in \boldsymbol{W_2}$**: First, consider the case when there exists a unique *target* fixed point in each of the wedges. Let f_i and f_j be the *target* fixed points in $\mathcal{W}_i$ and $\mathcal{W}_j$, respectively, at time t. At time t, $\mathcal{W}_i$ is said to have more *progress* than $\mathcal{W}_j$, if one of the following holds:

(i) $\mathscr{V}(f_i) < \mathscr{V}(f_j)$, or

(ii) $\mathscr{V}(f_i) = \mathscr{V}(f_j)$ and $D_i(t) < D_j(t)$, or

(iii) $\mathscr{V}(f_i) = \mathscr{V}(f_j)$ and $D_i(t) = D_j(t)$ and $d(f_i, r_1) < d(f_j, r_2)$ where r_1 and r_2 are the *candidate* robots for f_i and f_j respectively at time t.

Next, consider the case when there are two *target* fixed points in the same wedge. This would happen when the wedge is symmetric about the wedge bisector. The two *target* fixed points will be separated by the wedge bisector, which is considered to be the y-axis in that wedge. If there has been the same *progress* in both the half-planes (Definition 3.4.1) delimited by the wedge bisector, then one of the *target* fixed points is considered arbitrarily. Otherwise, the *target* fixed point from the half-plane, for which there has been more *progress* is considered. Next, similar to the case of unique *target* fixed points, the robots can identify the wedge in which there has been more *progress*.

An active robot executes $AlgorithmNoAxis$ unless $C(t)$ is a *final* configuration. The pseudo-code for $AlgorithmNoAxis$ is given in Algorithm 4.1.

4.4.3.1 Phases during $AlgorithmNoAxis$

We define the following additional phases at time $t \geq 0$:

1. $\boldsymbol{P_6}$: $C(t)$ have y-axis agreement.
2. $\boldsymbol{P_7}$: $C(t) \in \mathcal{F}CHIR$ and $\exists f \in F$ such that f lies on F_c and f is *unsaturated*.
3. $\boldsymbol{P_8}$: $C(t) \in \mathcal{F}CHIR$ and $\exists V_i \in W_1$ such that V_i contains an *unsaturated* fixed point.
4. $\boldsymbol{P_9}$: $C(t) \in \mathcal{F}MULT$ and $\exists f \in F$ such that f lies on F_c and f is *unsaturated*.

ALGORITHM 4.1: *AlgorithmNoAxis*

Input: $C(t) = (R(t), F)$
if *the robots have y-axis agreement* **then**
  Execute $\boldsymbol{A_1}$ in $C(t)$;
else if *the robots have an agreement on a common chirality* **then**
  if $C(t)$ *is in phase* $\boldsymbol{P_7}$ **then**
    Let $f \in F$ such that f is on F_c;
    Execute $\boldsymbol{A_1}$ in $(R(t), \{f\})$;
  else if $C(t)$ *is in phase* $\neg\boldsymbol{P_7} \wedge \boldsymbol{P_4}$ **then**
    Let $V_i \in W_1$ be a wedge that contains a *surplus* robot;
    Execute $\boldsymbol{A_3}$ in the configuration consisting of fixed points and robots in $V_i \cup T_i$;
  else if $C(t)$ *is in phase* $\neg\boldsymbol{P_7} \wedge \neg\boldsymbol{P_4} \wedge \boldsymbol{P_8}$ **then**
    Let $V_i \in W_1$ contain an *unsaturated* fixed point;
    // If there are multiple such wedges select a wedge that contains the maximum number of robots
    Execute $\boldsymbol{A_1}$ in the configuration consisting of fixed points and robots in $V_i \cup T_i$;
  end
else
  if $C(t)$ *is in phase* $\boldsymbol{P_1} \vee \boldsymbol{P_2} \vee \boldsymbol{P_3}$ **then**
    Execute $\boldsymbol{A_2}$ in $C(t)$;
  else if $C(t)$ *is in phase* $\neg(\boldsymbol{P_1} \vee \boldsymbol{P_2} \vee \boldsymbol{P_3}) \wedge \boldsymbol{P_9}$ **then**
    Let $f \in F$ such that f is on F_c;
    Action $\boldsymbol{A_1}$ is executed in the configuration consisting of $R(t) \cup \{f\}$;
  else if $C(t)$ *is in phase* $\neg(\boldsymbol{P_1} \vee \boldsymbol{P_2} \vee \boldsymbol{P_3}) \wedge \neg\boldsymbol{P_9} \wedge \boldsymbol{P_5}$ **then**
    Let $\mathcal{W}_i \in W_2$ contain a *surplus* robot;
    // If there are multiple such wedges, select the wedge in which maximum *progress* during $MovetoLine$ is ensured. If there is a tie select the one that contains the robot with the minimum view
    Execute $\boldsymbol{A_3}$ in the configuration consisting of fixed points and robots in $Area(\mathcal{W}_i)$;
  else if $C(t)$ *is in phase* $\neg(\boldsymbol{P_1} \vee \boldsymbol{P_2} \vee \boldsymbol{P_3}) \wedge \neg\boldsymbol{P_9} \wedge \neg\boldsymbol{P_5} \wedge \boldsymbol{P_{10}}$ **then**
    Let $\mathcal{W}_i \in W_2$ be a wedge such that it does not contain any *unsaturated* fixed points, but $\exists\, f \in \mathcal{B}_{i-1} \cup \mathcal{B}_i$ such that it is *unsaturated*;
    // If there are more than one such wedges, then select the wedge in which maximum *progress* has been ensured. If there are multiple such wedges, then select the one that contains the robot with the minimum view
    Execute $\boldsymbol{A_1}$ in the configuration consisting of $R(t)$ and fixed points in $\mathcal{B}_{i-1} \cup \mathcal{B}_i$;
  else if $C(t)$ *is in phase* $\neg(\boldsymbol{P_1} \vee \boldsymbol{P_2} \vee \boldsymbol{P_3}) \wedge \neg\boldsymbol{P_9} \wedge \neg\boldsymbol{P_5} \wedge \neg\boldsymbol{P_{10}} \wedge \boldsymbol{P_{11}}$ **then**
    Let $\mathcal{W}_i \in W_2$ contain an *unsaturated* fixed point;
    // If there are multiple such wedges, select the one that contains the maximum number of robots. In case of a tie, select the one in which maximum *progress* has been ensured. If there are multiple such wedges, select the one that contains the robot with minimum view
    Execute $\boldsymbol{A_1}$ in the configuration consisting of robots and fixed points in $\mathcal{W}_i$;
  end
end

5. $\boldsymbol{P_{10}}$: $C(t) \in \mathcal{F}MULT$ and $\exists \mathcal{W}_i \in W_2$ such that it does not contain any *unsaturated* fixed points, but $\exists\, f \in \mathcal{B}_{i-1} \cup \mathcal{B}_i$ such that it is *unsaturated*.

6. $\boldsymbol{P_{11}}$: $C(t) \in \mathcal{F}MULT$ and $\exists \mathcal{W}_i \in W_2$ such that $\mathcal{W}_i$ contains an *unsaturated* fixed point.

4.4.3.2 Actions during *AlgorithmNoAxis*

1. $\boldsymbol{A_1}$: $AlgorithmOneAxis$

2. $\boldsymbol{A_2}$: $SymmetryBreaking$

3. $\boldsymbol{A_3}$: $MovetoLine$

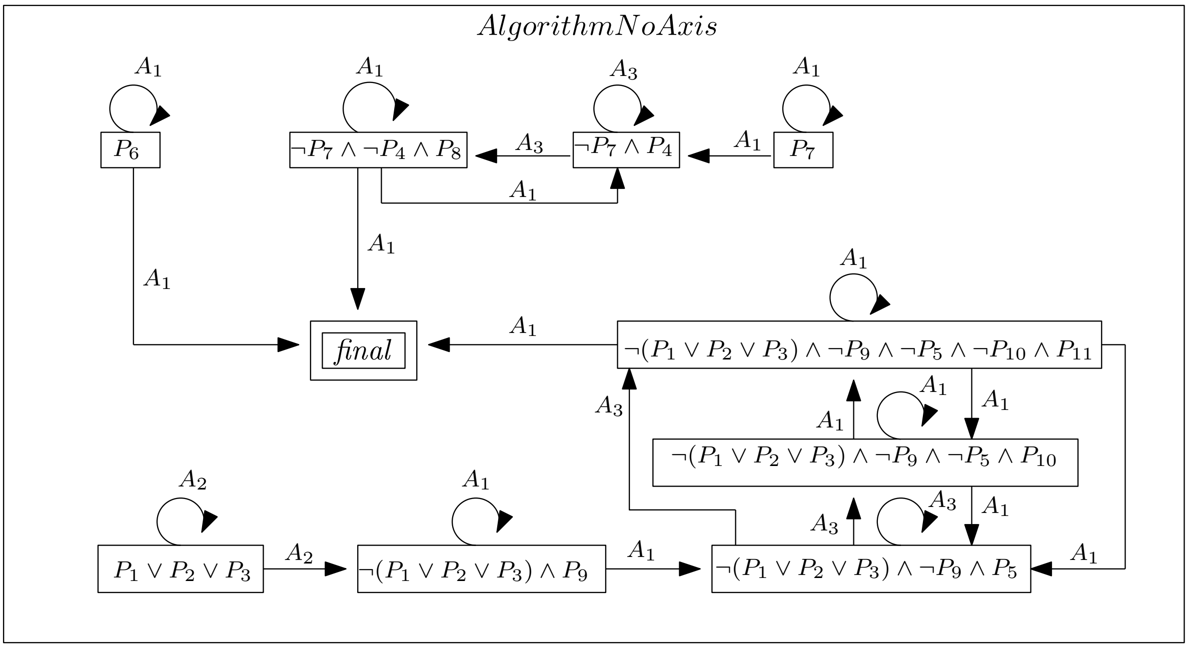


FIGURE 4.17: Phase transitions during $AlgorithmNoAxis$.

An active robot at time $t \geq 0$ considers the following cases during an execution of $AlgorithmNoAxis$:

1. **The robots have y-axis agreement, i.e., $C(t)$ is in phase $\boldsymbol{P_6}$.** $\boldsymbol{A_1}$ is executed.
2. **The robots have a common *chirality*.** The following cases are to be considered:
   (a) **$\boldsymbol{C(t)}$ is in phase $\boldsymbol{P_7}$.** Action $\boldsymbol{A_1}$ is executed in the configuration consisting of $R(t) \cup \{f\}$.
   (b) **$\boldsymbol{C(t)}$ is in phase $\boldsymbol{\neg P_7 \wedge P_4}$.** Action $\boldsymbol{A_3}$ is executed in the configuration consisting of robots and fixed points in $V_i \cup T_i$.
   (c) **$\boldsymbol{C(t)}$ is in phase $\boldsymbol{\neg P_7 \wedge \neg P_4 \wedge P_8}$.** Among all the wedges containing a *surplus* robot, let $V_i \in W_1$ be a wedge that contains the maximum number of robots. If there are more than one such wedges, then consider all such wedges. Action $\boldsymbol{A_1}$ is executed in the configuration consisting of robots and fixed points in $V_i \cup T_i$.
3. **The robots neither have one axis agreement nor agree on a common *chirality*.** The following cases are to be considered:
   (a) **$\boldsymbol{C(t)}$ is in phase $\boldsymbol{P_1 \vee P_2 \vee P_3}$.** Action $\boldsymbol{A_2}$ is executed in $C(t)$.

(b) $\boldsymbol{C(t)}$ **is in phase** $\boldsymbol{\neg(P_1 \vee P_2 \vee P_3) \wedge P_9}$. Action $\boldsymbol{A_1}$ is executed in the configuration consisting of $R(t) \cup \{f\}$.

(c) $\boldsymbol{C(t)}$ **is in phase** $\boldsymbol{\neg(P_1 \vee P_2 \vee P_3) \wedge \neg P_9 \wedge P_5}$. Let $\mathcal{W}_i \in W_2$ be a wedge that contains a *surplus* robot. If there are multiple such wedges, select the wedge in which maximum *progress* during $MovetoLine$ is ensured. If there is a tie select the one that contains the robot with the minimum view. Action $\boldsymbol{A_3}$ is executed for the configuration consisting of the robot positions, *virtual* robot positions and fixed points in $Area(\mathcal{W}_i)$.

(d) $\boldsymbol{C(t)}$ **is in phase** $\boldsymbol{\neg(P_1 \vee P_2 \vee P_3) \wedge \neg P_9 \wedge \neg P_5 \wedge P_{10}}$. Let $\mathcal{W}_i \in W_2$ be a wedge such that it does not contain any *unsaturated* fixed points, but $\exists\, f \in \mathcal{B}_{i-1} \cup \mathcal{B}_i$ such that it is *unsaturated*. If there are more than one such wedges, then select the wedge in which maximum *progress* has been ensured. If there are multiple such wedges, then select the one that contains the robot with the minimum view. Action $\boldsymbol{A_1}$ is executed for the configuration consisting of $R(t)$ and set of fixed points in $\mathcal{B}_{i-1} \cup \mathcal{B}_i$.

(e) $\boldsymbol{C(t)}$ **is in phase** $\boldsymbol{\neg(P_1 \vee P_2 \vee P_3) \wedge \neg P_9 \wedge \neg P_5 \wedge \neg P_{10} \wedge P_{11}}$. Let $\mathcal{W}_i \in W_2$ be a wedge that contains the maximum number of robot positions among all the wedges that contain a *unsaturated* fixed point. If there are multiple such wedges, select the wedge(s), in which maximum *progress* has been ensured. If there are more than one such wedge, the wedge containing the robot position with the minimum view is selected. Action $\boldsymbol{A_1}$ is executed for the configuration consisting of robot positions and fixed points in $\mathcal{W}_i$.

Figure 4.17 depicts the phase transitions during $AlgorithmNoAxis$. A summary of the $AlgorithmNoAxis$ is presented in the following table 4.3.

4.5 Correctness

To prove the correctness of algorithm $AlgorithmNoAxis$, the following points are shown:

1. *Solvability*: If the *initial* configuration is *solvable*, then during any execution of algorithm $AlgorithmNoAxis$, the configuration would remain *solvable*.

Classes	Phases	Actions	Phases after the Movements
$\mathcal{F}MULT$	$P_1 \vee P_2 \vee P_3$	A_2	$\neg(P_1 \vee P_2) \wedge P_3$ or $\neg(P_1 \vee P_2 \vee P_3)$
$\mathcal{F}MULT$	$\neg(P_1 \vee P_2 \vee P_3) \wedge P_9$	A_1	$\neg(P_1 \vee P_2 \vee P_3) \wedge P_9$ or $\neg(P_1 \vee P_2 \vee P_3) \wedge \neg P_9$
$\mathcal{F}MULT$	$\neg(P_1 \vee P_2 \vee P_3) \wedge \neg P_9 \wedge P_5$	A_3	$\neg(P_1 \vee P_2 \vee P_3) \wedge \neg P_9 \wedge P_5$ or $\neg(P_1 \vee P_2 \vee P_3) \wedge \neg P_9 \wedge \neg P_5 \wedge P_{10}$ or $\neg(P_1 \vee P_2 \vee P_3) \wedge \neg P_9 \wedge \neg P_5 \wedge \neg P_{10} \wedge P_{11}$
$\mathcal{F}MULT$	$\neg(P_1 \vee P_2 \vee P_3) \wedge \neg P_9 \wedge \neg P_5 \wedge P_{10}$	A_1	$\neg(P_1 \vee P_2 \vee P_3) \wedge \neg P_9 \wedge \neg P_5 \wedge P_{10}$ or $\neg(P_1 \vee P_2 \vee P_3) \wedge \neg P_9 \wedge \neg P_5 \wedge \neg P_{10} \wedge P_{11}$ or $\neg(P_1 \vee P_2 \vee P_3) \wedge \neg P_9 \wedge P_5$
$\mathcal{F}MULT$	$\neg(P_1 \vee P_2 \vee P_3) \wedge \neg P_9 \wedge \neg P_5 \wedge \neg P_{10} \wedge P_{11}$	A_1	$\neg(P_1 \vee P_2 \vee P_3) \wedge \neg P_9 \wedge \neg P_5 \wedge \neg P_{10} \wedge P_{11}$ or $\neg(P_1 \vee P_2 \vee P_3) \wedge \neg P_9 \wedge \neg P_5 \wedge \neg P_{10} \wedge \neg P_{11}$ or $\neg(P_1 \vee P_2 \vee P_3) \wedge \neg P_9 \wedge \neg P_5 \wedge P_{10}$ or $\neg(P_1 \vee P_2 \vee P_3) \wedge \neg P_9 \wedge P_5$
$\mathcal{F}CHIR$	P_7	A_1	P_7 or $\neg P_7$
$\mathcal{F}CHIR$	$\neg P_7 \wedge P_4$	A_3	$\neg P_7 \wedge P_4$ or $\neg P_7 \wedge \neg P_4$
$\mathcal{F}CHIR$	$\neg P_7 \wedge \neg P_4 \wedge P_8$	A_1	$\neg P_7 \wedge \neg P_4 \wedge P_8$ or $\neg P_7 \wedge \neg P_4 \wedge \neg P_8$ or $\neg P_7 \wedge P_4$
$\mathcal{F}ASYM \cup \mathcal{F}REFL$	P_6	A_1	P_6

TABLE 4.3: Transitions during *AlgorithmNoAxis*

2. *Progress*: During any execution of *AlgorithmNoAxis*, *progress* must be ensured, which would guarantee that the robots would solve the *k-circle formation* problem within finite time. In Figure 4.17, a phase transition implies *progress*. We must ensure that self-loops in a phase also ensures *progress*.

4.5.1 *Solvability*

Lemma 4.5.1. *If $C(0) \in \mathcal{F}ASYM \cup \mathcal{F}REFL$ and solvable, then the configuration $C(t)$ for $t \geq 0$ remains solvable during any execution of AlgorithmNoAxis.*

Proof. The robots have y-axis agreement. The proof follows from Theorem 3.5.10. □

Lemma 4.5.2. *If $C(0) \in \mathcal{F}CHIR$ and solvable, then the configuration $C(t)$ for $t \geq 0$ remains solvable during any execution of AlgorithmNoAxis.*

Proof. The robots have common *chirality*. Since F does not admit any line of symmetry, the configuration would never admit a line of symmetry. Therefore, the configuration $C(t)$ for $t \geq 0$ remains *solvable* during any execution of *AlgorithmNoAxis*. □

Lemma 4.5.3. *If $C(0) \in \mathcal{F}MULT$ and solvable, then the configuration $C(t)$ for $t \geq 0$ remains solvable during any execution of AlgorithmNoAxis.*

Proof. The following cases are to be considered:

Case 1. A_2 is executed. $C(t)$ is in phase $P_1 \vee P_2 \vee P_3$. From Lemmata 4.4.3 and 4.4.4, it follows that $C(t)$ would remain *solvable.*

Case 2. A_1 is executed. Consider the following subcases:

Subcase 1. $C(0)$ is asymmetric. Execution of A_1 will start in a unique wedge (say W). Let the wedges W and W' be mirror images about some $L_i \in \mathcal{L}'$. At time t, W is guaranteed to have more *progress* than W'. As a consequence, $C(t)$ will remain asymmetric about L_i. Since the choice of L_i was arbitrary, $C(t)$ would remain asymmetric about each $L_i \in \mathcal{L}'$.

Subcase 2. $C(0)$ admits rotational symmetry. Since $C(0)$ is in $\neg(P_1 \vee P_2 \vee P_3)$, if $C(0)$ is symmetric about a line $L \in \mathcal{L}$, then $L \in \mathcal{L} \setminus \mathcal{L}'$. A_1 would be executed in multiple wedges. Let W be such a wedge. Let W' be the mirror image of W about an $L_i \in \mathcal{L}'$. The following scenarios are possible:

1. Execution of A_1 has not started in W'. W is guaranteed to have more *progress* than W'. Thus, $C(t)$ would remain asymmetric about L_i.

2. Execution of A_1 has started in W'. Both W and W' had the same progress in $C(0)$. Also, both the wedges contain a robot with the minimum view. Let r_1 and r_2 be the robots with the minimum view in W and W', respectively. Thus, $\mathscr{V}(r_1) = \mathscr{V}(r_2)$. If $\mathscr{V}^+(r_1) = \mathscr{V}^-(r_2)$, then $C(0)$ was symmetric about L_i (Lemma 4.2.1). It contradicts that $C(0)$ was in $\neg(P_1 \vee P_2 \vee P_3)$. Therefore, $\mathscr{V}^+(r_1) = \mathscr{V}^+(r_2)$. From Lemma 4.2.2, it follows that the positive x-direction in both the wedges would be either in a clockwise or counter-clockwise direction (Figure 4.18). As a result, during the execution of A_1 in both the wedges, the half-plane with more *progress* in W cannot be symmetric to the half-plane with more *progress* in W'. Thus, $C(t)$ would remain asymmetric about each $L_i \in \mathcal{L}'$.

Case 3. A_3 is executed. $C(t)$ is in phase $\neg(P_1 \vee P_2 \vee P_3) \wedge \neg P_9 \wedge P_5$. From Lemma 4.4.9, it follows that $C(t)$ would remain *solvable.*

Hence, if $C(0) \in \mathcal{F}MULT$ and solvable, then the configuration $C(t)$ for $t \geq 0$ remains solvable during any execution of $AlgorithmNoAxis$. □

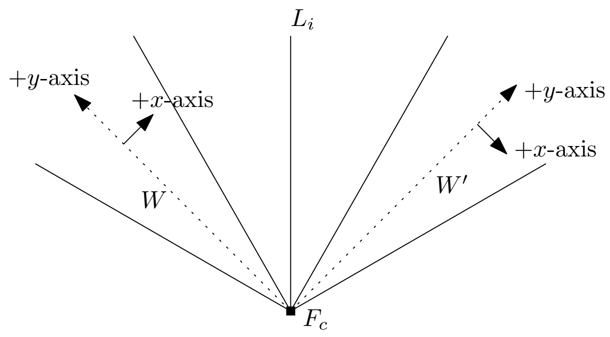


FIGURE 4.18: $C(t)$ is asymmetric about L_i.

4.5.2 *Progress*

Theorem 4.5.4. *If $C(0) \in \{\mathcal{F}ASYM \cup \mathcal{F}REFL \cup \mathcal{F}CHIR \cup \mathcal{F}MULT\}$ and it is solvable, then the robots would eventually solve the k-circle formation problem without any axis agreement, by the execution of $AlgorithmNoAxis$.*

Proof. If $C(t)$ is not a *final* configuration for $t \geq 0$, then the robots execute $AlgorithmNoAxis$. From Lemmata 4.5.1, 4.5.2 and 4.5.3, it follows that $C(t)$ would remain *solvable.* We have the following cases:

Case 1. The robots have y-axis agreement. $C(t)$ for $t \geq 0$ is in $\boldsymbol{P_6}$ and action $\boldsymbol{A_1}$ is executed. From Theorem 3.5.10, it follows that the robots would eventually solve the k-*circle formation* problem.

Case 2. The robots have a common *chirality*. The following cases are to be considered:

Subcase 1. $\boldsymbol{C(t)}$ is in phase $\boldsymbol{P_7}$. Action $\boldsymbol{A_1}$ is executed. Let $f = F_c$. From Theorem 3.5.10, it follows that f would eventually become *saturated.*

Subcase 2. $\boldsymbol{C(t)}$ is in phase $\boldsymbol{\neg P_7 \wedge P_4}$. Action $\boldsymbol{A_2}$ is executed. From the Lemma 4.4.13, it follows that the configuration would eventually satisfy $\boldsymbol{\neg P_7 \wedge \neg P_4}$.

Subcase 3. $\boldsymbol{C(t)}$ is in phase $\boldsymbol{\neg P_7 \wedge \neg P_4 \wedge P_8}$. Suppose $V_i \cup T_i$ for some $V_i \in W_1$ contains an *unsaturated* fixed point. Also, suppose $V_i \cup T_i$ contains the maximum number of robots. Action $\boldsymbol{A_1}$ is executed. Theorem 3.5.10 ensures that eventually the robots would solve the k-*circle formation* problem in $V_i \cup T_i$.

Since there can be only a finite number of wedges, the robots would solve the *k-circle formation* problem eventually.

Case 3. The robots neither have y-axis agreement nor have agreement on a common *chirality*. The following subcases are to be considered:

Subcase 1. $\boldsymbol{C(t)}$ is in phase $\boldsymbol{P_1 \vee P_2 \vee P_3}$. Action $\boldsymbol{A_2}$ is executed in $C(t)$. Lemma 4.4.2 guarantees that eventually $C(t)$ will satisfy $\boldsymbol{\neg(P_1 \vee P_2 \vee P_3)}$

Subcase 2. $\boldsymbol{C(t)}$ is in phase $\boldsymbol{\neg(P_1 \vee P_2 \vee P_3) \wedge P_9}$. Action $\boldsymbol{A_1}$ is executed in the configuration consisting of $R(t) \cup \{f\}$. Theorem 3.5.10 ensures that eventually f will become *saturated*.

Subcase 3. $\boldsymbol{C(t)}$ is in phase $\boldsymbol{\neg(P_1 \vee P_2 \vee P_3) \wedge \neg P_9 \wedge P_5}$. Action $\boldsymbol{A_3}$ is executed. From Lemma 4.4.13, it follows that the configuration would eventually satisfy the condition $\boldsymbol{\neg(P_1 \vee P_2 \vee P_3) \wedge \neg P_9 \wedge \neg P_5}$.

Subcase 4. $\boldsymbol{C(t)}$ is in phase $\boldsymbol{\neg(P_1 \vee P_2 \vee P_3) \wedge \neg P_9 \wedge \neg P_5 \wedge P_{10}}$. Suppose action $\boldsymbol{A_1}$ is executed for the configuration consisting of $R(t)$ and set of fixed points in $\mathcal{B}_{i-1} \cup \mathcal{B}_i$. Theorem 3.5.10 ensures that eventually all the fixed points in $\mathcal{B}_{i-1} \cup \mathcal{B}_i$ will become *saturated*.

Subcase 5. $\boldsymbol{C(t)}$ is in phase $\boldsymbol{\neg(P_1 \vee P_2 \vee P_3) \wedge \neg P_9 \wedge \neg P_5 \wedge \neg P_{10} \wedge P_{11}}$. Suppose action $\boldsymbol{A_1}$ is executed for the configuration consisting of robot positions and fixed points in $\mathcal{W}_i$. Theorem 3.5.10 ensures that eventually the robots would solve the *k-circle formation* problem in $\mathcal{W}_i$.

Since there can be only a finite number of wedges, the robots would solve the *k-circle formation* problem eventually.

Hence, if $C(0) \in \{\mathcal{F}ASYM \cup \mathcal{F}REFL \cup \mathcal{F}CHIR \cup \mathcal{F}MULT\}$ and it is *solvable*, then the robots would eventually solve the k-circle formation problem without any axis agreement, by the execution of $AlgorithmNoAxis$. □

4.6 Conclusions

This chapter investigates the *k-circle formation* problem by asynchronous, autonomous, anonymous and oblivious mobile robots in the *Euclidean* plane. The problem has been studied for a set of completely disoriented robots, i.e., they neither have any agreement on a global coordinate system nor on a common *chirality*. Since there can be multiple lines of symmetry, the set of unsolvable cases is larger than the set of unsolvable cases under one axis agreement. The following two results have been proved:

1. If $C(0)$ admits a line of symmetry (say L) such that $L \cap F \neq \emptyset$ and $L \cap R(0) = \emptyset$, then the k-circle formation problem is deterministically unsolvable without any axis agreement.
2. If $C(0) \in \{\mathcal{F}ASYM \cup \mathcal{F}REFL \cup \mathcal{F}CHIR \cup \mathcal{F}MULT\}$ and it is *solvable*, then the k-circle formation problem is deterministically solvable without any axis agreement.

Chapter 5

k-Circle Formation by Opaque Robots

Contents

5.1 Overview

In this chapter, the *k-circle formation* problem is investigated under an *obstructed* visibility model. The robots are assumed to be *opaque*, i.e., a robot cannot see another robot if a third robot is placed on the line segment joining them. The robots may not know the positions of all the robots. As a consequence, some of the robots have to decide their strategy based on their partial visibility. The proposed distributed algorithms discussed in the previous chapters (Chapters 3 and 4) would fail to solve the *k-circle formation* problem as both the algorithms are based on the assumption that the robots have *unlimited* visibility. The primary motivation is to investigate the solvability of the k-*circle formation* problem under *obstructed* visibility model.

The problem has been investigated in two different settings: complete knowledge of fixed points and zero knowledge of fixed points. If the robots are *oblivious* and *silent*, then to identify the termination condition (the robots have solved the k-*circle formation* problem) the robots should have knowledge of the positions of all other robots and fixed points. If the robots have complete knowledge of the fixed points, then they only need to solve the *mutual visibility* problem for robot positions. By solving the *mutual visibility* problem for robot positions, they will be able to identify whether the position of a robot is on a circle or not. If the robots have zero knowledge of the fixed points, then all the fixed points in addition to robot positions must be visible to the robots so as to identify that the robots are positioned on a circle. In this case, the robots need to solve both the k-*circle formation* problem and the *mutual visibility* problem. If the robots have zero knowledge of the fixed points, then the *oblivious* and *silent* robots may not be able to solve the *mutual visibility* problem. To solve the k-*circle formation* problem in this setting, the robots are assumed to be equipped with one bit of persistent memory.

5.2 The Model

The robots are represented by points in the Euclidean plane. They are assumed to be *autonomous*, *anonymous*, and *homogeneous*. The robots are assumed to be *opaque*, i.e., the view of a robot gets obstructed due to the presence of other robots. However, a fixed point cannot obstruct the view of a robot. The robots are assumed to be completely *disoriented*. They are assumed to be activated under a fair ASYNC scheduler. We have considered two different settings based upon the visibility of the fixed points:

1. Complete knowledge of fixed points. The fixed points are tower-like structures which are always visible to the robots. Thus, the positions of the fixed points are known to the robots. As a consequence, the robots have the knowledge of the total

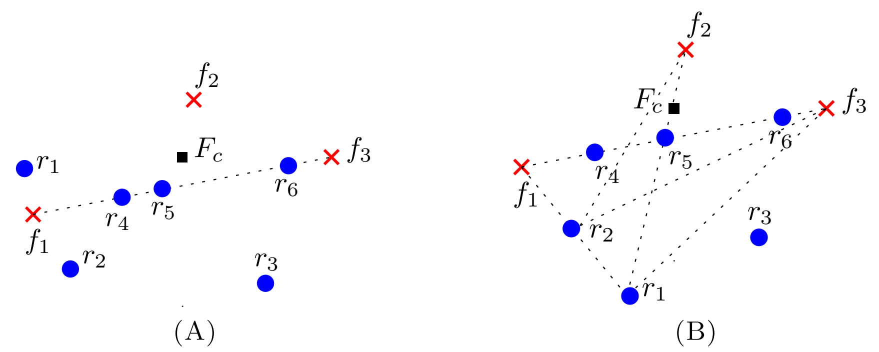


FIGURE 5.1: (**A**) Complete knowledge of the fixed points, (**B**) Zero knowledge of the fixed points.

number of fixed points. For example, in Figure 5.1(A), r_4 and r_6 cannot see each other but both of them can see the fixed points f_1, f_2 and f_3.

2. Zero knowledge of fixed points. A robot cannot see a fixed point if another robot is positioned on the line segment joining them. Thus, the positions of all the fixed points may be unknown to the robots. As a consequence, the robots do not have the knowledge of the total number of fixed points. For example, in the Figure 5.1(B), r_5 cannot see the fixed point f_3 due to presence of r_6 on $\overline{r_5 f_3}$. Similarly, r_5 cannot see the fixed point f_1 due to presence of r_4 on $\overline{r_5 f_1}$. $VFr_1(t) = \{f_3\}$, $VFr_2(t) = \{f_1, f_2, f_3\}$, $VFr_3(t) = \{f_1, f_2, f_3\}$, $VFr_4(t) = \{f_1, f_2\}$, $VFr_5(t) = \{f_2\}$, $VFr_6(t) = \{f_2, f_3\}$

For the first setting, we have assumed that the robots are *oblivious* and *silent*, i.e., they have no explicit direct communications. For the next setting, we consider the light model introduced by Peleg [47] where the robots are assumed to be equipped with a externally visible light that can assume a constant number of pre-defined colors. The color of the lights are persistent and serves as an explicit direct communication and as an internal memory. Note that a robot having light with only one color is equivalent to the one with no light. Therefore, the light model is a generalization of the classical model.

5.2.1 Notations and Definitions

(1) $VRr_i(t)$ and $VFr_i(t)$ denote the total number of visible robots and fixed points to the robot r_i at time t.

(2) $\mathcal{F}_i$ denotes the ray starting from F_c and passing through $f_i \in F$. Suppose $\mathcal{F} = \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_c\}$ represents the set of all such rays for some $c > 0$.

(3) $Ray(F_c, r_i)$ denotes the ray starting from F_c and passing through r_i. When $VFr_j(t) = m$, then r_j selects $\rho = \frac{1}{3}\min_{f_i, f_j \in F} d(f_i, f_j)$. ξ represents the radius of the minimum enclosing circle for F. Let $\mathscr{C} = C(F_c, p\xi)$ where $p > 0$ is smallest positive integer for which $\mathscr{C}$ is the minimum enclosing circle for $C(t)$ centered at F_c.

5.2.1.1 Convex Hull

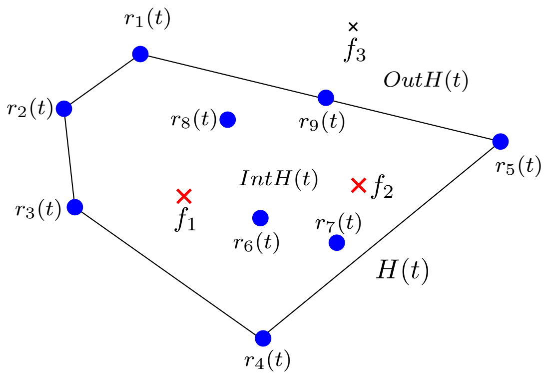


FIGURE 5.2: $r_6(t), r_7(t), r_8(t), f_1, f_2 \in IntH(t)$ and $f_3 \in OutH(t)$.

Given a set of points $S \subset \mathbb{R}^2$, a convex hull of S is the smallest convex set that contains S [119]. Let $H(t)$ denote the convex hull of $R(t)$ at time $t \geq 0$. Suppose $IntH(t)$ and $OutH(t)$ represents all the points in interior and exterior of $H(t)$, respectively, at $t \geq 0$ (Figure 5.2).

Definition 5.2.1. *A robot $r_i \in H(t)$ identifies itself to be on a boundary of $H(t)$ if $\exists r_j \in R$ such that $j \neq i$ and one of the open half-planes demarcated by the straight line passing along $\overline{r_j(t) r_i(t)}$ does not contain any other robots. Otherwise, r_i identifies itself to be in $IntH(t)$.*

Suppose r_i denotes a robot on a boundary of $H(t)$. Suppose r_j and r_k are the adjacent robots of r_i, also positioned on a boundary of $H(t)$. If $\measuredangle r_j r_i r_k < \pi$, then r_i identifies itself to be a vertex of $H(t)$. In case $\measuredangle r_j r_i r_k = \pi$, r_i identifies itself to be a non-vertex robot.

5.2.2 The k-Circle Formation Problem

At $t \geq 0$, $C(t)$ is said to be a *final* configuration, if it satisfies the following conditions:

i) $\forall r_i \in R$, $VRr_i(t) = n$, $VFr_i(t) = m$ and $r_i(t) \in C(f_j, \rho)$ $for\ some$ $f_j \in F$,

ii) $C(f_i, \rho) \cap C(f_j, \rho) = \emptyset$ for $f_i \neq f_j$, and

iii) $|C(f_i, \rho) \cap R(t)| = k$, $\forall f_i \in F$.

To solve the *k-circle formation* problem, starting from a given *initial* configuration the robots need to reach and remain in a *final* configuration.

5.2.3 Partitioning of the Configurations

If $VFr_i(t) = m$, a robot r_i can easily identify the class of a configuration by observing all the fixed points as discussed in section 4.2.2 of Chapter 4. All the configurations can be partitioned into the following disjoint classes:

1. $\mathcal{F}ASYM - F$ is asymmetric (Figure 4.2(A)).
2. $\mathcal{F}REFL - F$ has a single line of symmetry (Figure 4.2(B) and 4.2(C)).
3. $\mathcal{F}CHIR - F$ admits rotational symmetry without any line of symmetry (Figure 4.2(D) and 4.2(E)).
4. $\mathcal{F}MULT - F$ admits multiple lines of symmetry (Figure 4.2(F), 4.2(G), 4.2(H) and 4.2(I)).

5.3 Complete Knowledge of the Fixed Points

In this section, we consider the model where the robots have complete knowledge of the fixed points. We have $\forall t \geq 0$, $\forall r_i \in R$, $VFr_i(t) = m$.

5.3.1 Impossibility Result

Theorem 5.3.1. *Let $C(0) \in \mathcal{F}REFL \cup \mathcal{F}MULT$ be such that $\exists L \in \mathcal{L}'$, $C(0)$ is symmteric about L, and the following conditions hold:*

i) $L \cap F \neq \emptyset$.

ii) $L \cap R(0) = \emptyset$.

If k is an odd integer, then the k-circle formation problem is deterministically unsolvable by opaque disoriented robots.

Proof. The proof follows from Theorem 4.3.1. □

Let $\mathcal{U}_1$ denote the set of all the configurations which satisfy the conditions stated in Theorem 5.3.1.

5.3.2 Suitable Configurations

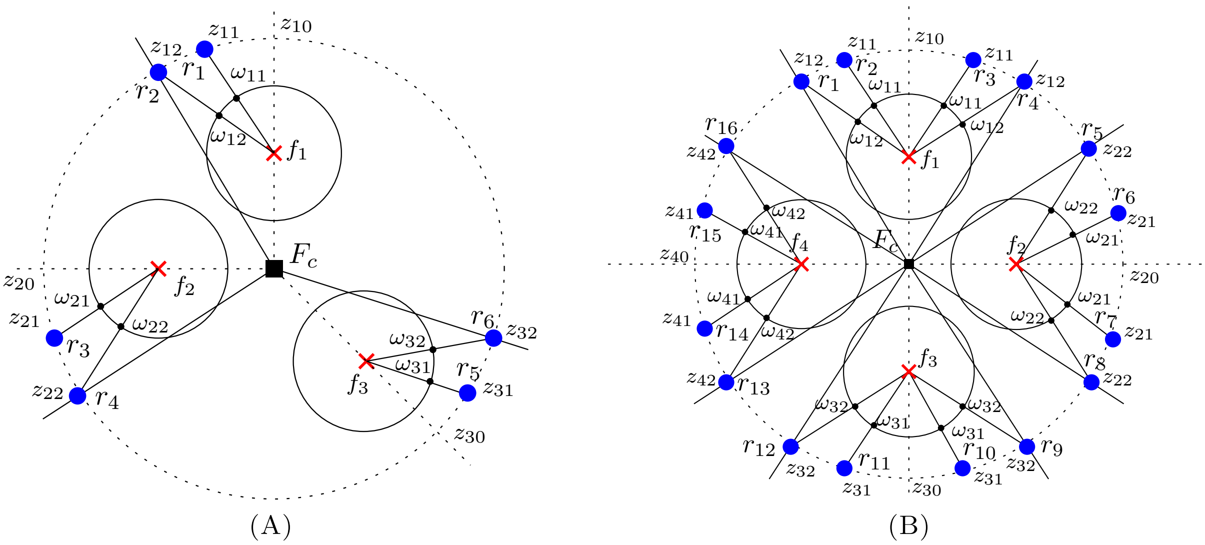


FIGURE 5.3: Examples of Suitable Configurations (**A**) $C(t) \in \mathcal{F}ASYM$, $k = 2$ and $m = 3$. (**B**) $C(t) \in \mathcal{F}REFL$, $k = 4$ and $m = 4$.

In this section, we discuss the construction of a *suitable* configuration. In a *suitable* configuration each robot (say r_i) is chosen for some fixed point (say f_j) and the position of r_i on $\mathscr{C}$ is selected by ensuring that r_i can directly move towards $C(f_j, \rho)$ along $\overline{r_i f_j}$. First, we introduce some new notations required for defining a *suitable configuration.*

1. $\alpha_m = \min\limits_{\mathcal{F}_i, \mathcal{F}_j \in \mathcal{F},\, \mathcal{F}_i \neq \mathcal{F}_j} \measuredangle \mathcal{F}_i F_c \mathcal{F}_j$.

2. β_i denotes the number of fixed points on $\mathcal{F}_i \in \mathcal{F}$. Suppose $\{f_1, f_2, \ldots f_{\beta_i}\}$ denotes the set of all the fixed points on $\mathcal{F}_i$ such that $d(F_c, f_1) > d(F_c, f_2) \ldots > d(F_c, f_{\beta_i})$.

3. $\forall \mathcal{F}_i \in \mathcal{F}$, z_{i0} denotes the the intersection point between $\mathscr{C}$ and $\mathcal{F}_i$. $z_{i(\beta_i k)}$ is defined to be the point on $\mathscr{C}$ such that $\measuredangle \overline{z_{i0} F_c} F_c \overline{z_{i(\beta_i k)} F_c} = \frac{1}{3}\alpha_m$. Note that there are two such $z_{i(\beta_i k)}$ points.

4. Let ω_{i0} and $\omega_{i(\beta_i k)}$ denote the intersection points between $C(f_{\beta_i}, \rho)$ and $\overline{F_c z_{i0}}$ and $\overline{F_c z_{i(\beta_i k)}}$, respectively.

Definition 5.3.2. *When $VFr_i(t) = m$, by considering only fixed points, the robot r_i can compute the configuration view as discussed in section 4.2.1. Let $f_i \in \mathcal{F}_m$ be a fixed point that has the highest configuration view. $\mathcal{F}_m$ is said to be a master ray. Note that there can be multiple master rays.*

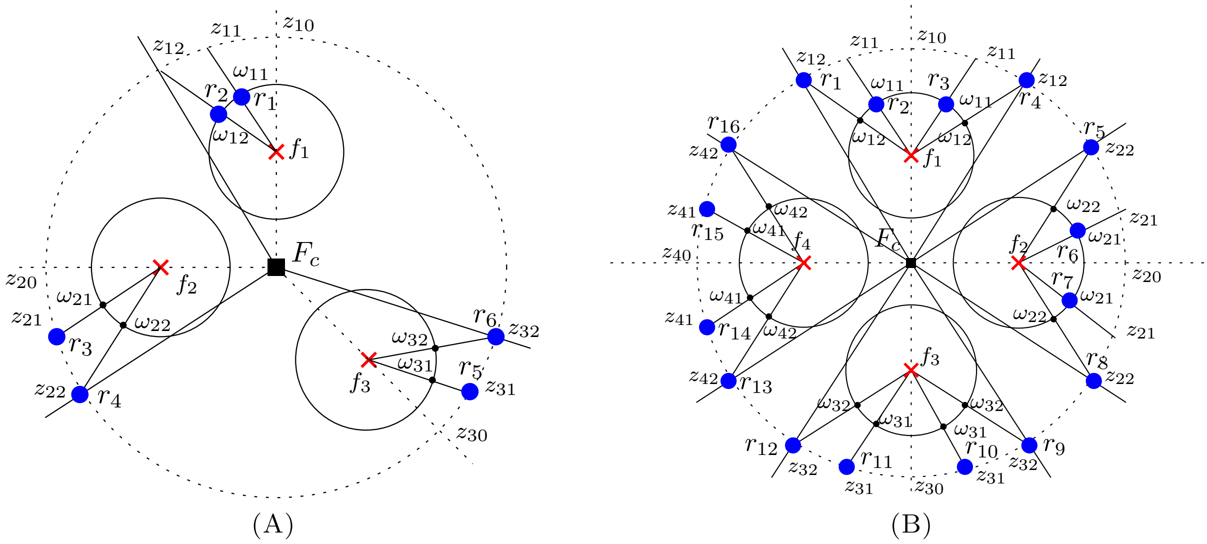


FIGURE 5.4: Examples of Partially Suitable Configurations (**A**) $C(t) \in \mathcal{F}ASYM$, $k = 2$ and $m = 3$. (**B**) $C(t) \in \mathcal{F}REFL$, $k = 4$ and $m = 4$.

Construction of a suitable configuration. Without loss of generality, assume that $\mathcal{F}_m$ is a master ray. Suppose $\mathcal{F}_i$ represents the i^{th} ray encountered in the clockwise direction from $\mathcal{F}_m$. For some $j \in \{\beta_i k - 1, \ldots, 2, 1\}$, let $\mathscr{L}_{ij}$ denote the set of all the straight lines such that any $L \in \mathscr{L}_{ij}$ passes through exactly two points from the set $F \cup \{\omega_{ab}|\ a \in \{1, 2, \ldots, i-1\}\ and\ b \in \{1, 2, \ldots, \beta_i k\}\} \cup \{\omega_{ab}|\ a = i\ and\ b \in \{j+1, \ldots, \beta_i k\}\}$. Let $j = kk_1 + k_2$ where $0 \leq k_2 < k$ and $1 \leq k_1 < \beta_i$. When $k_2 = 0$, then define ω_{ij} as the point on $C(f_{k_1}, \rho)$ such that

$$\measuredangle \overline{f_{k_1} z_{i(j+1)}} f_{k_1} \overline{f_{k_1} \omega_{ij}} = \frac{1}{p} \measuredangle \overline{f_{k_1} z_{i(j+1)}} f_{k_1} \overline{f_{k_1} z_{i0}}$$

where p is the smallest positive integer for which none of the lines in $\mathscr{L}_{ij}$ passes through ω_{ij}. Also, define z_{ij} to be the intersection point between $Ray(f_{k_1}, \omega_{ij})$ and $\mathscr{C}$. If $k_2 \neq 0$, then define ω_{ij} as the point on $C(f_{k_1+1}, \rho)$ such that

$$\measuredangle \overline{f_{k_1+1} z_{i(j+1)}} f_{k_1+1} \overline{f_{k_1+1} \omega_{ij}} = \frac{1}{p} \measuredangle \overline{f_{k_1+1} z_{i(j+1)}} f_{k_1+1} \overline{f_{k_1+1} z_{i0}}$$

where p is the smallest positive integer for which none of the lines in $\mathscr{L}_{ij}$ passes through ω_{ij}. Also, define z_{ij} to be the intersection point between $Ray(f_{k_1+1}, \omega_{ij})$ and $\mathscr{C}$. Define the following conditions for $C(t)$ at $t \geq 0$:

1. $\boldsymbol{c_1}$: $C(t) \in \mathcal{F}ASYM \cup \mathcal{F}CHIR$ or $C(t) \in \mathcal{F}REFL \cup \mathcal{F}MULT$ and k is odd. Each $r_i \in \mathscr{C}$ is located on some z_{ip} in the counter-clockwise direction from $\mathcal{F}_i \in \mathcal{F}$ for $p \in \{1, 2, \ldots, \beta_i k\}$ (Figures 5.3(A) and 5.4(A)).

2. $\boldsymbol{c_2}$: $C(t) \in \mathcal{F}REFL \cup \mathcal{F}MULT$ and k is even. Each $r_i \in \mathscr{C}$ is located on some z_{ip} for some $p \in \{1, 2, \ldots, \frac{\beta_i k}{2}\}$(Figures 5.3(B) and 5.4(B)).

Definition 5.3.3. *At $t \geq 0$, let $C(t)$ be a given configuration such that all the robots lie on $\mathscr{C}$. If $C(t)$ satisfies either $\boldsymbol{c_1}$ or $\boldsymbol{c_2}$, then it is said to be a suitable configuration. In case, there exists a robot that does not lie on $\mathscr{C}$ and $C(t)$ satisfies either $\boldsymbol{c_1}$ or $\boldsymbol{c_2}$, then it is said to be a partially suitable configuration.*

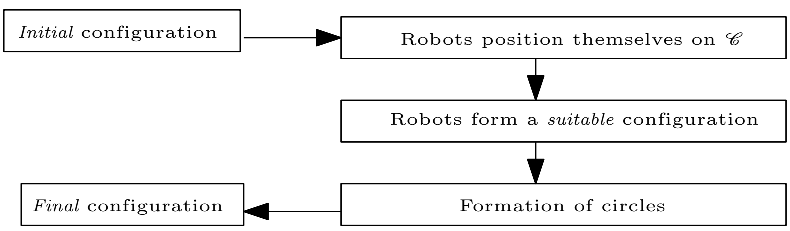


FIGURE 5.5: *OpaqueAlgorithm*1.

5.3.3 Algorithm

In this section, we propose a deterministic distributed algorithm that will solve the *k-circle formation* problem by *oblivious* and *silent* robots. Our proposed distributed algorithm solves the *k-circle formation* problem for $C(0) \notin \mathcal{U}_1$. Since the robots have the knowledge of all the fixed points, $\forall r_i \in R, \forall t \geq 0,\ VFr_i(t) = m$. An overview of our proposed algorithm, $OpaqueAlgorithm1$ is as follows:

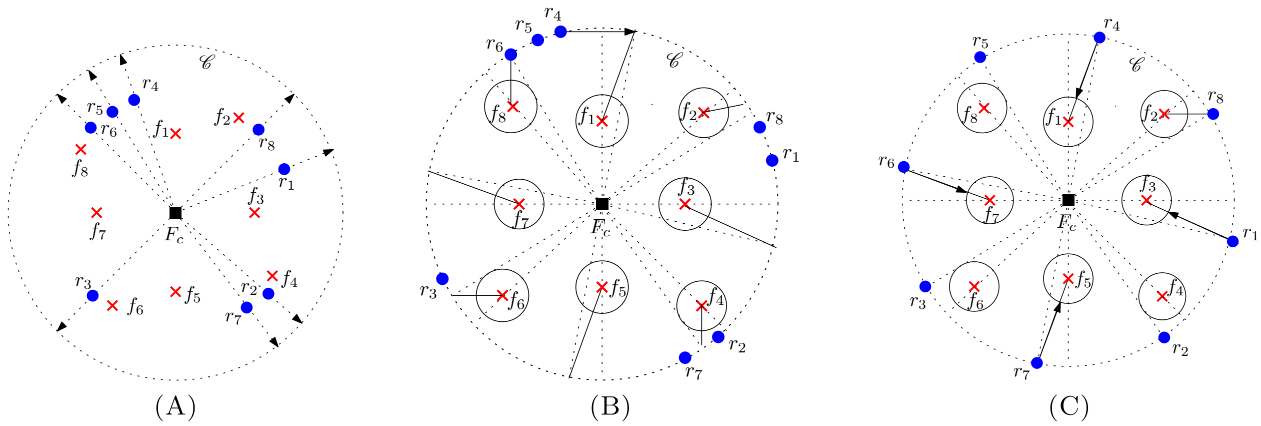


FIGURE 5.6: (**A**) All the robots move towards $\mathscr{C}$, (**B**) The robots form a *suitable* configuration (**C**) The robots start forming circles.

1. All the robots position themselves on the circle $\mathscr{C}$ (Figure 5.6(A)).

2. The robots re-position themselves on $\mathscr{C}$ so that the configuration transforms into a *suitable* configuration (Figure 5.6(B)).

3. The robots start forming circles around the fixed points (Figure 5.6(C)).

Figure 5.5 represents a diagramatic representation of $OpaqueAlgorithm1$. All the phase conditions during $OpaqueAlgorithm1$ are defined in Table 5.1.

5.3.3.1 Phases during $OpaqueAlgorithm1$

We have the following phases during $OpaqueAlgorithm1$:

Conditions	**Descriptions**
P_1	$\forall r \in R,\ VRr(t) = n$
P_2	$C(t) \in \mathcal{F}ASYM$
P_3	$C(t) \in \mathcal{F}REFL$
P_4	$C(t) \in \mathcal{F}CHIR$
P_5	$C(t) \in \mathcal{F}MULT$
P_6	$C(t)$ is a *suitable* configuration
P_7	$C(t)$ is a *partially suitable* configuration
P_8	$\exists f_i \in F$ such that f_i is *unsaturated*
P_9	$\forall r_i \in R,\ r_i(t) \in \mathscr{C}$
P_{10}	There exists exactly one robot (say r) such that $r \notin C(f_j, \rho), \forall f_j \in F$ and $d(r, F_c) < \xi$
P_{11}	There are atmost two robots (say r_1 and r_2) such that $r_i \notin C(f_j, \rho), \forall f_j \in F$ and $d(r_i, F_c) < \xi$ for $r_i \in \{1, 2\}$
P_{12}	There are atmost κ robots (say κ is the degree of rotational symmetry) such that $r_i \notin C(f_j, \rho), \forall f_j \in F$ and $d(r_i, F_c) < \xi$ for $r_i \in \{1, 2, \ldots, \kappa\}$
P_{13}	There are atmost $2\kappa'$ robots (say κ' is the number of lines of symmetry) such that $r_i \notin C(f_j, \rho), \forall f_j \in F$ and $d(r_i, F_c) < \xi$ for $r_i \in \{1, 2, \ldots, \kappa'\}$

TABLE 5.1: Descriptions of the Phase Conditions

1. *Phase*1: In this phase, all the robots position themselves on the circle $\mathscr{C}$. If all the robots lie on $\mathscr{C}$ (condition P_9), then $\forall r_i,\ VRr_i(t) = n$ (condition P_1). A configuration $C(t)$ is said to be in *Phase*1 if it satisfies one of the following conditions:

$$\neg P_1 \wedge \neg P_9 \wedge \neg P_7 \wedge (\neg P_{10} \vee \neg P_{11} \vee \neg P_{12} \vee \neg P_{13}),\ \textit{or}$$

$$P_1 \wedge \neg P_9 \wedge \neg P_6 \wedge (\neg P_{10} \vee \neg P_{11} \vee \neg P_{12} \vee \neg P_{13})$$

The robots would identify this phase by checking whether there exists a robot that does not lie on $\mathscr{C}$ or not.

2. *Phase*2: In this phase, all the robots position themselves on the circle $\mathscr{C}$ so as to form a *suitable* configuration. A configuration $C(t)$ is said to be in *Phase*2 if it satisfies one of the following conditions:

$$P_1 \wedge P_9 \wedge \neg P_6 \wedge (\neg P_{10} \vee \neg P_{11} \vee \neg P_{12} \vee \neg P_{13}),\ \textit{or}$$

$$P_1 \wedge \neg P_9 \wedge \neg P_7 \wedge (P_{10} \vee P_{11} \vee P_{12} \vee P_{13})$$

The robots identify this phase by checking whether $C(t)$ is a *suitable* (condition P_6) or *partially suitable* (condition P_7) configuration.

3. $Phase3$: In this phase the robots start forming circles. A configuration $C(t)$ is said to be in $Phase3$ if it satisfies one of the following conditions:

$$P_1 \wedge P_9 \wedge P_6 \wedge (\neg P_{10} \vee \neg P_{11} \vee \neg P_{12} \vee \neg P_{13}) \wedge P_8, \text{ or}$$
$$P_1 \wedge \neg P_9 \wedge P_7 \wedge (\neg P_{10} \vee \neg P_{11} \vee \neg P_{11} \vee \neg P_{13}) \wedge P_8 \text{ or}$$
$$(\neg P_1 \vee P_1) \wedge \neg P_9 \wedge P_7 \wedge (P_{10} \vee P_{11} \vee P_{12} \vee P_{13}) \wedge P_8$$

   The robots distinguishes this phase from $Phase1$ by checking whether the configuration satisfies conditions P_6 or P_7 and $P_{10} \vee P_{11} \vee P_{12} \vee P_{13}$.

4. $Final$: $C(t)$ is said to be in $Final$ phase if it satisfies the following condition:

$$P_1 \wedge \neg P_9 \wedge (\neg P_{10} \vee \neg P_{11} \vee \neg P_{12} \vee \neg P_{13}) \wedge \neg P_8$$

5.3.3.2 Movements during $OpaqueAlgorithm1$

We define the following types of movements at any arbitrary point of time $t \geq 0$ during an execution of $OpaqueAlgorithm1$:

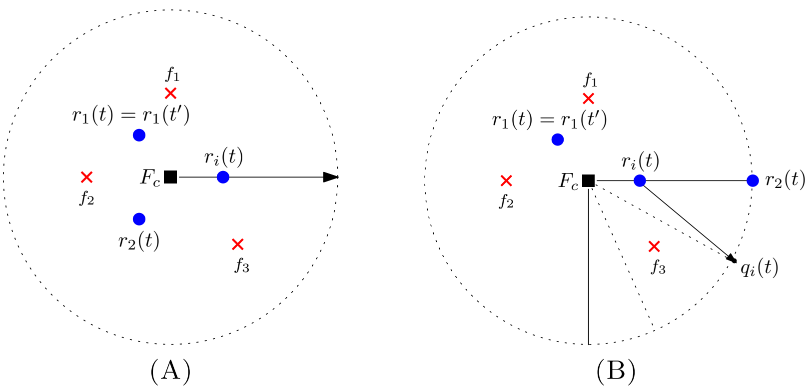


FIGURE 5.7: Movement $\boldsymbol{M}_1$.

1. $\boldsymbol{M_1}$: This movement is executed when $\boldsymbol{C(t)}$ is in $Phase1$. By the execution of movement $\boldsymbol{M}_1$, the robots will eventually position themselves on $\mathscr{C}$. For some $r_i \in R$, let $p_i(t)$ be the intersection point between $\mathscr{C}$ and $Ray(F_c, r_i(t))$. Let $r_i \notin \mathscr{C}$ be a robot such that $d(r_i(t), p_i(t)) = \min_{r_j \in R} d(r_j(t), p_j(t))$. Among all such robots, let r_i be a robot which makes the smallest angle with some $L \in \mathcal{L}'$ centered at

F_c. Let $d = d(r_i(t), p_i(t))$. First, consider the case when $\forall L \in \mathcal{L}'$, mirror image of $\overline{r_i(t)p_i(t)}$ about L is visible to r_i. If $p_i(t)$ is neither a robot position nor a *virtual* robot position (Recall that a point p is said to be a *virtual* robot position at time t, if $\exists\, r_k \in R(t)$ such that p and r_k are symmetric about a line of symmetry $L \in \mathcal{L}'$ as defined in Definition 4.2.3), then r_i starts moving towards $p_i(t)$ along $\overline{p_i(t)r_i(t)}$ (Figure 5.7(A)). If $p_i(t)$ is either a robot position or a virtual robot position or $r_i(t) \in L$ for some $L \in \mathcal{L}'$, let $r_k(t)$ be such that $\measuredangle Ray(F_c, r_i(t))F_cRay(F_c, r_k(t)) = \min_{r_j \in R} \measuredangle Ray(F_c, r_i(t))F_cRay(F_c, r_j(t))$. Assume that B denotes the ray that starts from F_c and satisfies the condition

$$\measuredangle Ray(F_c, r_i(t))F_cB = \frac{1}{3d}\min(\measuredangle Ray(F_c, r_i(t))F_cRay(F_c, r_k(t)), \frac{\pi}{4})$$

Suppose q denotes the intersection point between B and $\mathscr{C}$. Robot r_i moves towards q along $\overline{r_i(t)q}$ (Figure 5.7(B)). Next, consider the case when $\exists L \in \mathcal{L}'$ such that mirror image of $\overline{r_i(t)p_i(t)}$ about L is not visible to r_i. Let $q_1 \in \overline{r_i(t)p_i(t)}$ be the point that lies at the closest distance from r_i such that mirror image of q_1 is not visible to r_i for some $L \in \mathcal{L}'$. Let $q_2 \in \overline{r_i(t)p_i(t)}$ be the point such that $\overline{q_2r_i(t)} \perp L$. Robot r_i moves towards q_2 along $\overline{r_i(t)q_2}$ (Figure 5.8).

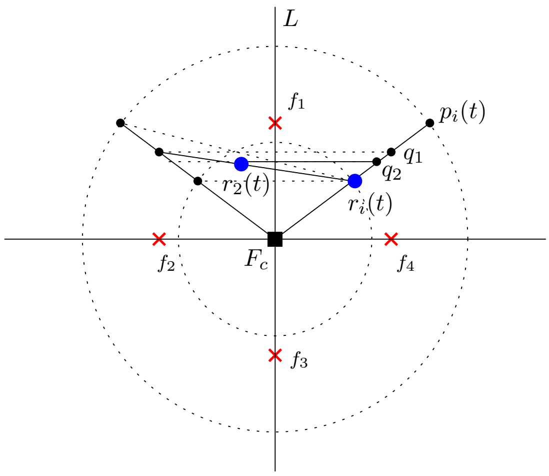


FIGURE 5.8: Movement $\boldsymbol{M}_1$.

2. $\boldsymbol{M_{21}}$: When $\boldsymbol{C(t)}$ is in *Phase*2 and $C(t) \in \mathcal{F}ASYM$ or k is odd and $C(t) \in \mathcal{F}REFL \cup \mathcal{F}MULT$, movement $\boldsymbol{M_{21}}$ is executed. By the execution of movement $\boldsymbol{M}_{21}$, the robots will form a *suitable* configuration. In case k is odd and $C(t) \in \mathcal{F}REFL \cup \mathcal{F}MULT$, there can be more than one *master* rays. As $C(0) \notin \{\mathcal{U}_1 \cup \mathcal{U}_2\}$,

in such cases, the configuration must be asymmetric or admit a line of symmetry (say L) such that $L \cap R(t) \neq \emptyset$. A robot $r \in L$ moves away from L so that the configuration becomes asymmetric about L. The destination point of r is computed by avoiding collision with other robots. Next, the configuration becomes asymmetric. Suppose $\mathcal{F}_m$ is the *master* ray that contains the fixed point with the minimum configuration view as discussed in section 4.2.1. Let $\mathcal{F}_i \in \mathcal{F}$ be such that $\measuredangle \mathcal{F}_m F_c \mathcal{F}_i = \min_{\mathcal{F}_m \neq \mathcal{F}_b} \mathcal{F}_m F_c \mathcal{F}_b$ measured in the counter clockwise direction and $\exists z_{ip}$ for some $p \in \{1, 2, \ldots, \beta_i k\}$ such that z_{ip} does not contain any robot positions. Suppose $q \in \{1, 2, \ldots, \beta_i k\}$ denotes the smallest positive integer for which z_{iq} does not contain any robot positions. We have the following cases:

(a) $\boldsymbol{C(t)}$ satisfies the phase condition $P_1 \wedge P_9 \wedge \neg P_6 \wedge \neg P_{10}$. Let r_k be such that $\measuredangle \overline{F_c r_k} F_c \overline{F_c z_{iq}} = \min_{r_j \neq r_k} \measuredangle \overline{F_c r_j} F_c \overline{F_c z_{iq}}$ measured in the counter clockwise direction. r_k moves along $\overline{r_k(t) z_{iq}}$ towards z_{iq} (Figure 5.9(A)).

(b) $\boldsymbol{C(t)}$ satisfies the phase condition $P_1 \wedge \neg P_9 \wedge \neg P_7 \wedge P_{10}$. Let r_k be the robot such that $d(r_k(t), z_{iq}) < \xi$ and r_k lies at the closest distance from z_{iq}. r_k moves along $\overline{r_k(t) z_{iq}}$ towards z_{iq} (Figure 5.9(B)).

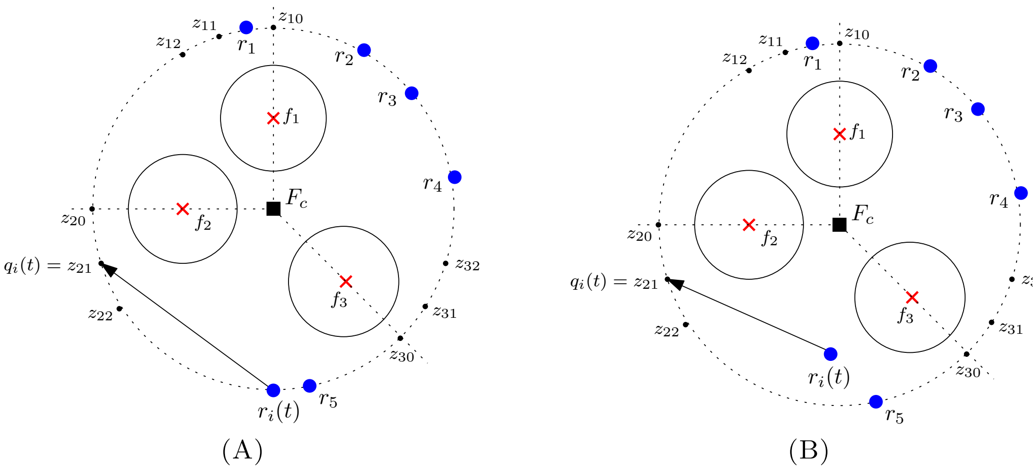


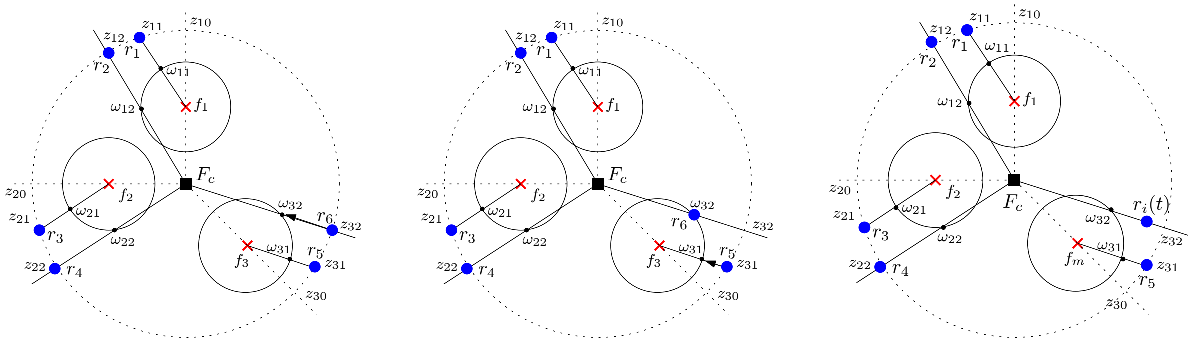
FIGURE 5.9: Movement $\boldsymbol{M_{21}}$.

3. $\boldsymbol{M_{22}}$: When the configuration $C(t)$ is in $Phase2$ and $C(t) \in \mathcal{F}REFL \cup \mathcal{F}MULT$ and k is even, movement $\boldsymbol{M_{22}}$ is carried out. Assume that $\mathcal{F}_m \in \mathcal{F}$ is a *master* ray. Note that there can be multiple *master* rays. Let $\mathcal{F}_i \in \mathcal{F}$ be such that $\measuredangle \mathcal{F}_i F_c \mathcal{F}_m = \min_{\mathcal{F}_b \neq \mathcal{F}_m} \mathcal{F}_b F_c \mathcal{F}_m$ (there can be two such rays) and $\exists z_{ip}$ for some $p \in$

$\{1, 2, \ldots, \frac{\beta_i k}{2}\}$ such that z_{ip} does not contain any robot positions. Suppose $q \in \{1, 2, \ldots, \frac{\beta_i k}{2}\}$ denotes the smallest positive integer for which z_{iq} does not contain any robot positions. There can be two such positions. We have the following cases:

(a) $\boldsymbol{C(t)}$ satisfies $P_1 \wedge P_9 \wedge \neg P_6 \wedge (\neg P_{11} \vee \neg P_{13})$. Let r_k be such that $\measuredangle \overline{F_c r_k} F_c \overline{F_c z_{iq}} = \min_{r_j \neq r_k} \measuredangle \overline{F_c r_j} F_c \overline{F_c z_{iq}}$. r_k moves along $\overline{r_k(t) z_{iq}}$ towards z_{iq}.

(b) $\boldsymbol{C(t)}$ satisfies $P_1 \wedge \neg P_9 \wedge \neg P_7 \wedge (P_{11} \vee P_{13})$. Let r_k be the robot such that $d(r_k(t), z_{iq}) < \xi$ and r_k lies at the closest distance from z_{iq}. r_k moves along $\overline{r_k(t) z_{iq}}$ towards z_{iq}.



(A) (B) (C)

FIGURE 5.10: Movement $\boldsymbol{M_3}$.

4. $\boldsymbol{M_{23}}$: When $C(t)$ is in $Phase2$ and $C(t) \in \mathcal{FCHIR}$, movement $\boldsymbol{M_{23}}$ is executed. As there are multiple *master* rays, movement $\boldsymbol{M_{23}}$ represents the execution of movement $\boldsymbol{M_{21}}$ in multiple wedges.

5. $\boldsymbol{M_3}$: This movement is executed when $\boldsymbol{C(t)}$ is in $Phase3$. By the execution of movement $\boldsymbol{M}_3$, the robots will form a *final* configuration. Since $\forall r_i \in R$, $VFr_i(t) = m$, the robots can compute the radius ρ without any conflict. Suppose $\mathcal{F}_m$ is a *master* ray. We have the following cases:

(a) $\boldsymbol{C(t)}$ satisfies $P_1 \wedge P_9 \wedge P_6 \wedge (\neg P_{10} \vee \neg P_{11} \vee \neg P_{11} \vee \neg P_{13}) \wedge P_8$. Let $f_i \in \mathcal{F}_m$ be the *unsaturated* fixed point that lies at shortest distance from F_c. Since $C(t)$ satisfies P_9, $f_i \in F_m$ is the 1^{st} fixed point according to distance from F_c. Suppose r lies on $z_{i(\beta_i k)}$. r moves towards $\omega_{i(\beta_i k)}$ along $\overline{\omega_{i(\beta_i k)} z_{i(\beta_i k)}}$ (Figure 5.10(A)).

(b) $\boldsymbol{C(t)}$ satisfies $P_1 \wedge \neg P_9 \wedge P_7 \wedge (\neg P_{10} \vee \neg P_{11} \vee \neg P_{11} \vee \neg P_{13}) \wedge P_8$. Let $\mathcal{F}_j \in \mathcal{F}$ be such that $\measuredangle \mathcal{F}_j F_c \mathcal{F}_m = \min_{\mathcal{F}_k \in \mathcal{F}} \measuredangle \mathcal{F}_k F_c \mathcal{F}_m$ and it contains an *unsaturated* fixed point. Let p be the smallest positive integer for which ω_{jp} does not contain a robot position. Suppose r lies on z_{jp}. r moves towards ω_{jp} along $\overline{\omega_{jp} z_{jp}}$ (Figure 5.10(B)).

(c) $\boldsymbol{C(t)}$ satisfies $(\neg P_1 \vee P_1) \wedge \neg P_9 \wedge P_7 \wedge (P_{10} \vee P_{11} \vee P_{11} \vee P_{13}) \wedge P_8$. Let $\mathcal{F}_j \in \mathcal{F}$ be such that $\measuredangle \mathcal{F}_j F_c \mathcal{F}_m = \min_{\mathcal{F}_k \in \mathcal{F}} \measuredangle \mathcal{F}_k F_c \mathcal{F}_m$ and it contains an *unsaturated* fixed point. Let p be the smallest positive integer for which ω_{jp} does not contain a robot position. Let r be the robot that lies at the closest distance from ω_{jp} such that $d(r(t), F_c) < \xi$. Also, r does not lie on any *saturated* circles and on any ω_{jb} such that $b \in \{1, 2, \ldots, p-1\}$. r moves towards ω_{jp} along $\overline{r(t)\omega_{jp}}$ (Figure 5.10(C)).

Phases	Movements	Transformed Phases
*Phase*1	M_1	*Phase*1 or *Phase*2
*Phase*2	M_{21}	*Phase*2 or *Phase*3
*Phase*2	M_{22}	*Phase*2 or *Phase*3
*Phase*2	M_{23}	*Phase*2 or *Phase*3
*Phase*3	M_3	*Phase*3 or *Final*

TABLE 5.2: Phase Transitions during *OpaqueAlgorithm*1

ALGORITHM 5.1: *OpaqueAlgorithm*1

Input: $C(t) = (R(t), F)$
if *$C(t)$ is in Phase1* **then**
  Execute M_1;
else if *$C(t)$ is in Phase2* **then**
  if *$C(t) \in \mathcal{F}ASYM$ or k is odd and $C(t) \in \mathcal{F}REFL \cup \mathcal{F}MULT$* **then**
    Execute M_{21};
  else if *k is even and $C(t) \in \mathcal{F}REFL \cup \mathcal{F}MULT$* **then**
    Execute M_{22};
  else if *$C(t) \in \mathcal{F}CHIR$* **then**
    Execute M_{23};
  end
else if *$C(t)$ is in Phase3* **then**
  Execute M_3;
end

5.3.3.3 *OpaqueAlgorithm*1

An active robot executes *OpaqueAlgorithm*1 unless $C(t)$ is a *final* configuration. During an execution of *OpaqueAlgorithm*1 the following cases are to be considered:

1. $\boldsymbol{C(t)}$ is in *Phase*1, movement $\boldsymbol{M_1}$ is executed.

2. $\boldsymbol{C(t)}$ is in *Phase*2. First, consider the case when $C(t) \in \mathcal{F}ASYM$ or $C(t) \in \mathcal{F}REFL \cup \mathcal{F}MULT$ and k is odd. Movement $\boldsymbol{M_{21}}$ is executed. When $C(t) \in \mathcal{F}REFL \cup \mathcal{F}MULT$ and k is even, movement $\boldsymbol{M_{22}}$ is executed. Movement $\boldsymbol{M_{23}}$ is executed for $C(t) \in \mathcal{F}CHIR$.

3. $\boldsymbol{C(t)}$ is in *Phase*3. Movement $\boldsymbol{M_3}$ is executed.

A summary of the movemnets during an execution of *OpaqueAlgorithm*1 is presented in Table 5.2. Figures 5.11(A), 5.11(B) and 5.11(C) represent the phase transitions of *OpaqueAlgorithm*1. The pseudocode of *OpaqueAlgorithm*1 is presented in Algorithm 5.1.

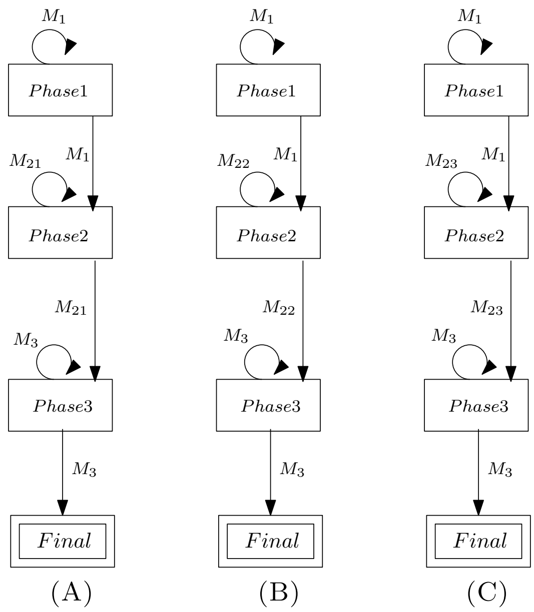


FIGURE 5.11: Phase transitions during *OpaqueAlgorithm*1. (**A**) $C(t) \in \mathcal{F}ASYM$ or $C(t) \in \mathcal{F}CHIR \cup \mathcal{F}MULT$ and k is odd, (**B**) $C(t) \in \mathcal{F}CHIR \cup \mathcal{F}MULT$ and k is even, (**C**) $C(t) \in \mathcal{F}CHIR$.

5.3.4 Correctness of *OpaqueAlgorithm*1

We first show that when the *initial* configuration $C(0)$ is *solvable*, i.e., $C(0) \notin \mathcal{U}_1$, then $C(t)$ at any arbitrary point of time $t > 0$ would remain *solvable*, i.e., $C(t) \notin \mathcal{U}_1$.

Lemma 5.3.4. *If $C(0) \in \mathcal{F}ASYM \cup \mathcal{F}CHIR$, then $\forall t \geq 0$, $C(t)$ would remain solvable during any execution of OpaqueAlgorithm*1.

Proof. As discussed in section 4.2.4, if $C(t) \in \mathcal{F}CHIR$, then the robots can make an agreement on a common *chirality*. Consider the case when $C(t) \in \mathcal{F}ASYM$. Let $f \in F$ be the fixed point that has the minimum configuration view. The direction of $\mathscr{V}(f)$ is globally considered to be the clockwise direction. If there is a tie due to symmetric positions, then such a tie can be broken with respect to *chirality*. Therefore, during any execution of $OpaqueAlgorithm1$, $\forall t \geq 0$, $C(t)$ would remain *solvable*. □

If $C(0) \in \mathcal{F}REFL \cup \mathcal{F}MULT$, then we only need to consider configurations when k is odd, $|F|$ is even and $\mathcal{L}' \neq \emptyset$ (Observations 1, 2 and 3).

Lemma 5.3.5. *If k is odd, $C(0) \in \mathcal{F}REFL \cup \mathcal{F}MULT$ and $C(0) \notin \mathcal{U}_1$, then during an execution of $OpaqueAlgorithm1$, $C(t)$ would remain solvable $\forall t \geq 0$.*

Proof. Consider the following cases:

Case 1. $C(t)$ would remain *solvable* during movement $\boldsymbol{M_1}$.

Subcase 1. $C(0)$ is symmetric about an $L \in \mathcal{L}'$. It follows from Theorem 5.3.1 that L must contain a robot position. First, consider that r would move along $\overline{F_c r}$. Since F_c lies on L, r would remain on L. As a consequence, $C(t)$ would remain *solvable*. If r moves from L towards one of the half-planes delimited by L, the intersection point between $\mathscr{C}$ and L must already have one robot position. As a consequence, $C(t)$ would remain *solvable*.

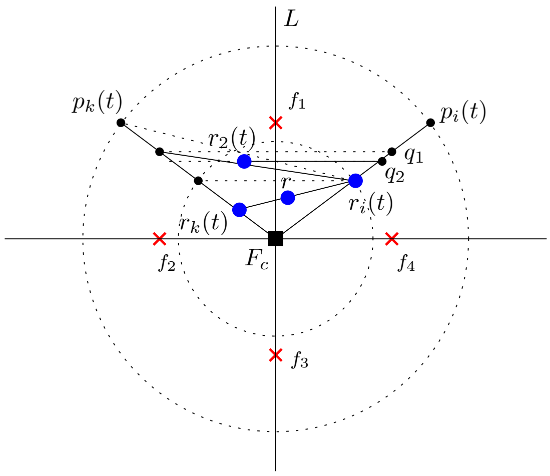


FIGURE 5.12: Illustration of solvability during Movement $\boldsymbol{M}_1$.

Subcase 2. $C(0)$ is asymmetric about each $L \in \mathcal{L}'$. During movement $\boldsymbol{M_1}$, a robot r_i only moves towards the point $p_i(t) \in \mathscr{C}$ if mirror images of $\overline{r_i(t)p_i(t)}$ about each $L \in \mathcal{L}'$

is visible to r_i. If r_i is able to see a robot position or a *virtual* robot position on $\overline{r_i(t)p_i(t)}$ other than $p_i(t)$ it does not move. This is because $r_i \notin \mathscr{C}$ identifies itself to be not the farthest robot from F_c. In case, r_i is able to see a robot position or a *virtual* robot position on $p_i(t)$, it suitably selects a destination point on $\mathscr{C}$ such that the configuration remain asymmetric (Figure 5.7(B)). If $\exists L \in \mathcal{L}'$ such that mirror image of $\overline{r_i(t)p_i(t)}$ about L is not visible to r_i, then it selects a point on $\overline{r_i(t)p_i(t)}$ by avoiding possible symmetry. Assume that there exists a robot position $r_k(t)$ such that $\overline{F_c p_k(t)}$ and $\overline{F_c p_i(t)}$ are mirror images about L. If $d(F_c, r_k(t)) = d(F_c, r_i(t))$, then the configuration must have a different asymmetric pair of robots about L. Consider the case when $d(F_c, r_k(t)) > d(F_c, r_i(t))$. If r_k is visible to r_i, then r_i identifies itself to be not the farthest robot from F_c and r_i does not move. If r_k is not visible, then by the choice of q_2 (as discussed in section 5.3.3.2), r_k and r_i would not become symmetric about L. Assume that $d(F_c, r_k(t)) < d(F_c, r_i(t))$, and r_k and r_i cannot see each other due to presence of a robot position (say r)(Figure 5.12). If r_k decides to move, then by the choice of q_2' for r_k (as discussed in section 5.3.3.2), it is ensured that $d(F_c, q_2') < d(F_c, r_i(t))$. As a consequence, r_k and r_i would not become symmetric about L.

Case 2. $C(t)$ would remain *solvable* during movement $\boldsymbol{M_{21}}$. If $C(t)$ is symmetric about L, then r moves towards one of half-planes delimited by L. As a result, $C(t)$ would become asymmetric about L. Next, the robots would form a *suitable* configuration. During movement $\boldsymbol{M_1}$, a unique robot (say r_1) is selected for moving towards its destination point. During its motion, r_1 is the only robot that would satisfy $d(F_c, r_1) < \xi$. As a consequence, $C(t)$ would remain asymmetric about each $L \in \mathcal{L}'$. From the definition of a *suitable* configuration, it follows that $C(t)$ would remain asymmetric about each $L \in \mathcal{L}'$. As a consequence, $C(t)$ would remain *solvable*.

Case 3. $C(t)$ would remain *solvable* during movement $\boldsymbol{M_3}$. From the definition of a *suitable* configuration, it follows that all the robot positions in a *suitable* configuration are asymmetric about each $L \in \mathcal{L}$. For each robot on a z_{ij} for some $\mathcal{F}_i \in \mathcal{F}$ and $j \in \{1, 2, \ldots, \beta_i\}$ the destination point ω_{ij} would remain invariant. From the definition of ω_{ij}, it also follows that a robot position on a ω_{ij} for some $\mathcal{F}_i \in \mathcal{F}$ and $j \in \{1, 2, \ldots, \beta_i\}$ would remain asymmetric. As a consequence, $C(t)$ would remain asymmetric about each $L \in \mathcal{L}$. Hence, $C(t)$ would remain *solvable*. □

Now we proceed to show that the robots will solve the *k-circle formation* problem within finite time. First, we will discuss *progress* during movement $\boldsymbol{M_1}$. Let $C(t)$ be in

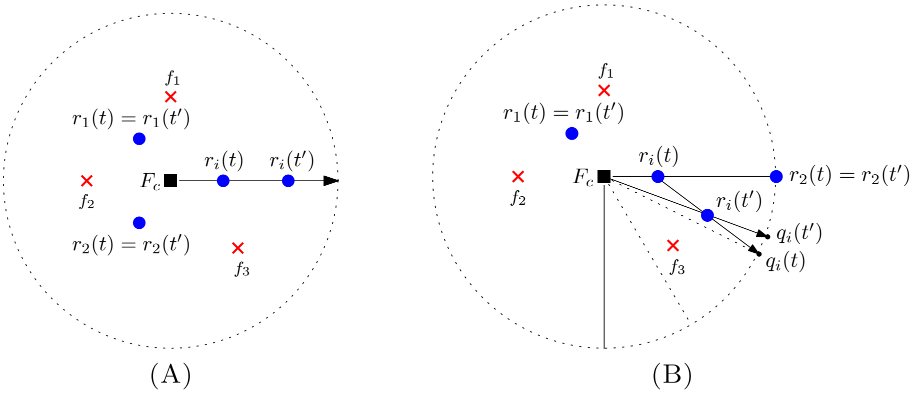


FIGURE 5.13: Progress during movement $\boldsymbol{M_1}$.

*Phase*1. Suppose r_i denotes a *candidate* robot and $q_i(t)$ represents the destination point of r at time t. Let $\mathcal{N}_2(t)$ denote the number of robots which do not lie on $\mathscr{C}$. Also, let $g_i(t) = d(r_i(t), q_i(t))$. Define $Z_1(t) = (\mathcal{N}_2(t), g_i(t))$.

Lemma 5.3.6. *Let $C(t)$ be in Phase*1. *Also, let r_i be a candidate robot and $t' > t$ be an arbitrary point of time at which r_i has completed at least one LCM cycle. Execution of movement $\boldsymbol{M_1}$ ensures that $g_i(t') + \delta \leq g_i(t)$.*

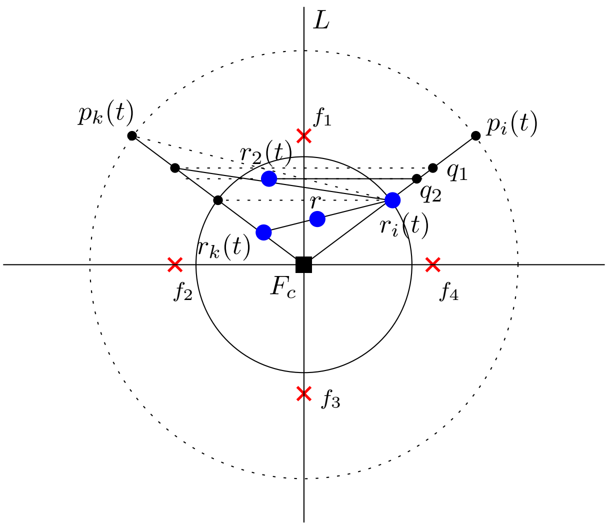


FIGURE 5.14: Illustration of progress during Movement $\boldsymbol{M}_1$.

Proof. Recall that $p_i(t)$ denotes the intersection point between $Ray(F_c, r_i(t))$ and $\mathscr{C}$. First, consider the case when $\exists L \in \mathcal{L}'$ such that mirror image of $\overline{r_i(t)p_i(t)}$ about L is not visible to r_i. Let $q_1 \in \overline{r_i(t)p_i(t)}$ be the point that lies at the closest distance from r_i

such that mirror image of q_1 is not visible to r_i for some $L \in \mathcal{L}'$ (say due to the robot position r_2). By movement $\boldsymbol{M_1}$, r_i selects a destination point q_2 on $\overline{r_i(t)p_i(t)}$ as discussed in section 5.3.3.2 and moves directly towards it (Figure 5.14). By the choice of r_i, we have $d(F_c, r_i(t)) > d(F_c, r_2(t))$. Thus, r_2 would not move. By movement $\boldsymbol{M_1}$, r_i would reach q_2 and r_2 would not block a point on the mirror image of $\overline{r_i(t)p_i(t)}$. Since there are only finite number of robot positions, r_i would reach a point on $\overline{r_i(t)p_i(t)}$ such that $\forall L \in \mathcal{L}'$ mirror image of $\overline{r_i(t)p_i(t)}$ about L is visible to r_i. Next, the following cases are to be considered:

Case 1. $p_i(t)$ is neither a robot position nor a *virtual* robot position. In this case, $q_i(t) = p_i(t)$ and r_i moves directly towards $p_i(t)$ (Figure 5.13(A)). As r_i moves by at least δ, $g_i(t') + \delta \leq g_i(t)$.

Case 2. $p_i(t)$ is either a robot position or a *virtual* robot position or $r_i(t) \in L$ for some $L \in \mathcal{L}'$. r_i computes its destination point according to movement $\boldsymbol{M_1}$ (Figure 5.13(B)). At t', $d(r_i(t'), q_i(t')) > d(r_i(t'), q_i(t))$. As a consequence, we have $d(r_i(t), q_i(t)) - d(r_i(t'), q_i(t)) > d(r_i(t), q_i(t)) - d(r_i(t'), q_i(t')) \geq \delta$. Thus, $g_i(t') + \delta \leq g_i(t)$.

Hence, an execution of movement $\boldsymbol{M_1}$ ensures $g_i(t') + \delta \leq g_i(t)$. □

Lemma 5.3.7. *During an execution of movement $\boldsymbol{M_1}$, let $t' > t$ be an arbitrary point of time at which a candidate robot r_i has completed at least one LCM cycle. An execution of movement $\boldsymbol{M_1}$ ensures $Z_1(t') < Z_1(t)$.*

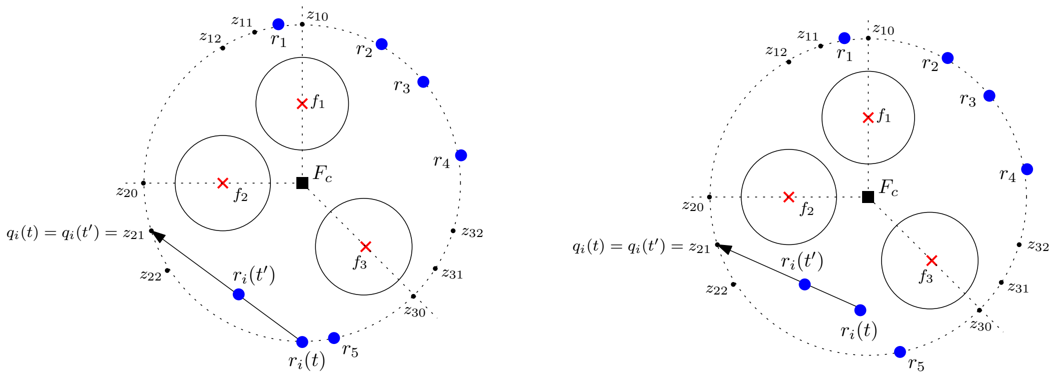
Proof. The following cases are to be considered:

Case 1. $q_i(t) = r_i(t')$. *As $\mathcal{N}_2(t') = \mathcal{N}_2(t) - 1$, $Z_1(t') < Z_1(t)$ is ensured.*

Case 2. $q_i(t) \neq r_i(t')$. *Lemma 5.3.6 ensures that $g_i(t') + \delta \leq g_i(t)$.*

Hence, an execution of movement $\boldsymbol{M_1}$ ensures $Z_1(t') < Z_1(t)$. □

Let $\boldsymbol{C(t)}$ be in *Phase*2. Suppose r_i denotes a *candidate* robot. Assume that $q_i(t)$ represents the destination point of r at time t. Let $\mathcal{N}_3(t)$ denote the number of robots which do not lie on any z_{ij} for some $\mathcal{F}_i \in \mathcal{F}$ and $j \in \{1, 2, \ldots, \beta_i k\}$. When k is even and $C(t) \in \mathcal{F}REFL \cup \mathcal{F}MULT$, $\mathcal{N}_3(t)$ denotes the number of robots which do not lie on



(A) (B)

FIGURE 5.15: Progress during movement $\boldsymbol{M_{21}}$

any z_{ij} for some $\mathcal{F}_i \in \mathcal{F}$ and $j \in \{1, 2, \ldots, \dfrac{\beta_i k}{2}\}$. Also, let $g_i(t) = d(r_i(t), q_i(t))$. Define $Z_2(t) = (\mathcal{N}_3(t), g_i(t))$.

Lemma 5.3.8. *Let $C(t)$ be in Phase2. Also, let r_i be a candidate robot and $t' > t$ be an arbitrary point of time at which r_i has completed at least one LCM cycle. Execution of movement $\boldsymbol{M_{21}}$ ensures that $g_i(t') + \delta \leq g_i(t)$.*

Proof. During movement $\boldsymbol{M_{21}}$, $q_i(t') = q_i(t)$ and r_i moves directly towards $q_i(t)$ (Figure 5.15(A) and 5.15(B)). Since r_i moves by at least δ, $g_i(t') + \delta \leq g_i(t)$. Hence, an execution of movement $\boldsymbol{M_{21}}$ ensures $g_i(t') + \delta \leq g_i(t)$. □

Lemma 5.3.9. *During an execution of movement $\boldsymbol{M_{21}}$, let $t' > t$ be an arbitrary point of time at which a candidate robot r_i has completed at least one LCM cycle. An execution of movement $\boldsymbol{M_{21}}$ ensures $Z_2(t') < Z_2(t)$.*

Proof. The following cases are to be considered:

Case 1. *$q_i(t) = r_i(t')$. As $\mathcal{N}_3(t') = \mathcal{N}_3(t) - 1$, $Z_2(t') < Z_2(t)$ is ensured.*

Case 2. *$q_i(t) \neq r_i(t')$. Lemma 5.3.8 ensures that $g_i(t') + \delta \leq g_i(t)$.*

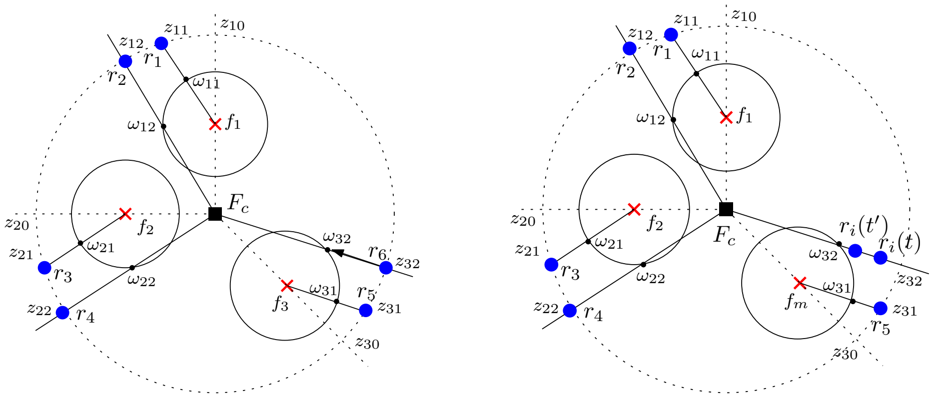
Hence, an execution of movement $\boldsymbol{M_{21}}$ ensures $Z_2(t') < Z_2(t)$. □

Lemma 5.3.10. *During an execution of movement $\boldsymbol{M_{22}}$, let $t' > t$ be an arbitrary point of time at which at least one candidate robot (say r_i) has completed at least one LCM cycle. An execution of movement $\boldsymbol{M_{22}}$ ensures $Z_2(t') < Z_2(t)$.*

Proof. During movement $\boldsymbol{M_{22}}$, $q_i(t') = q_i(t)$ and r_i moves directly towards $q_i(t)$. If $q_i(t) = r_i(t')$, then $\mathcal{N}_3(t') = \mathcal{N}_3(t) - 1$. Thus, $Z_2(t') < Z_2(t)$ is ensured. Consider the case when $q_i(t) \neq r_i(t')$. Since r_i moves by at least δ, $g_i(t') + \delta \leq g_i(t)$. Hence, an execution of movement $\boldsymbol{M_{22}}$ ensures $Z_2(t') < Z_2(t)$. □

Lemma 5.3.11. *During an execution of movement $\boldsymbol{M_{23}}$, let $t' > t$ be an arbitrary point of time at which a candidate robot r_i has completed at least one LCM cycle. An execution of movement $\boldsymbol{M_{23}}$ ensures $Z_2(t') < Z_2(t)$.*

Proof. During an execution of movement $\boldsymbol{M_{23}}$, movement $\boldsymbol{M_{21}}$ will be executed in multiple wedges. From Lemma 5.3.9, it follows that $Z_2(t') < Z_2(t)$ will be ensured. □



(A) (B)

FIGURE 5.16: Progress during movement $\boldsymbol{M_3}$.

Let $\boldsymbol{C(t)}$ be in *Phase*3. Suppose r_i denotes a *candidate* robot for the *target* fixed point $f_j \in \mathcal{F}_k \in \mathcal{F}$ and $q_i(t)$ represents the destination point of r_i at time t. Recall that $D_j(t) = k - |C(f_i, \rho) \cap R(t)|$ denote the deficit of number of robots on $C(f_j, \rho)$ to become *saturate* at time t. Also, recall that $n_k(t)$ denotes the number of *unsaturated* fixed points. Also, let $g_i(t) = d(r_i(t), q_i(t))$. Define $V_i(t) = (n_k(t), D_j(t), g_i(t))$.

Lemma 5.3.12. *Let $C(t)$ be in Phase3. Also, let r_i be a candidate robot and $t' > t$ be an arbitrary point of time at which r_i has completed at least one LCM cycle. Execution of movement $\boldsymbol{M_3}$ ensures that $g_i(t') + \delta \leq g_i(t)$.*

Proof. During movement $\boldsymbol{M_3}$, r_i moves directly towards $q_i(t)$ (Figures 5.16(A) and 5.16(B)). Since r_i moves by at least δ, $g_i(t') + \delta \leq g_i(t)$. Hence, an execution of movement $\boldsymbol{M_3}$ ensures $g_i(t') + \delta \leq g_i(t)$. □

Lemma 5.3.13. *During an execution of movement $\boldsymbol{M_3}$, let $t' > t$ be an arbitrary point of time at which at least one candidate robot r_i has completed at least one LCM cycle. An execution of movement $\boldsymbol{M_3}$ ensures $V_i(t') < V_i(t)$.*

Proof. The following cases are to be considered:

Case 1. $q_i(t) = r_i(t')$. If $C(f_j, \rho)$ has exactly k robots, then $n_k(t') = n_k(t) - 1$, ensuring $V_2(t') < V_2(t)$. If $C(f_j, \rho)$ has less than k robots on it, then $D_j(t') = D_j(t) - 1$, ensuring $V_i(t') < V_i(t)$.

Case 2. $q_i(t) \neq r_i(t')$. Lemma 5.3.12 ensures that $g_i(t') + \delta \leq g_i(t)$. As a result, $V_i(t') < V_i(t)$ is ensured.

Hence, an execution of movement $\boldsymbol{M_3}$ ensures $V_i(t') < V_i(t)$. □

Theorem 5.3.14. *Let $C(0) \notin \mathcal{U}_1$ be a given initial configuration. Execution of algorithm $OpaqueAlgorithm1$ would solve the k-circle formation problem within finite time under obstructed visibility model.*

Proof. Lemmata 5.3.4 and 5.3.5 ensure that $\forall t \geq 0$, $C(t)$ would remain *solvable*. At $t \geq 0$, we have the following cases:

Case 1. $\boldsymbol{C(t)}$ is in $Phase1$. Movement $\boldsymbol{M_1}$ is executed. Lemma 5.3.7 ensures that within finite time all the robots will reach $\mathscr{C}$.

Case 2. $\boldsymbol{C(t)}$ is in $Phase2$. If $C(t) \in \mathcal{F}ASYM$ or k is odd and $C(t) \in \mathcal{F}REFL \cup \mathcal{F}MULT$, then by Lemma 5.3.9, formation of a *suitable* configuration is ensured by movement $\boldsymbol{M_{21}}$. If k is even and $C(t) \in \mathcal{F}REFL \cup \mathcal{F}MULT$, then movement $\boldsymbol{M_{22}}$ is executed. Lemma 5.3.10 ensures that within finite time the robots will form a *suitable* configuration. In case $C(t) \in \mathcal{F}CHIR$, movement $\boldsymbol{M_{23}}$ is executed. Lemma 5.3.11 ensures that within finite the robots will form a *suitable* configuration.

Case 3. $\boldsymbol{C(t)}$ is in $Phase3$. Lemma 5.3.13 ensures that within finite time the robots will form a *final* configuration by the execution of movement $\boldsymbol{M_3}$.

Hence, $OpaqueAlgorithm1$ would solve the *k-circle formation* problem within finite time under obstructed visibility model for $C(0) \notin \mathcal{U}_1$. □

5.4 Zero Knowledge of the Fixed Points

In this section, we consider the setting in which the robots have zero knowledge of the fixed points. Since $\forall r_i \in R(0)$, $Fr_i(0) \leq m$, the robots must detect the total number of fixed points in order to solve the *k-circle formation* problem.

5.4.1 Impossibility Results

Theorem 5.4.1. *If the robots have zero knowledge of fixed points, then the k-circle formation problem is deterministically unsolvable by oblivious and silent robots.*

Proof. Let $C(0)$ be an *initial* configuration in which the *k-circle formation* problem has already been solved, i.e., $C(0)$ is itself a *final* configuration. The robots do not have the knowledge of the total number of fixed points or the total number of robots. As a consequence, the robots cannot identify a *final* configuration. Hence, the *k-circle formation* is deterministically unsolvable by *oblivious* and *silent* robots. □

Theorem 5.4.2. *If $C(0) \in \mathcal{U}_1$ and the robots have zero knowledge of fixed points, then the k-circle formation problem is deterministically unsolvable by robots equipped with finite color of lights.*

Proof. The idea of this proof is similar to the proof of Theorem 4.3.1. We consider the setting described in the proof of Theorem 4.3.1. Let the symmetric image of r with respect to L is denoted by $\phi(r)$. We assume the following setting:

(i) The scheduler is considered to be SSYNC. In addition, assume that both r and $\phi(r)$ are activated simultaneously.

(ii) All the robots are assumed to move with the same constant speed without any transient stops. Also, assume that both r and $\phi(r)$ would travel the same amount of distance.

As r and $\phi(r)$ run the same algorithm, they would have the same light color. Since the *initial* configuration was symmetric, the robots would not be able to deterministically break the symmetry in this setting. As a consequence, the k-*circle formation* problem is deterministically unsolvable. $\square$

Let $\mathcal{U}_2$ denote the set of all the configurations satisfying the following conditions:

1. k is odd and $C(0) \in \mathcal{F}REFL \cup \mathcal{F}MULT$,
2. $\mathcal{L}' \neq \emptyset$ and number of fixed points on each $L \in \mathcal{L}'$ is even.
3. Either $C(0)$ is asymmetric about each $L \in \mathcal{L}'$ or $C(0)$ is symmetric about an $L \in \mathcal{L}'$ such that $R(0) \cap L \neq \emptyset$.

5.4.2 Algorithm

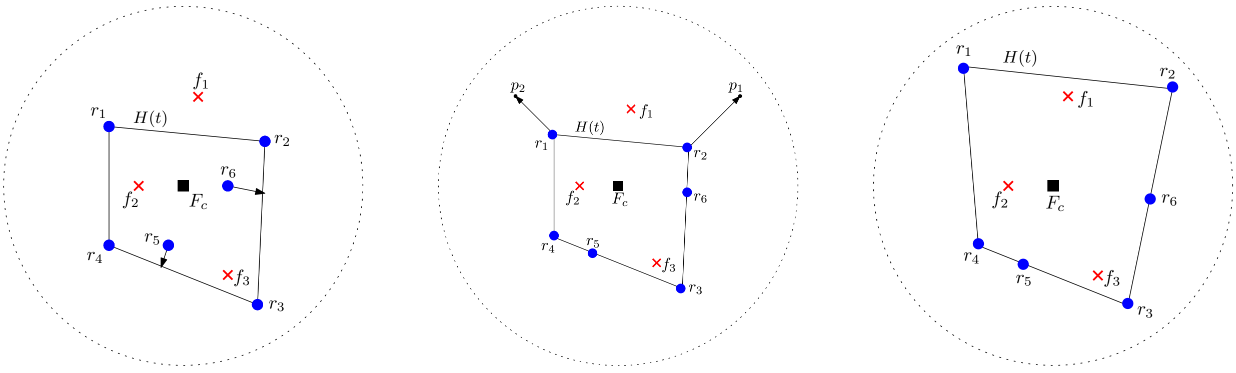


FIGURE 5.17: (**A**) $r_6(t), r_5(t) \in IntH(t)$ and both the robots would move towards boundary of $H(t)$, (**B**) $f_1 \in OutH(t)$ and the vertex robots r_1 and r_2 would move outwards to expand the boundary of $H(t)$, (**C**) $\forall f_i \in F,\ f_i \in IntH(t)$.

We propose a deterministic distributed algorithm that will solve the k-*circle formation* problem for disoriented opaque robots equipped with lights. Our proposed distributed algorithm solves the k-*circle formation* problem for $C(0) \notin \{\mathcal{U}_1 \cup \mathcal{U}_2\}$. Let $r_i(t).light$ denote the color of the light of r_i. COL represents the set of color of the lights. If the robots are *oblivious* and *silent*, then $|COL| = 1$. A robot can observe the color of its own light as well as the color of the other robots visible to it. We assume that $COL = \{Blue, Red\}$ and at $t = 0$, $\forall r_i \in R$, $r_i(0).light = Blue$. An overview of our proposed algorithm $OpaqueAlgorithm2$ is discussed as follows:

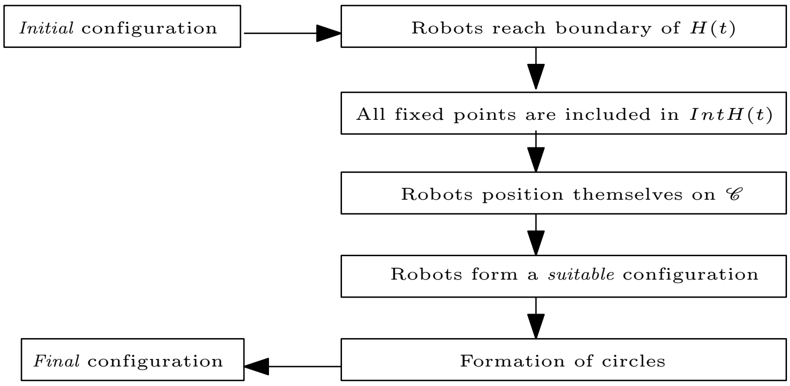
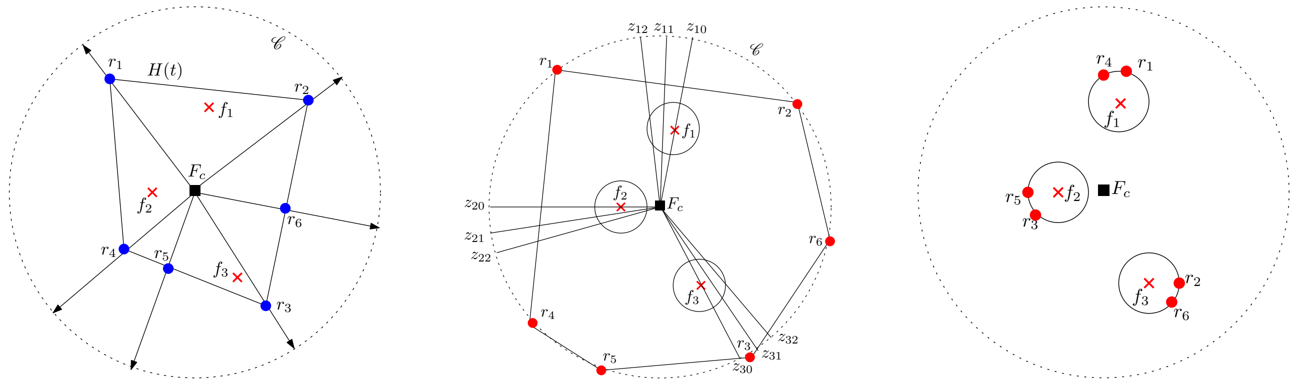


FIGURE 5.18: *OpaqueAlgorithm*2.



(A) (B) (C)

FIGURE 5.19: (**A**) All the robots r_1, r_2, r_3, r_4, r_5 and r_6 would move towards $\mathscr{C}$, (**B**)Each robot identies that all the robots are on $\mathscr{C}$ and changes the light colour to red, (**C**) *final* configuration.

1. All the robots reach the boundary of the convex hull $H(t)$ (Figure 5.17(A)).
2. If $\exists f_i \notin IntH(t)$, then the robots expand the boundary of $H(t)$ to include all the fixed points inside $H(t)$ (Figure 5.17(B) and 5.17(C)).
3. All the robots position themselves on the circle $\mathscr{C}$ (Figure 5.19(A)).
4. The robots re-position themselves on $\mathscr{C}$ so that the configuration transforms into a *suitable* configuration (Figure 5.19(B)).
5. The robots start forming circles around the fixed points (Figure 5.19(C)).

Figure 5.18 represents a diagramatic representation of *OpaqueAlgorithm*2.

Conditions	Descriptions
P_{14}	$\forall r_i,\ r_i.light = Blue$
P_{15}	$\forall r_i,\ r_i.light = Red$
P_{16}	$\forall r \in R,\ VFr(t) = m$
P_{17}	$\forall r_i,\ r_i$ identifies $\mathscr{C}$
P_{18}	$\exists r \in R$ such that $r(t) \in IntH(t)$
P_{19}	$\exists f \in F$ such that $f \in OutH(t)$

Table 5.3: Descriptions of Additional Phase Conditions

5.4.2.1 Phase Conditions during $OpaqueAlgorithm2$

All the phase conditions at any arbitrary point of time $t \geq 0$ during an execution of $OpaqueAlgorithm2$ are defined in Tables 5.2 and 5.3.

5.4.2.2 Phases during $OpaqueAlgorithm2$

We have the following phases during $OpaqueAlgorithm2$:

1. $PHASE1$: A configuration $C(t)$ is said to be in $PHASE1$ if it satisfies $P_{14} \wedge P_{18}$. In this phase, all the robots reach the boundary of $H(t)$. The robots identify this phase by checking whether the light color of all the robots is blue or not (condition P_{14}) and whether there exists a robot $r \in IntH(t)$ or not (condition P_{18}).

2. $PHASE2$: In this phase, the robots expand the boundary of $H(t)$ to include all the fixed points inside $H(t)$. A configuration $C(t)$ is said to be in $PHASE2$ if it satisfies $P_{14} \wedge \neg P_{18} \wedge P_{19}$. The robots identify this phase by checking whether the light color of all the robots is blue or not (condition P_{14}) and whether all the robots lie on the bounary of $H(t)$ or not (condition P_{18}). In addition, the robots check whether there exists a fixed point in $OutH(t)$ or not (condition P_{19}).

3. $PHASE3$: In this phase, all the robots reach the circle $\mathscr{C}$. $C(t)$ is said to be in $PHASE3$ if it satisfies $P_{14} \wedge \neg P_{18} \wedge \neg P_{19} \wedge P_1 \wedge P_{16} \wedge \neg P_7 \wedge \neg P_9$. A robot can identify this phase by checking whether there exists a robot that does not lie on $\mathscr{C}$ or not (condition P_9).

4. $PHASE4$: A configuration $C(t)$ is said to be in $PHASE4$ if it satisfies

$$P_{15} \wedge P_1 \wedge P_{16} \wedge \neg P_6 \wedge (\neg P_{10} \vee \neg P_{11} \vee \neg P_{12} \vee \neg P_{13}) \wedge P_9 \wedge P_8, \ or$$
$$P_{15} \wedge (P_1 \wedge \neg P_{16}) \wedge \neg P_7 \wedge (P_{10} \vee P_{11} \vee P_{12} \vee P_{13}) \wedge \neg P_9 \wedge P_8, \ or$$
$$P_{15} \wedge (\neg P_1 \wedge P_{16}) \wedge \neg P_7 \wedge (P_{10} \vee P_{11} \vee P_{12} \vee P_{13}) \wedge \neg P_9 \wedge P_8, \ or$$
$$P_{15} \wedge (\neg P_1 \wedge \neg P_{16}) \wedge \neg P_7 \wedge (P_{10} \vee P_{11} \vee P_{12} \vee P_{13}) \wedge \neg P_9 \wedge P_8$$

When all the robots lie on $\mathscr{C}$ and $C(t)$ satisfies P_{15} ($\forall r_i, \ r_i.light = Red$), the robots identify this phase by checking whether the configuration is a *suitable* or *partially suitable* configuration or not. However, in order to transform into a *suitable* configuration, a robot might not lie on $\mathscr{C}$ during its movement. In such a case, the robots identify this phase by identifying the condition $(P_{10} \vee P_{11} \vee P_{12} \vee P_{13})$.

5. $PHASE5$: A configuration $C(t)$ is said to be in $PHASE5$ if it satisfies

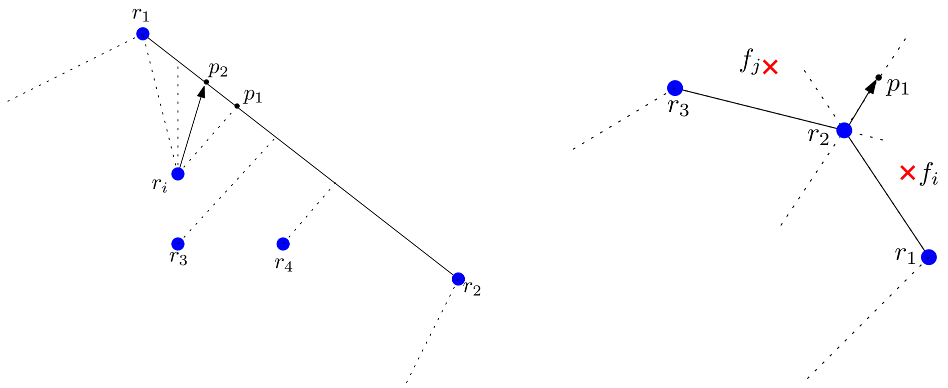
$$P_{15} \wedge P_1 \wedge P_{16} \wedge P_6 \wedge (\neg P_{10} \vee \neg P_{11} \vee \neg P_{12} \vee \neg P_{13}) \wedge P_9 \wedge P_8, \ or$$
$$P_{15} \wedge P_1 \wedge P_{16} \wedge P_6 \wedge (\neg P_{10} \vee \neg P_{11} \vee \neg P_{12} \vee \neg P_{13}) \wedge \neg P_9 \wedge P_8, \ or$$
$$P_{15} \wedge (P_1 \wedge P_{16}) \wedge P_7 \wedge (P_{10} \vee P_{11} \vee P_{12} \vee P_{13}) \wedge \neg P_9 \wedge P_8, \ or$$
$$P_{15} \wedge (P_1 \wedge \neg P_{16}) \wedge P_7 \wedge (P_{10} \vee P_{11} \vee P_{12} \vee P_{13}) \wedge \neg P_9 \wedge P_8, \ or$$
$$P_{15} \wedge (\neg P_1 \wedge P_{16}) \wedge P_7 \wedge (P_{10} \vee P_{11} \vee P_{12} \vee P_{13}) \wedge \neg P_9 \wedge P_8, \ or$$
$$P_{15} \wedge (\neg P_1 \wedge \neg P_{16}) \wedge P_7 \wedge (P_{10} \vee P_{11} \vee P_{12} \vee P_{13}) \wedge \neg P_9 \wedge P_8$$

The robots would identify that the configuration is either *suitable* or *partially suitable*. They identify this phase by checking whether there exists an *unsaturated* fixed point or not (condition P_8).

6. $FINAL$: The $FINAL$ phase is identified by the condition $P_{15} \wedge P_1 \wedge P_{16} \wedge \neg P_8$.

5.4.2.3 Movements during $OpaqueAlgorithm2$

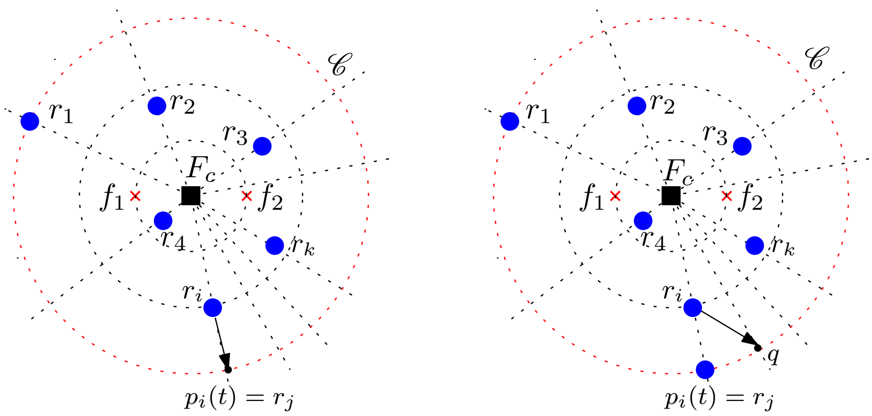
We define the following movements at any arbitrary point of time $t \geq 0$:



(A) (B)

FIGURE 5.20: (**A**) Movement $\boldsymbol{\mathcal{M}_1}$. (**B**) Movement $\boldsymbol{\mathcal{M}_2}$.

1. $\boldsymbol{\mathcal{M}_1}$: This movement is executed when $\boldsymbol{C(t)}$ is in $PHASE1$. Let $r_i \in IntH(t)$ be a robot that lies at the closest distance from the side $\overline{r_1r_2}$ of $H(t)$. If there are multiple such sides, then r selects one of the sides arbitrarily as its *destination line*. It may be the case that there are other robots which are also at the closest distance from $\overline{r_1r_2}$. In such a case, select the one that lies at the closest distance from one of the end points of $\overline{r_1r_2}$. Note that there may be two such robots. Let $p_1 \in \overline{r_1r_2}$ be the point such that $\overline{r_ip_1} \perp \overline{r_1r_2}$. Also, let $r_4 \in \overline{r_1r_2}$ such that $d(r_4, p_1) = \min_{r_i \in \overline{r_1r_2}} d(r_i, p_1)$. Assume that $p_2 \in \overline{r_1r_2}$ be such that $\measuredangle \overline{r_ip_2} r_i \overline{r_ip_1} = \frac{1}{3}\overline{r_ir_4} r \overline{r_ip_1}$. r_i moves towards p_2 along $\overline{rp_2}$ (Figure 5.20(A)).



(A)

FIGURE 5.21: (**A**)-(**B**) Movement $\boldsymbol{\mathcal{M}_3}$.

2. $\boldsymbol{\mathcal{M}_2}$: This movement is executed when $\boldsymbol{C(t)}$ is in $PHASE2$. Asssume that r_1, r_2 and r_3 are the vertices of $H(t)$ such that the sides $\overline{r_1r_2}$ and $\overline{r_2r_3}$ are adjacent. Let $f_i \in OutH(t)$ be a fixed point that lies at the farthest distance from the side $\overline{r_1r_2}$.

Similarly, let $f_j \in OutH(t)$ be such a fixed point from the side $\overline{r_2 r_3}$. Let d_1 represent the distance of f_i from $\overline{r_1 r_2}$. Similarly, let d_2 represent the distance of f_j from $\overline{r_2 r_3}$. Without loss of generality, suppose $d_2 \geq d_1$. Suppose the vertically opposite angle of $\measuredangle r_1 r_2 r_3$ is denoted by ζ. Let B denote the angle bisector of ζ and $p_1 \in B$ such that $d(r_2, p_1) = d_2$. r_2 moves towards p_1 along $\overline{r_2 p_1}$ (Figure 5.20(B)).

3. $\boldsymbol{\mathcal{M}_3}$: This movement is executed when $\boldsymbol{C(t)}$ is in $PHASE3$. For some $r_i \in R$, let $p_i(t)$ be the intersection point between $\mathscr{C}$ and $Ray(F_c, r_i(t))$. Let $r_i \notin \mathscr{C}$ be a robot such that $d(r_i(t), p_i(t)) = \min\limits_{r_j \in R} d(r_j(t), p_j(t))$. If $p_i(t)$ is not a robot position, then r_i starts moving towards $p_i(t)$ along $\overline{p_i(t) r_i(t)}$. Otherwise, let $r_k(t)$ be such that

$$\measuredangle Ray(F_c, r_i(t)) F_c Ray(F_c, r_k(t)) = \min_{r_j \in R} \measuredangle Ray(F_c, r_i(t)) F_c Ray(F_c, r_j(t))$$

Suppose B denotes the ray starting from F_c such that

$$\measuredangle Ray(F_c, r_i(t)) F_c B = \frac{1}{3} \measuredangle Ray(F_c, r_i(t)) F_c Ray(F_c, r_k(t))$$

Assume that q be the intersection point between B and $\mathscr{C}$. Robot r_i starts moving towards q along $\overline{r_i(t) q}$. This movement is similar to the movement $\boldsymbol{M_1}$ during $OpaqueAlgorithm1$.

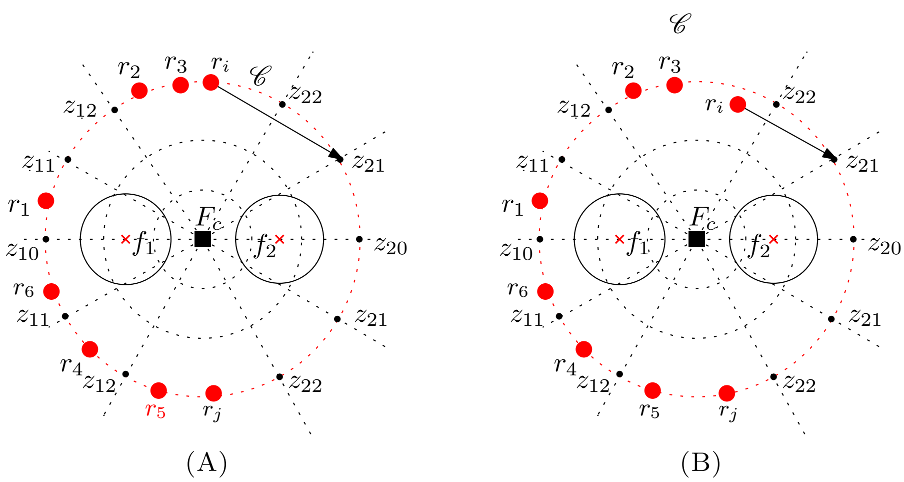


FIGURE 5.22: (**A**)-(**B**) Movement $\boldsymbol{\mathcal{M}_{41}}$.

4. $\boldsymbol{\mathcal{M}_{41}}$: This movement is executed when $C(t)$ is in $PHASE4$ and $C(t) \in \mathcal{F}ASYM$. The robots will form a *suitable* configuration by the execution of movement $\boldsymbol{\mathcal{M}_{41}}$.

Suppose $\mathcal{F}_m \in \mathcal{F}$ is the *master* ray. Let $\mathcal{F}_i \in \mathcal{F}$ be such that $\measuredangle \mathcal{F}_i F_c \mathcal{F}_m = \min\limits_{\mathcal{F}_b \neq \mathcal{F}_m} \mathcal{F}_b F_c \mathcal{F}_m$ measured in the counter clockwise direction and $\exists z_{ip}$ for some $p \in \{1, 2, \ldots, \beta_i k\}$ such that z_{ip} does not contain any robot positions. Assume that $q \in \{1, 2, \ldots, \beta_i k\}$ be the smallest positive integer for which z_{iq} does not contain any robot positions. Suppose r_k denotes the robot such that $\measuredangle \overline{F_c r_k} F_c \overline{F_c z_{iq}} = \min\limits_{r_j \neq r_k} \measuredangle \overline{F_c r_j} F_c \overline{F_c z_{iq}}$ measured in the counter clockwise direction and $d(F_c, r_k) \leq \xi$ (Figure 5.22(A) and 5.22(B)). r_k moves along $\overline{r_k(t) z_{iq}}$ towards z_{iq}.

5. $\boldsymbol{\mathcal{M}_{42}}$: When $C(t)$ is in $PHASE4$ and k is even and $C(t) \in \mathcal{F}REFL \cup \mathcal{F}MULT$, movement $\boldsymbol{\mathcal{M}_{42}}$ is executed. Suppose $\mathcal{F}_m \in \mathcal{F}$ is a *master* ray. Let $\mathcal{F}_i \in \mathcal{F}$ be such that $\measuredangle \mathcal{F}_i F_c \mathcal{F}_m = \min\limits_{\mathcal{F}_b \neq \mathcal{F}_m} \mathcal{F}_b F_c \mathcal{F}_m$ (there can be two such rays) and $\exists z_{ip}$ for some $p \in \{1, 2, \ldots, \dfrac{\beta_i k}{2}\}$ such that z_{ip} does not contain any robot positions. There can be two such robot positions. Let $q \in \{1, 2, \ldots, \dfrac{\beta_i k}{2}\}$ be the smallest positive integer for which z_{iq} does not contain any robot positions. Suppose r_k denotes the robot such that $\measuredangle \overline{F_c r_k} F_c \overline{F_c z_{iq}} = \min\limits_{r_j \neq r_k} \measuredangle \overline{F_c r_j} F_c \overline{F_c z_{iq}}$ measured in the counter clockwise direction and $d(F_c, r_k) \leq \xi$ (Figure 5.22(A) and 5.22(B)). r_k moves along $\overline{r_k(t) z_{iq}}$ towards z_{iq}.

6. $\boldsymbol{\mathcal{M}_{43}}$: This movement is executed when $C(t) \in \mathcal{F}CHIR$ is in $PHASE4$. Since there are multiple *master* rays, movement $\boldsymbol{\mathcal{M}_{43}}$ represents the execution of movement $\boldsymbol{\mathcal{M}_{41}}$ in multiple wedges.

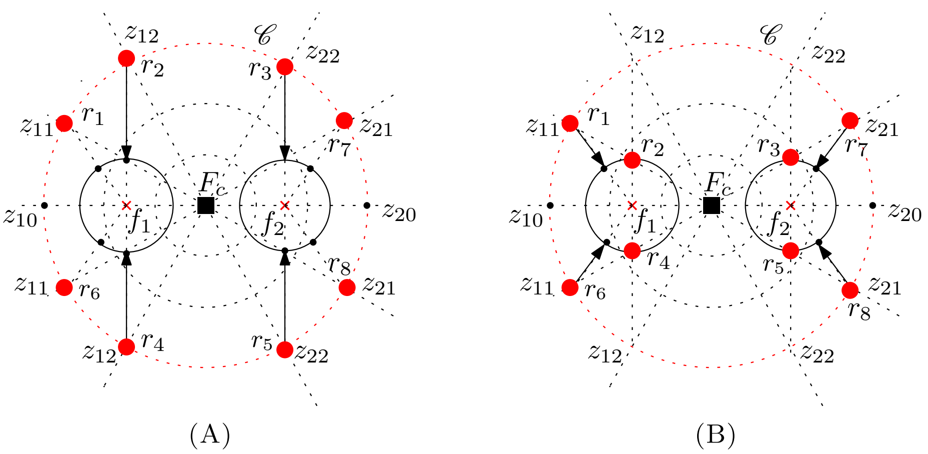


FIGURE 5.23: (**A**)-(**B**) Movement $\boldsymbol{\mathcal{M}_5}$.

7. $\boldsymbol{\mathcal{M}_5}$: This movememt is executed when $C(t)$ is in $PHASE5$. Suppose $\mathcal{F}_m$ is a *master* ray. There can be multiple *master* rays. We have the following cases:

   (a) $C(t)$ satisfies $P_{15} \wedge P_1 \wedge P_{16} \wedge P_6 \wedge (\neg P_{10} \vee \neg P_{11} \vee \neg P_{12} \vee \neg P_{13}) \wedge P_8 \wedge P_9$. Let $f_i \in \mathcal{F}_m$ be the *unsaturated* fixed point that lies at the closest distance from F_c. Since $C(t)$ satisfies P_9, $f_i \in F_m$ is the 1^{st} fixed point according to distance from F_c. Suppose r lies on $z_{i(\beta_i k)}$. Since $C(t)$ satisfies P_9, $\forall r_i \in R$, $VFr_i(t) = m$. As a consequence, r can compute the radius ρ without any conflict. r moves towards $\omega_{i(\beta_i k)}$ along $\overline{\omega_{i(\beta_i k)} z_{i(\beta_i k)}}$ (Figure 5.23(A)).

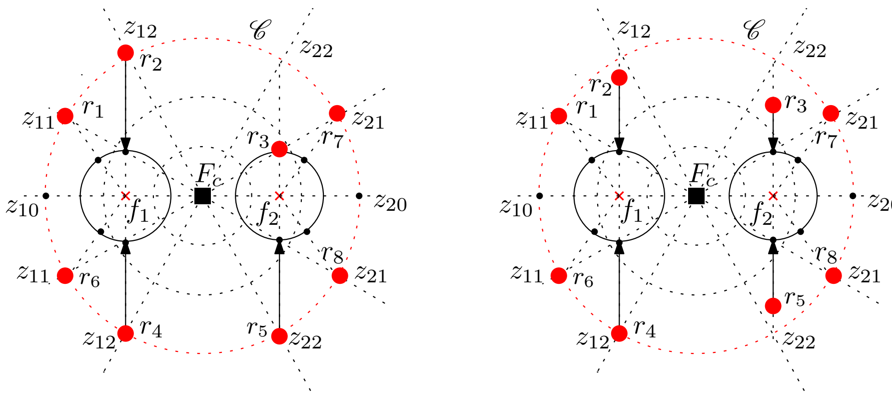


FIGURE 5.24: (A)-(B) Movement $\boldsymbol{\mathcal{M}_5}$.

   (b) $C(t)$ satisfies $P_{15} \wedge P_1 \wedge P_{16} \wedge P_6 \wedge (\neg P_{10} \vee \neg P_{11} \vee \neg P_{12} \vee \neg P_{13}) \wedge P_8 \wedge \neg P_9$. Let $\mathcal{F}_j \in \mathcal{F}$ be such that $\measuredangle \mathcal{F}_j F_c \mathcal{F}_m = \min\limits_{\mathcal{F}_k \in \mathcal{F}} \measuredangle \mathcal{F}_k F_c \mathcal{F}_m$ and it contains an *unsaturated* fixed point. Let p be the smallest positive integer for which ω_{jp} does not contain a robot position. There can be two such positions. Let r be the robot that lies at the closest distance from ω_{jp}. r moves towards ω_{jp} along $\overline{r(t)\omega_{ip}}$ (Figures 5.23(B) and 5.24(A)).

   (c) $C(t)$ satisfies

$$P_{15} \wedge ((P_1 \wedge P_{16}) \vee (P_1 \wedge \neg P_{16}) \vee (\neg P_1 \wedge P_{16}) \vee \neg P_1 \wedge \neg P_{16}))\wedge$$
$$P_7 \wedge P_{10} \wedge P_8 \wedge \neg P_9$$

   Let $\mathcal{F}_j \in \mathcal{F}$ be such that $\measuredangle \mathcal{F}_j F_c \mathcal{F}_m = \min\limits_{\mathcal{F}_k \in \mathcal{F}} \measuredangle \mathcal{F}_k F_c \mathcal{F}_m$ and it contains an *unsaturated* fixed point. Let p be the smallest positive integer for which ω_{jp} does not contain a robot position. Let r be the robot that lies at the closest distance from ω_{jp} such that $d(r(t), F_c) < \xi$. Also, r does not lie on any *saturated* circles and on any ω_{jb} such that $b \in \{1, 2, \ldots, p-1\}$. r moves towards ω_{jp} along $\overline{r(t)\omega_{jp}}$ (Figure 5.24(B)). When there are multiple *master* rays, more than one robots would be moving towards their respective destination points in separate wedges. If r stops before reaching $\omega_{i(\beta_i k)}$, it might be the case that $C(t)$ satisfies $\neg P_{16}$. However, in such a case r can still compute ρ without any conflict by considering all the fixed points in its wedge.

Phases	Movements	Phases after the Movements
$PHASE1$	$\mathcal{M}_1$	$PHASE1$ or $PHASE2$ or $PHASE3$
$PHASE2$	$\mathcal{M}_2$	$PHASE1$ or $PHASE2$ or $PHASE3$
$PHASE3$	$\mathcal{M}_3$	$PHASE1$ or $PHASE3$ or $P_{14} \wedge P_1 \wedge P_{16} \wedge \neg P_6 \wedge (\neg P_{10} \vee \neg P_{11} \vee \neg P_{12} \vee \neg P_{13}) \wedge P_9 \wedge P_8$
$PHASE4$	$\mathcal{M}_{41}$	$PHASE4$ or $PHASE5$
$PHASE4$	$\mathcal{M}_{42}$	$PHASE4$ or $PHASE5$
$PHASE4$	$\mathcal{M}_{43}$	$PHASE4$ or $PHASE5$
$PHASE5$	$\mathcal{M}_5$	$PHASE5$ or $FINAL$

TABLE 5.4: Phase Transitions during *OpaqueAlgorithm2*

ALGORITHM 5.2: *OpaqueAlgorithm2*

Input: $C(t) = (R(t), F)$
if $C(t)$ *is in* $PHASE1$ **then**
    Execute $\mathcal{M}_1$;
else if $C(t)$ *is in* $PHASE2$ **then**
    Execute $\mathcal{M}_2$;
else if $C(t)$ *is in* $PHASE3$ **then**
    Execute $\mathcal{M}_3$;
else if $C(t)$ *satisfies* $P_{14} \wedge P_1 \wedge P_{16} \wedge \neg P_6 \wedge (\neg P_{10} \vee \neg P_{11} \vee \neg P_{12} \vee \neg P_{13}) \wedge P_8 \wedge P_9$ **then**
    if $r_k.light = Blue$ **then**
        r_k changes the color of its light $r_k.light = Red$;
    end
else if $C(t)$ *satisfies* $\neg P_{15} \wedge \neg P_{14} \wedge P_1 \wedge P_{16} \wedge \neg P_6 \wedge (\neg P_{10} \vee \neg P_{11} \vee \neg P_{12} \vee \neg P_{13}) \wedge P_8 \wedge P_9$ **then**
    if $r_k.light = Blue$ **then**
        r_k changes the color of its light $r_k.light = Red$;
    end
else if $C(t)$ *is in* $PHASE4$ **then**
    if $C(t) \in \mathcal{F}ASYM$ **then**
        Execute $\mathcal{M}_{41}$;
    else if $C(t) \in \mathcal{F}REFL \cup \mathcal{F}MULT$ *and* k *is even* **then**
        Execute $\mathcal{M}_{42}$;
    else if $C(t) \in \mathcal{F}CHIR$ **then**
        Execute $\mathcal{M}_{43}$;
    end
else if $C(t)$ *is in* $PHASE5$ **then**
    Execute $\mathcal{M}_5$;
end

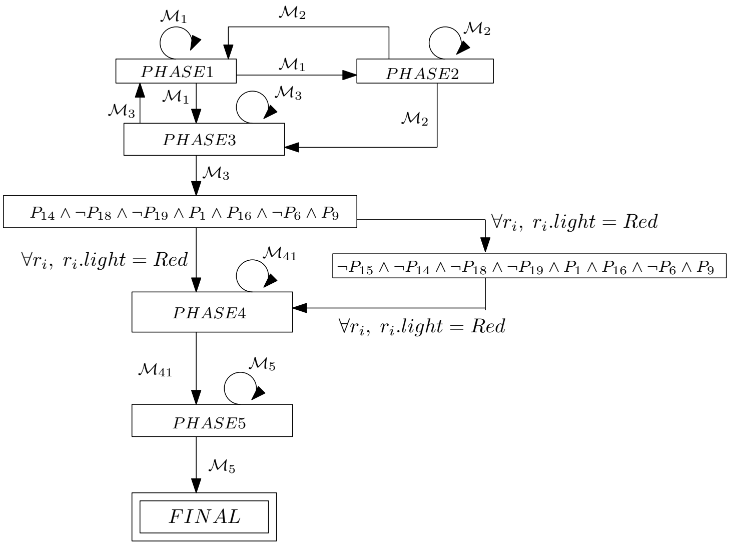


FIGURE 5.25: Phase transitions during $OpaqueAlgorithm2$ when $C(t) \in \mathcal{F}ASYM$.

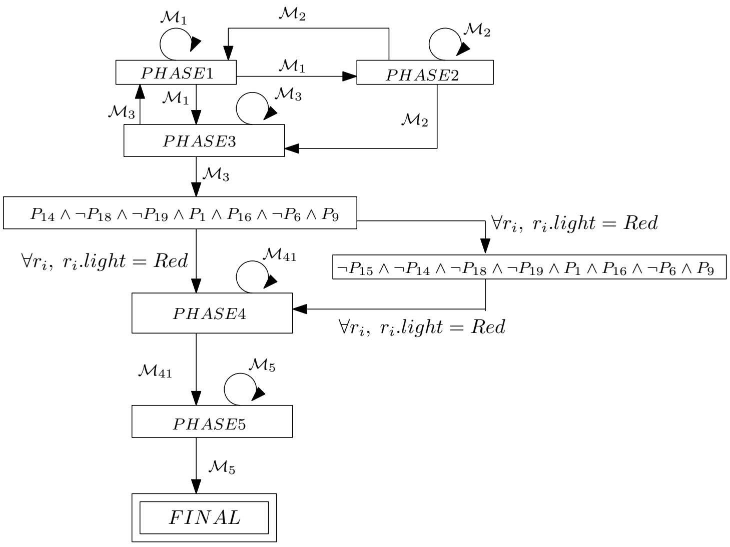


FIGURE 5.26: Phase transitions during $OpaqueAlgorithm2$ when $C(0) \in \mathcal{F}REFL \cup \mathcal{F}MULT$ and k is even.

5.4.2.4 $OpaqueAlgorithm2$

At $t \geq 0$, if $C(t)$ is not a *final* configuration, then an active robot executes algorithm $OpaqueAlgorithm2$. The following cases are to be considered:

1. $\boldsymbol{C(t)}$ is in $PHASE1$. There exists a robot that lies in $IntH(t)$. In this phase, the robots execute movement $\boldsymbol{\mathcal{M}_1}$. By performing movement $\boldsymbol{\mathcal{M}_1}$, all the robots reach the boundary of $H(t)$.

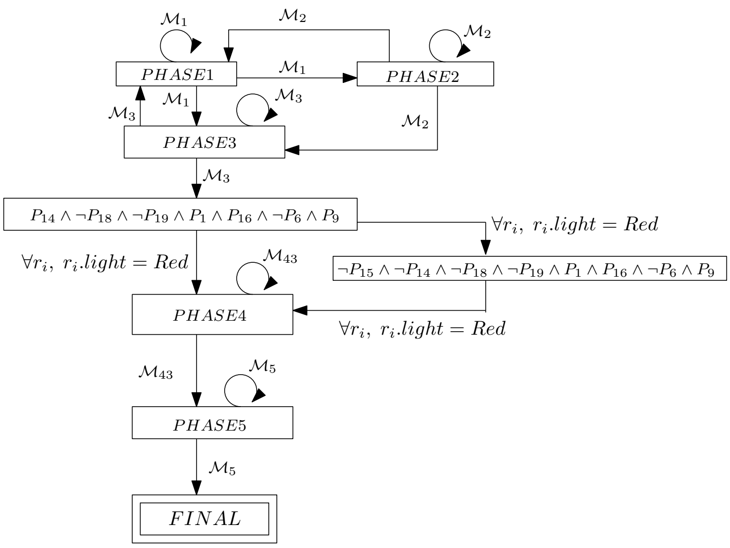


FIGURE 5.27: Phase transitions during $OpaqueAlgorithm2$ when $C(0) \in \mathcal{F}CHIR$.

2. $\boldsymbol{C(t)}$ is in $PHASE2$. There exists a fixed point that lies in $OutH(t)$. In this phase, the robots execute movement $\boldsymbol{\mathcal{M}_2}$. By performing movement $\boldsymbol{\mathcal{M}_2}$, all the robots include all the fixed points inside the boundary of $H(t)$.

3. $\boldsymbol{C(t)}$ is in $PHASE3$. There exists a robot that does not lie on $\mathscr{C}$. All the robots reach the circle $\mathscr{C}$, by the execution of movement $\boldsymbol{\mathcal{M}_3}$.

4. $\boldsymbol{C(t)}$ satisfies one of the following conditions:

$$P_{14} \wedge P_1 \wedge P_{16} \wedge \neg P_6 \wedge (\neg P_{10} \vee \neg P_{11} \vee \neg P_{12} \vee \neg P_{13}) \wedge P_8 \wedge P_9, or$$
$$\neg P_{15} \wedge \neg P_{14} \wedge P_1 \wedge P_{16} \wedge \neg P_6 \wedge (\neg P_{10} \vee \neg P_{11} \vee \neg P_{12} \vee \neg P_{13}) \wedge P_8 \wedge P_9$$

If $r_i.light = Blue$, then r_i changes the color of its light to $r_i.light = Red$.

5. $\boldsymbol{C(t)}$ is in $PHASE4$. The robots re-position themselves on $\mathscr{C}$ to form a *suitable* configuration. In case $C(t) \in \mathcal{F}ASYM$, then movement $\boldsymbol{\mathcal{M}_{41}}$ is executed. When $C(t) \in \mathcal{F}REFL \cup \mathcal{F}MULT$ and k is even, movement $\boldsymbol{\mathcal{M}_{42}}$ is executed. In case $C(t) \in \mathcal{F}CHIR$, then movement $\boldsymbol{\mathcal{M}_{43}}$ is executed.

6. $\boldsymbol{C(t)}$ is in $PHASE5$. In this phase, the robots start forming circles around the fixed points. Movement $\boldsymbol{\mathcal{M}_5}$ is executed.

A summary of the movements during an execution of $OpaqueAlgorithm2$ is presented in Table 5.4. Figures 5.25, 5.26 and 5.27 represent the phase transitions during an execution of $OpaqueAlgorithm2$. The psuedocode of $OpaqueAlgorithm2$ is given in Algorithm 5.2.

5.4.3 Correctness of $OpaqueAlgorithm2$

We need to show that if the *initial* configuration $C(0)$ is *solvable*, then $C(t)$ would remain *solvable* $\forall t > 0$.

Lemma 5.4.3. *If $C(0) \in \mathcal{F}ASYM \cup \mathcal{F}CHIR$, then the configuration $C(t)$ at $t \geq 0$ remains solvable during any execution of $OpaqueAlgorithm2$.*

Proof. When $C(t) \in \mathcal{F}CHIR$, then the robots can make an agreement on a common *chirality*, as discussed in section 4.2.4. Consider the case when $C(t) \in \mathcal{F}ASYM$. Let $f \in F$ be the fixed point that has the minimum configuration view. The direction of $\mathscr{V}(f)$ is globally considered to be the clockwise direction. If there is a tie between robots due to symmetric positions, then such a tie can be broken with respect to *chirality*. Therefore, during any execution of $OpaqueAlgorithm2$, $\forall t \geq 0$, $C(t)$ would remain *solvable*. □

Lemma 5.4.4. *Let $C(0) \in \mathcal{F}REFL \cup \mathcal{F}MULT$ and $C(0) \notin \{\mathcal{U}_1 \cup \mathcal{U}_2\}$. If $C(0)$ is solvable, then the configuration $C(t)$ at $t \geq 0$ remains solvable during any execution of $OpaqueAlgorithm2$.*

Proof. By Observation 3, it follows that if $\mathcal{L}' = \emptyset$, then $C(t)$ would remain *solvable*. Assume that $\mathcal{L}' \neq \emptyset$. If k is even, then from Observation 1 it follows that $C(t)$ would remain *solvable*. Also, $C(t)$ would remain *solvable* when $|F|$ is odd (Observation 2). We need to consider the configurations when k is odd, $|F|$ is even and $\mathcal{L}' \neq \emptyset$. Note that such a configuration belongs to the set $\{\mathcal{U}_1 \cup \mathcal{U}_2\}$ and we have considered that $C(0) \notin \{\mathcal{U}_1 \cup \mathcal{U}_2\}$. Hence, during an execution of $OpaqueAlgorithm2$, if $C(0) \in \mathcal{F}REFL \cup \mathcal{F}MULT$ and *solvable*, then $C(t)$ would remain *solvable* for $t \geq 0$. □

First, we discuss progress during movement $\boldsymbol{\mathcal{M}_1}$. Assume that $C(t)$ is in $PHASE1$. Suppose r_i denotes a *candidate* robot and $q_i(t)$ represents the destination point of r at time t. Let $\mathcal{N}_4(t)$ denote the number of robots which lie in $IntH(t)$. Also, let $g_i(t) = d(r_i(t), q_i(t))$. Define $Z_3(t) = (\mathcal{N}_4(t), g_i(t))$.

Lemma 5.4.5. *Let $C(t)$ be in $PHASE1$. Also, let r_i be a candidate robot and $t' > t$ be an arbitrary point of time at which r_i has completed at least one LCM cycle. Execution of movement $\boldsymbol{\mathcal{M}_1}$ ensures that $g_i(t') + \delta \leq g_i(t)$.*

Proof. Recall that $p_i(t)$ denotes the intersection point between $Ray(F_c, r_i(t))$ and $\mathscr{C}$. We have the following cases:

Case 1. $p_i(t)$ is not a robot position. In this case, $q_i(t) = p_i(t)$ and r_i moves directly towards $p_i(t)$ (Figure 5.13(A)). As r_i moves by at least δ, $g_i(t') + \delta \leq g_i(t)$.

Case 2. $p_i(t)$ is a robot position. r_i computes its destination point according to movement $\boldsymbol{\mathcal{M}_1}$ (Figure 5.13(B)). At t', $d(r_i(t'), q_i(t')) > d(r_i(t'), q_i(t))$. As a consequence, we have $d(r_i(t), q_i(t)) - d(r_i(t'), q_i(t)) > d(r_i(t), q_i(t)) - d(r_i(t'), q_i(t')) \geq \delta$. Thus, $g_i(t') + \delta \leq g_i(t)$.

Hence, an execution of movement $\boldsymbol{\mathcal{M}_1}$ ensures $g_i(t') + \delta \leq g_i(t)$. □

Lemma 5.4.6. *During an execution of movement $\boldsymbol{\mathcal{M}_1}$, let $t' > t$ be an arbitrary point of time at which a candidate robot r_i has completed at least one LCM cycle. An execution of movement $\boldsymbol{\mathcal{M}_1}$ ensures $Z_3(t') < Z_3(t)$.*

Proof. The following cases are to be considered:

Case 1. $q_i(t) = r_i(t')$. *As $\mathcal{N}_4(t') = \mathcal{N}_4(t) - 1$, $Z_3(t') < Z_3(t)$ is ensured.*

Case 2. $q_i(t) \neq r_i(t')$. *Lemma 5.4.5 ensures that $g_i(t') + \delta \leq g_i(t)$.*

Hence, an execution of movement $\boldsymbol{\mathcal{M}_1}$ ensures $Z_3(t') < Z_3(t)$. □

Next, we discuss the *progress* during movement $\boldsymbol{\mathcal{M}_2}$. The aim is to include all the fixed points inside $H(t)$. Let $\mathcal{N}_5(t)$ denote the number of robots which lie in $OutH(t)$. Assume that $C(t)$ is in $PHASE2$. Suppose r_i denotes a *candidate* robot and p_1 denotes its destination point. Let r_j be the robot such that $\overline{r_i(t)r_j(t)}$ is a side of $H(t)$. Assume that $p(t) \in \overline{r_j(t)r_i(t)}$ be such that $\overline{f_j p(t)} \perp \overline{r_j(t)r_i(t)}$. Also, let $g(t) = d(r_i(t), p(t))$. Define $Z_4(t) = (\mathcal{N}_5(t), g(t))$.

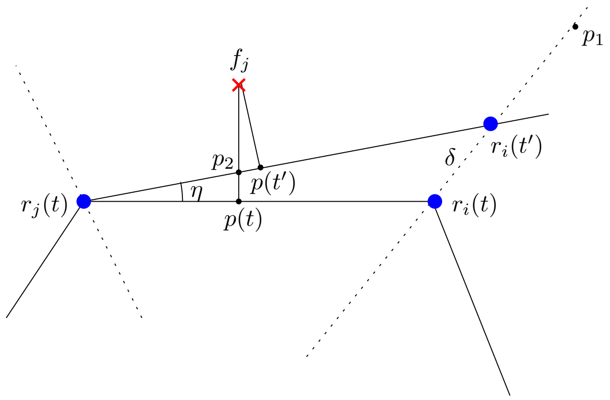


FIGURE 5.28: Progress during movement $\boldsymbol{\mathcal{M}_2}$

Lemma 5.4.7. *Let $C(t)$ be in $PHASE2$. Also, let r_i be a candidate robot and $t' > t$ be an arbitrary point of time at which r_i has completed at least one LCM cycle. Execution of movement $\boldsymbol{\mathcal{M}_2}$ ensures that $Z_4(t') < Z_4(t)$.*

Proof. Suppose $r_j(t)$ denotes the robot such that such that $\overline{r_i(t)r_j(t)}$ is a side of $H(t)$. First, consider the case when $\exists f_k \in F \cap OutH(t)$ such that $f_k \in F \cap IntH(t')$. As $\mathcal{N}_5(t') = \mathcal{N}_5(t) - 1$, $Z_4(t') < Z_4(t)$ is ensured. Otherwise, r_i has moved by at least δ amount not along $\overline{r_j(t)r_i(t)}$. Also, r_i has moved towards $OutH(t)$. As a consequence, $\measuredangle r_i(t)r_j(t)r_i(t') = \eta > 0$ and $d(p_2, p(t)) > 0$. Thus, $g(t) = d(f_j, p(t)) > d(f_j, p(t)) - d(p_2, p(t)) = d(f_j, p_2) > d(f_j, p(t')) = g(t')$ (Figure 5.28). Hence, movement $\boldsymbol{\mathcal{M}_2}$ ensures $Z_4(t') < Z_4(t)$. □

Next, we discuss the *progress* during movement $\boldsymbol{\mathcal{M}_3}$. The goal is to place all the robots on $\mathscr{C}$. Let $\mathcal{N}_6(t)$ denote the number of robots which do not lie on $\mathscr{C}$. Assume that $C(t)$ is in $PHASE3$. Suppose r_i denotes a *candidate* robot and $q_i(t)$ denotes its destination point. Let $g_i(t) = d(r_i(t), q_i(t))$. Define $Z_5(t) = (\mathcal{N}_6(t), g_i(t))$.

Lemma 5.4.8. *During an execution of movement $\boldsymbol{\mathcal{M}_3}$, let $t' > t$ be an arbitrary point of time at which a candidate robot r_i has completed at least one LCM cycle. An execution of movement $\boldsymbol{\mathcal{M}_3}$ ensures $g_i(t') + \delta \leq g_i(t)$.*

Proof. Recall that $p_i(t)$ denotes the intersection point between $Ray(F_c, r_i(t))$ and $\mathscr{C}$. We have the following cases:

Case 1. $p_i(t)$ is not a robot position. In this case, $q_i(t) = p_i(t)$ and r_i moves directly towards $p_i(t)$ (Figure 5.13(A)). As r_i moves by at least δ, $g_i(t') + \delta \leq g_i(t)$.

Case 2. $p_i(t)$ is a robot position. r_i computes its destination point according to movement $\boldsymbol{\mathcal{M}_3}$ (Figure 5.13(B)). At t', $d(r_i(t'), q_i(t')) > d(r_i(t'), q_i(t))$. As a consequence, we have $d(r_i(t), q_i(t)) - d(r_i(t'), q_i(t)) > d(r_i(t), q_i(t)) - d(r_i(t'), q_i(t')) \geq \delta$. Thus, $g_i(t') + \delta \leq g_i(t)$. □

Lemma 5.4.9. *During an execution of movement $\boldsymbol{\mathcal{M}_3}$, let $t' > t$ be an arbitrary point of time at which a candidate robot r_i has completed at least one LCM cycle. An execution of movement $\boldsymbol{\mathcal{M}_3}$ ensures $Z_5(t') < Z_5(t)$.*

Proof. The following cases are to be considered:

Case 1. $q_i(t) = r_i(t')$. *As $\mathcal{N}_6(t') = \mathcal{N}_6(t) - 1$, $Z_5(t') < Z_5(t)$ is ensured.*

Case 2. $q_i(t) \neq r_i(t')$. *Lemma 5.4.8 ensures that $g_i(t') + \delta \leq g_i(t)$.*

Hence, an execution of movement $\boldsymbol{\mathcal{M}_3}$ ensures $Z_5(t') < Z_5(t)$. □

Next, we discuss the *progress* during movement $\boldsymbol{\mathcal{M}_{41}}$. The aim is to form a *suitable* configuration. Assume that $C(t)$ is in $PHASE4$. Let $\mathcal{N}_7(t)$ denote the number of robots which do not lie on any z_{ij} for some $\mathcal{F}_i \in \mathcal{F}$ and $j \in \{1, 2, \ldots, \beta_i k\}$. When k is even and $C(t) \in \mathcal{F}REFL \cup \mathcal{F}MULT$, $\mathcal{N}_7(t)$ denotes the number of robots which do not lie on any z_{ij} for some $\mathcal{F}_i \in \mathcal{F}$ and $j \in \{1, 2, \ldots, \frac{\beta_i k}{2}\}$. Let r_i be a *candidate* robot. $q_i(t)$ denotes the destination point of r_i at time t. Let $g_i(t) = d(r_i(t), q_i(t))$. Define $Z_6(t) = (\mathcal{N}_7(t), g_i(t))$.

Lemma 5.4.10. *During an execution of movement $\boldsymbol{\mathcal{M}_{41}}$, let $t' > t$ be an arbitrary point of time at which a candidate robot r_i has completed at least one LCM cycle. An execution of movement $\boldsymbol{\mathcal{M}_{41}}$ ensures $g_i(t') + \delta \leq g_i(t)$.*

Proof. Recall that $p_i(t)$ denotes the intersection point between $Ray(F_c, r_i(t))$ and $\mathscr{C}$. We have the following cases:

Case 1. $p_i(t)$ is not a robot position. $q_i(t) = p_i(t)$ and r_i moves directly towards $p_i(t)$ (Figure 5.13(A)). Since r_i moves by at least δ, $g_i(t') + \delta \leq g_i(t)$.

Case 2. $p_i(t)$ is a robot position. r_i computes its destination point according to movement $\boldsymbol{\mathcal{M}_{41}}$ (Figure 5.13(B)). At time t', $d(r_i(t'), q_i(t')) > d(r_i(t'), q_i(t))$. We have $d(r_i(t), q_i(t)) - d(r_i(t'), q_i(t)) > d(r_i(t), q_i(t)) - d(r_i(t'), q_i(t')) \geq \delta$. Thus, $g_i(t') + \delta \leq g_i(t)$. □

Lemma 5.4.11. *During an execution of movement* $\boldsymbol{\mathcal{M}}_{41}$*, let* $t' > t$ *be an arbitrary point of time at which a candidate robot* r_i *has completed at least one LCM cycle. An execution of movement* $\boldsymbol{\mathcal{M}}_{41}$ *ensures* $Z_6(t') < Z_6(t)$*.*

Proof. The following cases are to be considered:

Case 1. $q_i(t) = r_i(t')$. *As* $\mathcal{N}_7(t') = \mathcal{N}_7(t) - 1$, $Z_6(t') < Z_6(t)$ *is ensured.*

Case 2. $q_i(t) \neq r_i(t')$. *Lemma 5.4.10 ensures that* $g_i(t') + \delta \leq g_i(t)$*.*

Hence, an execution of movement $\boldsymbol{\mathcal{M}}_{41}$ ensures $Z_6(t') < Z_6(t)$. □

Lemma 5.4.12. *During an execution of movement* $\boldsymbol{\mathcal{M}}_{42}$*, let* $t' > t$ *be an arbitrary point of time at which a candidate robot* r_i *has completed at least one LCM cycle. An execution of movement* $\boldsymbol{\mathcal{M}}_{42}$ *ensures* $Z_6(t') < Z_6(t)$*.*

Proof. During movement $\boldsymbol{\mathcal{M}}_{42}$, $q_i(t') = q_i(t)$ and r_i moves directly towards $q_i(t)$. If $q_i(t) = r_i(t')$, then $\mathcal{N}_7(t') = \mathcal{N}_7(t) - 1$. Thus, $Z_6(t') < Z_6(t)$ is ensured. Consider the case when $q_i(t) \neq r_i(t')$. Since r_i moves by at least δ, $g_i(t') + \delta \leq g_i(t)$. Hence, an execution of movement $\boldsymbol{\mathcal{M}}_{42}$ ensures $Z_6(t') < Z_6(t)$. □

Lemma 5.4.13. *During an execution of movement* $\boldsymbol{\mathcal{M}}_{43}$*, let* $t' > t$ *be an arbitrary point of time at which at least one candidate robot* r_i *has completed at least one LCM cycle. An execution of movement* $\boldsymbol{\mathcal{M}}_{43}$ *ensures* $Z_6(t') < Z_6(t)$*.*

Proof. During an execution of movement $\boldsymbol{\mathcal{M}}_{43}$, movement $\boldsymbol{\mathcal{M}}_{41}$ will be executed in multiple wedges. From Lemma 5.4.11, it follows that $Z_6(t') < Z_6(t)$ will be ensured. □

Next, we discuss *progress* during movement $\boldsymbol{\mathcal{M}}_{5}$. The goal is to form a *final* configuration. Assume that $C(t)$ is in $PHASE5$. Suppose r_i denotes a *candidate* robot of the *target* fixed point $f_j \in \mathcal{F}_k \in \mathcal{F}$. Also, suppose $q_i(t)$ represents the destination point of r at time t. We have $g_i(t) = d(r_i(t), q_i(t))$. Recall that $V_i(t) = (n_k(t), D_j(t), g_i(t))$ where $D_j(t) = k - |C(f_i, \rho) \cap R(t)|$ denotes the deficit of number of robots on $C(f_j, \rho)$ to become *saturated* and $n_k(t)$ denotes the number of *unsaturated* fixed points.

Lemma 5.4.14. *During an execution of movement $\boldsymbol{\mathcal{M}_5}$, let r_i be a candidate robot and $t' > t$ be an arbitrary point of time at which r_i has completed at least one LCM cycle. Execution of movement $\boldsymbol{\mathcal{M}_5}$ ensures that $g_i(t') + \delta \leq g_i(t)$.*

Proof. During movement $\boldsymbol{\mathcal{M}_5}$, r_i moves directly towards $q_i(t)$ (Figure 5.16(A) and 5.16(B)). Since r_i moves by at least δ, $g_i(t') + \delta \leq g_i(t)$. Hence, an execution of movement $\boldsymbol{\mathcal{M}_5}$ ensures $g_i(t') + \delta \leq g_i(t)$. □

Lemma 5.4.15. *During an execution of movement $\boldsymbol{\mathcal{M}_5}$, let $t' > t$ be an arbitrary point of time at which at least one candidate robot r_i has completed at least one LCM cycle. An execution of movement $\boldsymbol{\mathcal{M}_5}$ ensures $V_i(t') < V_i(t)$.*

Proof. The following cases are to be considered:

Case 1. $q_i(t) = r_i(t')$. *If $C(f_j, \rho)$ has exactly k robots, then $n_k(t') = n_k(t) - 1$ ensuring $V_2(t') < V_2(t)$. If $C(f_j, \rho)$ has less than k robots on it, then $D_j(t') = D_j(t) - 1$ ensuring $V_i(t') < V_i(t)$.*

Case 2. $q_i(t) \neq r_i(t')$. *Lemma 5.4.14 ensures that $g_i(t') + \delta \leq g_i(t)$. As a result, $V_i(t') < V_i(t)$ is ensured.* □

Theorem 5.4.16. *Let $C(0) \notin \{\mathcal{U}_1 \cup \mathcal{U}_2\}$ be a given configuration. Execution of algorithm OpaqueAlgorithm2 would solve the k-circle formation problem within finite time under obstructed visibility model.*

Proof. Lemmata 5.4.3 and 5.4.4 ensure that $\forall t \geq 0,\ C(t) \notin \{\mathcal{U}_1 \cup \mathcal{U}_2\}$. At time $t \geq 0$, we have the following cases:

Case 1. $C(t)$ is in $PHASE1$. Movement $\boldsymbol{\mathcal{M}_1}$ is executed. Lemma 5.4.6 ensures that within finite time all the robots would reach the boundary of $H(t)$.

Case 2. $C(t)$ is in $PHASE2$. Movement $\boldsymbol{\mathcal{M}_2}$ is executed. Lemma 5.4.7 guarantees that all the robots would include all the fixed points inside the boundary of $H(t)$.

Case 3. $C(t)$ is in $PHASE3$. There exists a robot that does not lie on $\mathscr{C}$. Movement $\boldsymbol{\mathcal{M}_3}$ is executed. Lemma 5.4.9 ensures that within finite time the robots would reach $\mathscr{C}$.

Case 4. $\boldsymbol{C(t)}$ satisfies one of the following conditions:

$$P_{14} \wedge P_1 \wedge P_{16} \wedge \neg P_6 \wedge (\neg P_{10} \vee \neg P_{11} \vee \neg P_{12} \vee \neg P_{13}) \wedge P_8 \wedge P_9, or$$

$$\neg P_{15} \wedge \neg P_{14} \wedge P_1 \wedge P_{16} \wedge \neg P_6 \wedge (\neg P_{10} \vee \neg P_{11} \vee \neg P_{12} \vee \neg P_{13}) \wedge P_8 \wedge P_9$$

Each robot r_i changes the color of its light to $r_i.light = Red$. Since the scheduler is assumed to be fair, within finite time all the robots will change its light color.

Case 5. $C(t)$ is in $PHASE4$. If $C(t) \in \mathcal{F}ASYM$, then movement $\boldsymbol{\mathcal{M}_{41}}$ is executed. Lemma 5.4.11 ensures that within finite time the robots will form a *suitable* configuration. If $C(t) \in \mathcal{F}REFL \cup \mathcal{F}MULT$ and k is even, movement $\boldsymbol{\mathcal{M}_{42}}$ is executed. Lemma 5.4.12 guarantees that within finite time the robots will form a *suitable* configuration. When $C(t) \in \mathcal{F}CHIR$, then movement $\boldsymbol{\mathcal{M}_{43}}$ is executed. Lemma 5.4.13 ensures that within finite time the robots will form a *suitable* configuration.

Case 6. $C(t)$ is in $PHASE5$. Movement $\boldsymbol{\mathcal{M}_5}$ is executed. Lemma 5.4.15 guarantees that within finite the robots will form a *final* configuration.

Hence, $OpaqueAlgorithm2$ would solve the k-*circle formation* problem within finite time under obstructed visibility model for $C(0) \notin \{\mathcal{U}_1 \cup \mathcal{U}_2\}$. □

For an *initial* configuration $C(0) \in \mathcal{U}_2$, the robots can solve the *mutual visibilty* problem by using light colors. Next, they can deterministically solve the k-*circle formation* problem similar to the idea of $OpaqueAlgorithm2$. The robots can solve the *mutual* visibility problem using two light colors [65]. However, it must be ensured that when solving the *mutual visibilty* problem, the configuration does not fall into the set $\mathcal{U}_1$. Designing such an algorithm for the *mutual visibilty* problem is under investigation.

5.5 Conclusions

In this chapter, we have investigated the k-*circle formation* problem under *obstructed visibility* model. The robots have been assumed to be completely *disoriented*. They operate their LCM cycle under ASYNC scheduler. This chapter studies the k-*circle formation* problem under two different settings based on the visibility of the fixed points:

1. **Complete knowledge of the fixed points:** $\forall r_i \in R,\ VRr_i(t) \leq n$ but $VFr_i(t) = m$. The robots are *oblivious* and *silent*. All the *initial* configurations and values of k for which the k-*circle formation* problem is deterministically unsolvable are characterized. A deterministic distributed algorithm is proposed that solves the k-*circle formation* problem within a finite amount of time.

2. **Zero knowledge of the fixed points:** As a consequence, $\forall r_i \in R,\ VRr_i(t) \leq n$ and $VFr_i(t) \leq m$. It has been shown that the problem is deterministically unsolvable by *oblivious* and *silent* robots. A deterministic distributed algorithm is proposed that solves the k-*circle formation* problem within finite time. The proposed algorithm considers one bit of persistent memory.

Chapter 6

Uniform k-Circle Formation by Fat Robots

Contents

6.1 Overview

In the real world, a robot can not possibly be dimensionless. Czyzowicz et al. [15] studied the *gathering* problem for unit disk robots in the plane. This chapter aims at investigating the *uniform k-circle formation* problem in a more realistic model where the robots have a dimensional extent. They are represented by unit disks in the Euclidean plane.

In order to solve the *uniform k-circle formation* problem, the proposed algorithm must ensure that all the k robots on a circle form a regular k-gon. The assumption on the dimension of a robot introduces additional challenges. A point robot can always pass through the gap between any two points in the plane. It can compute a path in the plane that lies at an infinitesimal distance apart from another robot. In comparison, a fat robot can not do so due to the dimensional extent. A fat robot would act as a physical barrier for the other robots. If a robot is punctiform, then either a robot lies on a circle or it does not. However, for a fat robot, there are two scenarios (e.g., the unit disk intersects the circle or the center of the unit disk lies on the circle) when a robot can be said to lie on a circle. Also, the robots need to compute a suitable radius for the circles so that k robots can be accommodated without any overlapping. Therefore, the solutions proposed in Chapters 3, 4 and 5 would fail to work for fat robots.

6.2 Model and Definitions

The robots are represented by unit disks in the plane. The radius of a unit disk is considered to be one unit distance by all the robots. We assume that the robots have an agreement on the direction of the y-axis. They are *autonomous*, *anonymous*, *homogeneous*, *oblivious* and *silent*. The robots are assumed to be activated under ASYNC scheduler with non-rigid motion.

(1) $R = \{R_1, R_2, \ldots, R_n\}$ denotes the set of all the unit disk robots in the plane. $R_i(t)$ represents the centre of R_i at time t. $R(t) = \{R_1(t), R_2(t), \ldots, R_n(t)\}$ denotes the set of all the robot centers at time t. $U_i(t)$ represents the unit disk centered at $R_i(t)$. Two distinct robots are said to be symmetric if their centers have the same *configuration rank* as defined in section 3.2. $C(t)$ is said to be symmetric if $R(t) \cup F$ is symmetric about the y-axis. The radii of the circles are assumed to be homogeneous. The choice of the value of the radius ρ is arbitrary. However, it must be ensured that k robots can be accommodated on a circle without any overlapping. If $R_i(t)$ lies on a circle, then R_i is said to lie on that circle.

(2) All the configurations can be partitioned into the following disjoint classes:

   (a) $\mathcal{J}_1-$ F is asymmetric about the y-axis (Figure 6.1(A)).

   (b) $\mathcal{J}_2-$ F is symmetric about the y-axis and $F_y = \emptyset$ (Figure 6.1(B)).

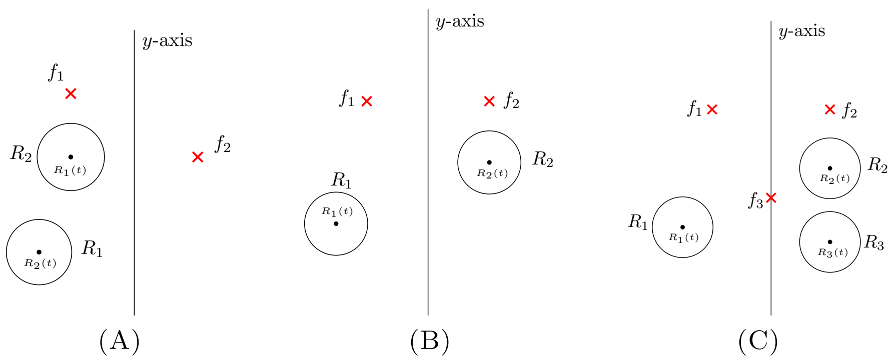


FIGURE 6.1: Small black circles represent the center of a robot. (**A**) $\mathcal{J}_1$-configuration. (**B**) $\mathcal{J}_2$-configuration. (**C**) $\mathcal{J}_3$-configuration.

(c) $\mathcal{J}_3-$ F is symmetric about the y-axis and $F_y \neq \emptyset$ (Figure 6.1(C)).

Since the partition is based upon fixed points, the robots can easily identify the class of a configuration by observing the fixed points.

(3) **Half-planes:** Let F_i denote the set of fixed points in $\mathcal{H}_i \in \{\mathcal{H}_1, \mathcal{H}_2\}$. $C_i(t) = (R(t), F_i)$ represents the part of the configuration consisting of $R(t) \cup F_i$, where $i \in \{1, 2\}$. $C_3(t) = (R(t), F_y)$ denotes the part of the configuration consisting of $R(t) \cup F_y$. In $\mathcal{H}_1$, the positive x-axis direction is considered along the perpendicular direction away from the y-axis. Similarly, the positive x-axis direction is the perpendicular direction away from the y-axis in $\mathcal{H}_2$.

6.2.1 The Uniform k-Circle Formation Problem

A configuration $C(t)$ for some $t \geq 0$ is said to be a *final* configuration, if it satisfies the following conditions:

i) $\forall R_i \in R,\ R_i(t) \in C(f_j, \rho)$ for some $f_j \in F$,

ii) $|C(f_i, \rho) \cap R(t)| = k,\ \forall f_i \in F$, and

iii) All the k robots which lie on the same circle form a regular k-gon.

To solve the *uniform k-circle formation* problem, starting from a given *initial* configuration the robots need to reach and remain in a *final* configuration.

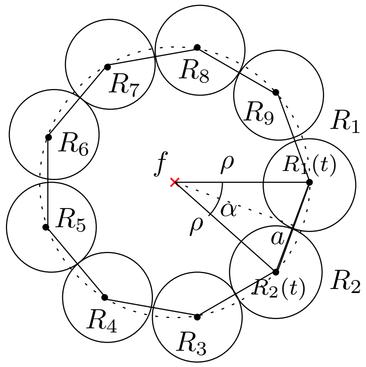


FIGURE 6.2: The minimum radius required to form a circle containing exactly k robots.

6.2.2 Radii of the Circles

Let $\rho > 0$ denote the radius of a circle. The minimum radius for a circle for fat robots is achieved when there are no gaps between any two adjacent robots on the circle. When $k = 1$, we assume that the radius is one unit. For $k > 1$, let $\alpha = \frac{2\pi}{k}$ and a be the mid-point of the line segment $\overline{R_1(t)R_2(t)}$ (Figure 6.2).

$$\text{We have, } \sin\frac{\alpha}{2} = \frac{\overline{R_2(t)a}}{\overline{R_2(t)f}} = \frac{1}{\rho} \implies \rho = \frac{1}{\sin\frac{\alpha}{2}}.$$

The choice of ρ would ensure that all the k robots which lie on the same circle would form a regular k-gon. Let $\mathcal{P}_i$ denote the regular k-gon centered at $f_i \in F$ with $\{\beta_1, \beta_2, \ldots, \beta_k\}$ as the set of vertices such that $d(\beta_i, \beta_j) = 2$, where $i \in \{1, 2, \ldots, k\}$ and $j = i + 1\ mod(k)$. We assume that the minimum distance between any two fixed points is greater than or equal to $2(\rho + 1)$. This would always ensure that even if two adjacent k-gons are rotated, the formation of disjoint circles without any overlapping of robots would be guaranteed.

Definition 6.2.1. *If k is some odd integer and $C(t) \in \mathcal{I}_3$, then $C(t)$ is said to be an unsafe configuration. A pivot position is defined for an unsafe configuration. Suppose $f \in F_y$ be the topmost fixed point. Let $\rho_1 \in \mathcal{H}_1$ be the point such that $\rho_1 \in C(f, \eta)$ and $x(\rho_1) - x(f) = \frac{1}{2}$ unit distance. Similarly, let ρ_2 be such a point in $\mathcal{H}_2$. The points ρ_1 and ρ_2 are said to be pivot positions. A robot placed on a pivot position is said to be a pivot robot.*

6.3 Impossibility Result

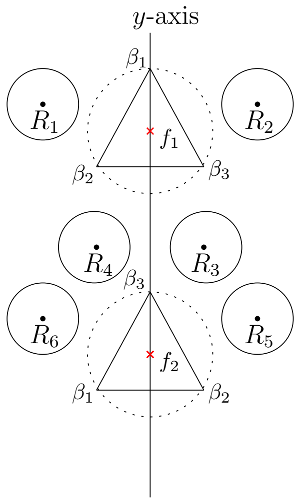


FIGURE 6.3: An example of a configuration satisfying the impossibility criteria (Theorem 6.3.1).

Theorem 6.3.1. *Let $C(0)$ be a given initial configuration. If k is some odd integer and $C(0) \in \mathcal{J}_3$, such that the following conditions hold:*

i) $R(0)$ is symmetric about the y-axis, and

ii) $R_y(0) = \emptyset$,

then the uniform k-circle formation problem is deterministically unsolvable.

Proof. If possible, let algorithm $\mathcal{A}$ solve the *uniform k-circle formation* problem. Suppose $\phi(R_i)$ denotes the symmetric image of R_i. Assume that the robots are activated under a semi-synchronous scheduler. Also, assume that both R_i and $\phi(R_i)$ are activated simultaneously. All the robots are assumed to move with the same constant speed without any transient stops. Consider that the distance traveled by R_i is the same as that by $\phi(R_i)$. Assume that both R_i and $\phi(R_i)$ have opposite notions of positive x-axis direction. They would have identical configuration views. Since the robots are homogeneous, their destinations and the corresponding paths for movements would be mirror images. Since we started with a symmetric configuration, no algorithm can deterministically break the symmetry. Let $f_i \in F_y$. Since the configuration is symmetric, $\mathcal{P}_i$ must be symmetric around the y-axis. As k is odd, $\mathcal{P}_i$ must contain a robot position on the y-axis. Since

$R_y(0) = \emptyset$, having a robot R_i moved to the y-axis would mean moving $\phi(R_i)$ to the same point. However, overlapping of the robots is not allowed. Hence, the *uniform k-circle formation* problem is deterministically unsolvable. □

Let $\mathcal{U}_4$ denote the set of all the *initial* configurations which satisfy the conditions stated in Theorem 6.3.1 (Figure 6.3).

6.4 Algorithm

Theorem 6.3.1 provides a sufficient condition for an *initial* configuration for which the *uniform k-circle formation* is deterministially *unsolvable*. In this section, a deterministic distributed algorithm $AlgorithmFatRobot$ is proposed that solves the *uniform k-circle formation* problem for an *initial* configuration $C(0) \notin \mathcal{U}_4$. An active robot would execute the proposed algorithm $AlgorithmFatRobot$ unless the current configuration is a *final* configuration.

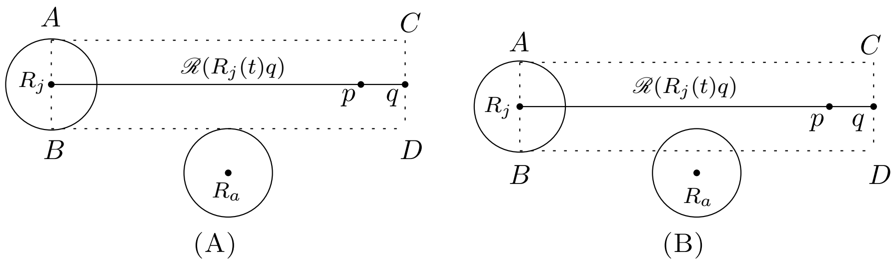


FIGURE 6.4: (**A**) $\mathscr{R}(R_j(t)q)$ is empty. (**B**) $\mathscr{R}(R_j(t)q)$ is non-empty

Definition 6.4.1. Let p be the destination point computed by R_j. Let q be the point such that $p \in \overline{R_j(t)q}$ and $d(p, q) = 1$. The rectangular strip $ABCD$ (Figures 6.4(A) and 6.4(B)) between $R_j(t)$ and q having width of two units is denoted by $\mathscr{R}(R_j(t)q)$. If $\nexists R_i \in R$ such that $U_i(t)$ intersects $\mathscr{R}(R_j(t)q)$, then $\mathscr{R}(R_j(t)q)$ is said to be empty (Figure 6.4(A)). Otherwise, it is said to be non-empty (Figure 6.4(B)). If $\mathscr{R}(R_j(t)q)$ is empty, then R_j is said to have a free path for movement towards p.

During the execution of $AlgorithmFatRobot$, the robots decide their strategy depending on the class of the *initial* configuration. An overview of algorithm $AlgorithmFatRobot$ is described below:

1. If $C(0) \in \mathcal{J}_1$ or $C(0)$ is an *unsafe* configuration, then the robots agree on the positive direction of the x-axis. In case, $C(0) \in \mathcal{J}_1$ the x-axis agreement is based on fixed points only. If $C(0)$ is an *unsafe* configuration, then the robots would execute the procedure $PivotSelection$ (Section 6.4.2) by which a *pivot* robot would be selected and placed on a *pivot* position. The *pivot* robot would remain fixed at the *pivot* position. The *pivot* position is selected by ensuring that the configuration would remain asymmetric once the *pivot* position is placed. In this case, the x-axis agreement is based on the *pivot* position.

2. The robots would execute the $CircleFormation$ (Section 6.4.3) for a unique fixed point (or for two fixed points when the robots do not have a global x-axis agreement). Such a fixed point is said to be a *target* fixed point. $CircleFormation$ is the procedure by which the robots would accomplish the formation of circles.

During the execution of the $AlgorithmFatRobot$, the robots would move downwards by the execution of procedure $DownwardMovement$ (Section 6.4.1). Since the robots have a dimensional extent, a robot can start to move towards its destination point only if it has a free path for movement.

6.4.1 $DownwardMovement$

$DownwardMovement$ is the procedure in $AlgorithmFatRobot$ by which the robots would move downwards. Assume that R_j has been selected for downward movement by one unit. R_j would move one unit vertically downwards by the execution of the procedure $DownwardMovement$. However, if the *pivot* robot falls in its path, then it can not move downwards. In such a case, it would move one unit horizontally. First, some new notations and definitions are introduced.

Suppose $VL_j(t)$ denotes the vertical line passing through $R_j(t)$. Let $p_j(t) \in VL_j(t)$ be the point such that $\gamma(R_j(t)) > \gamma(p_j(t))$ and $d(R_j(t), p_j(t)) = 2$. Define the set $M_j(t)$ as follows:

1. **Base case:** If $\mathscr{R}(R_j(t)p_j(t))$ is empty, then $M_j(t) = \emptyset$. Else, $M_j(t) = \{R_a \mid U_a(t) \textit{ intersects } \mathscr{R}(R_j(t)p_j(t))\}$.

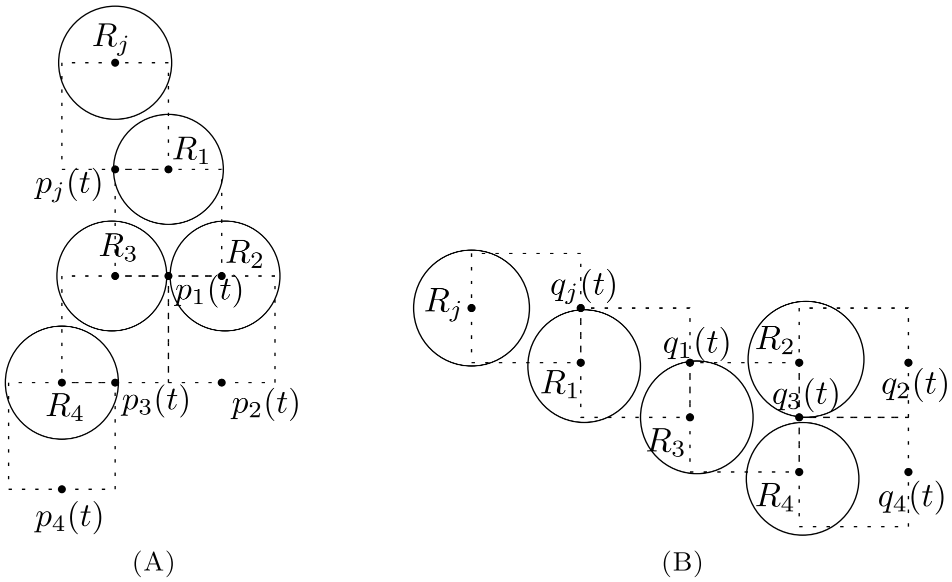


FIGURE 6.5: (**A**) $M_j(t) = \{R_1, R_2, R_3, R_4\}$. As $U_1(t)$ intersects $\mathscr{R}(R_j(t)p_j(t))$, $R_1 \in M_j(t)$. Also, $U_2(t)$ and $U_3(t)$ intersect $\mathscr{R}(R_1(t)p_1(t))$, $R_2 \in M_j(t)$ and $R_3 \in M_j(t)$. $U_4(t)$ intersects $\mathscr{R}(R_3(t)p_3(t))$ and $R_4 \in M_j(t)$. (**B**) $N_j(t) = \{R_1, R_2, R_3, R_4\}$. As $U_1(t)$ intersects $\mathscr{R}(R_j(t)q_j(t))$, $R_1 \in N_j(t)$. $U_3(t)$ intersects $\mathscr{R}(R_1(t)q_1(t))$ and $R_3 \in N_j(t)$. Also, $U_2(t)$ and $U_4(t)$ intersect $\mathscr{R}(R_3(t)q_1(t))$, $R_2 \in N_j(t)$ and $R_4 \in N_j(t)$.

2. **Constructor case:**

   $M_j(t) = M_j(t) \cup \{R_b \mid U_b(t)\ intersects\ \mathscr{R}(R_i(t)p_i(t))\ for\ some\ R_i \in M_j(t)\}.$

The set $M_j(t)$ contains all the robots that must be moved downwards before R_j can move one unit vertically downwards (Figure 6.5(A)). Let $HL_j(t)$ denote the horizontal line passing through $R_j(t)$. In case, R_j is selected for horizontal movement, let $q_j(t) \in HL_j(t)$ be the point such that $\gamma(R_j(t)) < \gamma(q_j(t))$ and $d(R_j(t), q_j(t)) = 2$. Define the set $N_j(t)$ as follows:

1. **Base case:** If $\mathscr{R}(R_j(t)q_j(t))$ is empty, then $N_j(t) = \emptyset$. Else, $N_j(t) = \{R_a \mid U_a(t)\ intersects\ \mathscr{R}(R_j(t)q_j(t))\}$.

2. **Constructor case:**

   $N_j(t) = N_j(t) \cup \{R_b \mid U_b(t)\ intersects\ \mathscr{R}(R_i(t)q_i(t))\ for\ some\ R_i \in N_j(t)\}.$

The set $N_j(t)$ contains all the robots that must be moved horizontally so that $\mathscr{R}(R_j(t)q_j(t))$ becomes empty (Figure 6.5(B)). During the execution of $DownwardMovement(R_j)$, the following cases are to be considered:

1. $M_j(t) = \emptyset$. R_j would start moving towards $p_j(t)$ along $\overline{R_j(t)p_j(t)}$.

2. $M_j(t) \neq \emptyset$. There are two possible cases:

   (a) $M_j(t)$ contains the *pivot* robot. If $N_j(t) = \emptyset$, then R_j moves towards $q_j(t)$ along $\overline{R_j(t)q_j(t)}$. Otherwise, let $R_a \in N_j(t)$ be such that $d(R_j(t), R_a(t)) = \max\limits_{R_k \in N_j(t)} d(R_j(t), R_k(t))$. R_a moves towards $q_a(t)$ along $\overline{R_a(t)q_a(t)}$. There may be multiple such robots which would perform the required movement.

   (b) $M_j(t)$ does not contain the *pivot* robot. Let $R_a \in M_j(t)$ be such that $\gamma(R_a(t)) \leq \min\limits_{R_k \in M_j(t)} \gamma(R_k(t))$. R_a moves towards $p_a(t)$ along $\overline{R_a(t)p_a(t)}$. If there are multiple such robots, then all of them would perform the required vertical movement.

6.4.2 *PivotSelection*

PivotSelection is the procedure in *AlgorithmFatRobot* by which a robot would be placed at one of the *pivot* positions. The robots would execute *PivotSelection* unless one of the *pivot* position is occupied by a robot. Let R_a be the robot that lies at the closest distance from *pivot* position ρ_1. If there are multiple such robots, then select the topmost one. In case there is a tie, select the one closest to the y-axis. Similarly, let R_b be the robot that lies at the closest distance from ρ_2. The following cases are to be considered:

1. $d(R_a(t), \rho_1) \neq d(R_b(t), \rho_2)$. Without loss of generality, let $d(R_a(t), \rho_1) < d(R_b(t), \rho_2)$. The robot R_a would start moving towards ρ_1 along $\overline{R_a(t)\rho_1}$.

2. $d(R_a(t), \rho_1) = d(R_b(t), \rho_2)$ and $R_y(t) = \emptyset$. Since $C(t) \notin \mathcal{U}_4$, it must be asymmetric about the y-axis. Let R_l be the topmost asymmetric robot. If there are multiple such robot then select the one which lies at the closest distance from the y-axis. Without loss of generality, assume that $R_l \in \mathcal{H}_1$. The robot R_a would start moving towards ρ_1 along $\overline{R_a(t)\rho_1}$.

3. $d(R_a(t), \rho_1) = d(R_b(t), \rho_2)$ and $R_y(t) \neq \emptyset$. There are two possible cases:

   (i) $C(t)$ is asymmetric. In this case, the robots will perform the required actions similarly as in case 2.

   (ii) $C(t)$ is symmetric. First, consider the case when $\exists R_a \in R_y(t)$ that can be moved horizontally half a unit away from the y-axis. If there are multiple such robots, select the topmost one. R_a would move horizontally half a unit away from the y-axis. Next, consider the case when there are no such robots on the y-axis. Let $R_a \in R_y(t)$ be the robot that has the minimum rank. $DownwardMovement(R_a)$ would be executed.

6.4.3 $CircleFormation$

$CircleFormation$ is the procedure in $AlgorithmFatRobot$ by which the robots would accomplish the formation of a circle. Let f_i be a *target* fixed point. The following additional notations and terminologies are introduced:

1. $A_i(t) = \{R_j \mid R_j(t) \in C(f_a, \rho)\ where\ f_a \in F\ be\ such\ that\ \gamma(f_a) \geq \gamma(f_i)\}$.
2. $f_l \in F$ denotes a fixed point such that $\gamma(f_l) \leq \gamma(f_j),\ \forall f_j \in F$.
3. R_j is said to satisfy condition $C1$ if it is not the *pivot* robot.
4. R_j is said to satisfy condition $C2$ if $y(R_j(t)) \geq y(f_l) - (\rho + 1)$.
5. $B_i(t) = \{R_j \mid R_j \notin A_i(t)\ and\ it\ satisfies\ C1\ and\ C2\}$.
6. Let $\beta_a \in \mathcal{P}_i$ be the empty vertex that has the highest rank in A_{max}. Assume that R_j has been selected for moving towards β_a. If $\mathscr{R}(R_j(t), \beta_a)$ is non-empty, then let $a_j \in HL_j(t)$ denote the point that lies at the closest distance from R_j such that $\mathscr{R}(R_j(t) = a_j, \beta_a)$ is empty.

Definition 6.4.2. *$C(f_i, \rho)$ is said to be a perfect circle, if the following conditions hold:*

1. *If $R_j(t) \in C(f_i, \rho)$, then $R_j(t) = \beta_k$ for some $\beta_k \in \mathcal{P}_i$.*
2. *If $R_j(t) \in C(f_i, \rho)$ and $R_j(t) = \beta_k \in \mathcal{P}_i$, then $\nexists\ \beta_j \in \mathcal{P}_i$ such that $\gamma(\beta_k) < \gamma(\beta_j)$ and β_j is not occupied.*

If $R_j \in C(f_i, \rho)$ be such that one of the above conditions is not satisfied, then it is said to be an imperfect robot. A circle is said to be imperfect if it contains an imperfect robot.

During an execution of $CircleFormation(C(t), f_i)$, an active robot R_i considers the following cases:

1. The robots have global x-axis agreement or $f_i \notin F_y$. The following cases are to be considered:

   (a) $|B_i(t)| > 1$. Let $R_j \in B_i(t)$ be the robot that has the maximum rank. The robots would execute $DownwardMovement(R_j)$.

   (b) $|B_i(t)| = 1$. Let $\beta_c \in \mathcal{P}_i$ be the empty vertex that has the maximum rank. Let $R_j \in B_i(t)$. If $\mathscr{R}(R_j(t)\beta_c)$ is empty, then R_j would start moving towards β_c. Otherwise, $DownwardMovement(R_j)$ would be executed.

   (c) $|B_i(t)| = 0$ and $C(f_i, \rho)$ is *imperfect*. Let $\beta_c \in \mathcal{P}_i$ be the empty vertex that has the maximum rank. Let $R_j \in C(f_i, \rho)$ be such that $\gamma(R_j(t)) < \gamma(\beta_c)$ and $d(R_j(t), \beta_c)$ is minimum. If there is a tie, select the one that has the maximum rank. R_j would start moving towards β_c along $\overline{R_j(t)\beta_c}$.

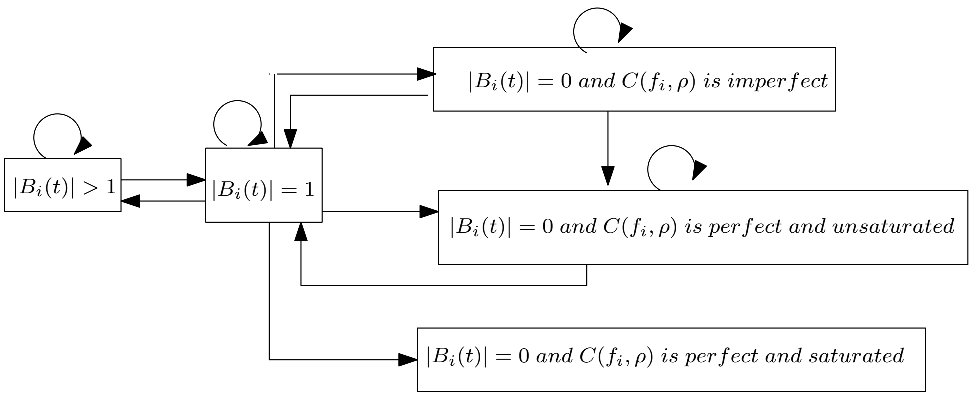


FIGURE 6.6: A flow chart showing the transformations among the various cases during an execution of $CircleFormation$ when the robots have a global x-axis agreement or the *target* fixed point does not belong to F_y.

   (d) $|B_i(t)| = 0$ and $C(f_i, \rho)$ is *perfect*. Let $\beta_c \in \mathcal{P}_i$ be the empty vertex that has the maximum rank. Let $R_j \in R(t) \setminus A_i(t)$ be such that $d(R_j(t), \beta_c)$ is minimum. If there is a tie, select the one that has the maximum rank. If $\mathscr{R}(R_j(t)\beta_c)$ is empty, then R_j would start moving towards β_c along $\overline{R_j(t)\beta_c}$. Else, R_j would start moving towards a_j along $\overline{R_j(t)a_j}$.

Figure 6.6 depicts the transformations among the above-mentioned cases.

2. The robots do not have any global x-axis agreement and $f_i \in F_y$. The following cases are to be considered:

   (a) $|B_i(t)| > 2$. Let $R_j \in B_i(t)$ be the robot that has the maximum rank. The robots would execute $DownwardMovement(R_j)$.

   (b) $0 < |B_i(t)| \leq 2$. Let $\beta_a \in \mathcal{H}_1$ be the empty vertex of $\mathcal{P}_i$ that has the highest rank. Similarly, let β_b be such a vertex in $\mathcal{H}_2$. Assume that R_j and R_k are the robots that are at the closest distance from β_a and β_b, respectively. If $\mathscr{R}(R_j(t)\beta_a)$ is empty, then R_j would start moving towards β_a. Otherwise, $DownwardMovement(R_j)$ would be executed. Similarly, if $\mathscr{R}(R_k(t)\beta_b)$ is empty, then R_k would start moving towards β_b. Otherwise, $DownwardMovement(R_k)$ would be executed.

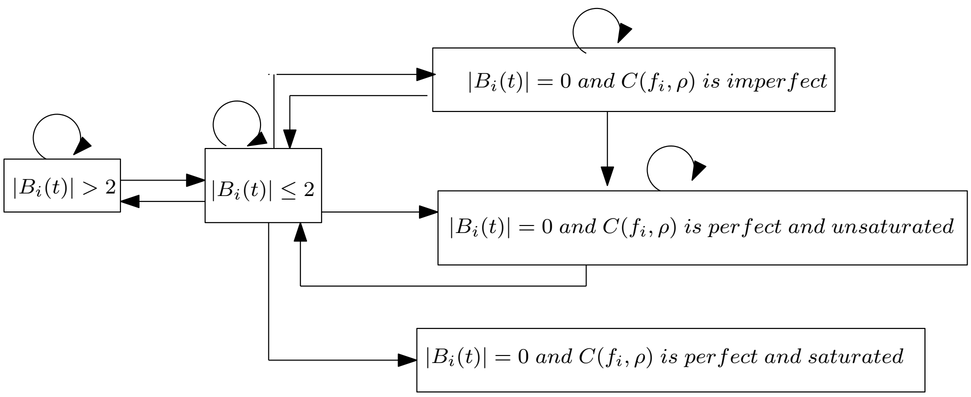


FIGURE 6.7: A flow chart showing the transformations among the various cases during an execution of *CircleFormation* when the robots do not have any global x-axis agreement and the *target* fixed point belongs to F_y.

   (c) $|B_i(t)| = 0$ and $C(f_i, \rho)$ is *imperfect*. Let $\beta_a \in \mathcal{H}_1$ be the empty vertex of $\mathcal{P}_i$ that has the highest rank. Similarly, let β_b be such a vertex in $\mathcal{H}_2$. Let $R_j \in C(f_i, \rho)$ be such that $\gamma(R_j(t)) < \gamma(\beta_a)$ and $d(R_j(t), \beta_a)$ is minimum. If there is a tie, select the one that has the maximum rank. Let R_k be such an robot for the vertex β_b. R_j would start moving towards β_a along $\overline{R_j(t)\beta_a}$. Similarly, R_k would start moving towards β_b along $\overline{R_k(t)\beta_b}$.

   (d) $|B_i(t)| = 0$ and $C(f_i, \rho)$ is *perfect*. Let $\beta_a \in \mathcal{H}_1$ be the empty vertex of $\mathcal{P}_i$ that has the highest rank. Similarly, let β_b be such a vertex in $\mathcal{H}_2$. Let $R_j \in R(t) \setminus A_i(t)$ such that $d(R_j(t), \beta_a)$ is minimum. If there is a tie, select the one that has the maximum rank. Assume that R_k be such a robot for β_b.

If $\mathscr{R}(R_j(t)\beta_a)$ is empty, then R_j would start moving towards β_a along $\overline{R_j(t)\beta_a}$. Otherwise, R_j would start moving towards a_j along $\overline{R_j(t)a_j}$. Similarly, R_k would select its destination point and start moving towards it.

If $R_j = R_k$ for any of the above cases, then R_j would select the *target* fixed point that lies at the closest distance from it. If there is a tie, it would select one of the *target* fixed point arbitrarily. Figure 6.7 depicts the transformations among the above mentioned cases.

6.4.4 *AlgorithmFatRobot*

AlgorithmFatRobot is the proposed deterministic distributed algorithm that solves the *uniform k-circle formation* problem within finite time. The pseudocode of algorithm *AlgorithmFatRobot* is presented in Algorithm 6.1. During an execution of algorithm *AlgorithmFatRobot*, the robots would form a circle centered at a *target* fixed point by the procedure *CircleFormation*. Let R_j be an active robot at time $t \geq 0$. If $C(t)$ is identified to be a *non-final* configuration, then *AlgorithmFatRobot*$(C(t))$ would be executed. Consider the following cases:

ALGORITHM 6.1: *AlgorithmFatRobot*

Input: $C(t) = (R(t), F)$
if $C(t) \in \mathcal{J}_1$ **then**
    Let f_j be the *target* fixed point;
    Execute $CircleFormation(C(t), f_j)$;
else if $C(t) \in \mathcal{J}_2$ **then**
    Let $f_j \in C_1(t)$ and $f_b \in C_2(t)$ be the *target* fixed points;
    Execute $CircleFormation(C_1(t), f_j)$ and $CircleFormation(C_2(t), f_j)$;
else if $C(t) \in \mathcal{J}_3$ **then**
    if *k is even and $C(t)$ is not an unsafe configuration* **then**
        if *$\exists f \in F_y$ such that f is unsaturated* **then**
            Let f_j be the *target* fixed point;
            Execute $CircleFormation(C_3(t), f_j)$;
        else if *$\exists f \in F_y$ such that f is unsaturated* **then**
            Let $f_j \in C_1(t)$ and $f_b \in C_2(t)$ be the *target* fixed points;
            Execute $CircleFormation(C_1(t), f_j)$ and $CircleFormation(C_2(t), f_j)$;
        end
    else if *$C(t)$ is an unsafe configuration* **then**
        Execute $PivotSelection(C(t))$;
    end
end

1. $\boldsymbol{C(t) \in \mathcal{J}_1}$. Since F is asymmetric, the fixed points can be ordered. Let f be the topmost asymmetric fixed point. In case there are multiple such fixed points, select

the one that has the minimum rank. The direction from the y-axis towards f is considered to be the positive x-axis direction. This is a global agreement. Let $f_i \in C(t)$ be the *unsaturated* fixed point that has the maximum rank. The robots would select f_i as the *target* fixed point. The robots would execute $CircleFormation(C(t), f_i)$.

2. $\boldsymbol{C(t) \in \mathcal{J}_2}$. Let $f_a \in C_1(t)$ be the *unsaturated* fixed point that has the maximum rank. The robots would select f_i as the *target* fixed point in $C_1(t)$. Similarly, the robots would select a unique *target* fixed point (say f_b) in $C_2(t)$. The robots would execute $CircleFormation(C_1(t), f_a)$ and $CircleFormation(C_2(t), f_b)$.

3. $\boldsymbol{C(t) \in \mathcal{J}_3}$. In this case, $F_y \neq \emptyset$. The following cases are to be considered:

   (a) k is even and $C(t)$ is not an *unsafe* configuration. Consider the following cases:

      (i) $\exists f \in F_y$ such that f is *unsaturated*. Let $f_j \in F_y$ be the topmost *unsaturated* fixed point. f_j is selected as the *target* fixed point. They would execute $CircleFormation(C_3(t), f_j)$.

      (ii) $\forall f \in F_y$, f is *saturated*. Let $f_a \in C_1(t)$ be the *unsaturated* fixed point that has the maximum rank. f_a is selected as the *target* fixed point in $C_1(t)$. Since the fixed points in $C_1(t)$ are orderable, f_a is unique. Similarly, the robots would select a unique *target* fixed point (say f_b) in $C_2(t)$. The robots would execute $CircleFormation(C_1(t), f_a)$ and $CircleFormation(C_2(t), f_b)$.

   (b) $C(t)$ is an *unsafe* configuration. If none of the *pivot* positions have been occupied, then the robots would execute $PivotSelection(C(t))$. Next, consider the case when one of the *pivot* positions has been occupied. The direction from the y-axis towards the *pivot* robot is considered as the positive x-axis direction. This is a global agreement. Next, the algorithm proceeds similarly to case 1.

6.5 Correctness

The following points are shown to prove the correctness of $AlgorithmFatRobot$:

1. *Solvability:* At any arbitrary point of time $t > 0$, $C(t) \notin \mathcal{U}_4$.

2. *Progress:* The *uniform k-circle formation* is solved within finite time.

6.5.1 Solvability

Lemma 6.5.1. *If $C(0) \in \mathcal{J}_1 \cup \mathcal{J}_2$ and $C(0) \notin \mathcal{U}_4$, then at any arbitrary point of time $t \geq 0$ during an execution of $AlgorithmFatRobot$, $C(t) \notin \mathcal{U}_4$.*

Proof. The following cases are to be considered:

Case 1. $C(0) \in \mathcal{J}_1$. Since F is asymmetric in $C(0)$, it would always remain asymmetric. Thus, $C(t) \notin \mathcal{U}_4$.

Case 2. $C(0) \in \mathcal{J}_2$. $F_y = \emptyset$. Since $F_y \neq \emptyset$ for each configuration in $\mathcal{U}_4$, $C(t) \notin \mathcal{U}_4$.

Hence, if $C(0) \in \mathcal{J}_1 \cup \mathcal{J}_2$ and $C(0) \notin \mathcal{U}_4$, then at any arbitrary point of time $t \geq 0$ during an execution of $AlgorithmFatRobot$, $C(t) \notin \mathcal{U}_4$. □

Lemma 6.5.2. *If $C(0) \in \mathcal{J}_3$ and $C(0) \notin \mathcal{U}_4$, then at any arbitrary point of time $t > 0$ during an execution of $AlgorithmFatRobot$, $C(t) \notin \mathcal{U}_4$.*

Proof. If k is even and $C(t)$ is not an *unsafe* configuration, then $C(t) \notin \mathcal{U}_4$, $\forall t \geq 0$. Assume that $C(t)$ is an *unsafe* configuration. $PivotSelection(C(0))$ is executed. Let t_1 be the point of time at which the *pivot* robot (say R_j) is placed at one of the *pivot* positions (say ρ_1). The *pivot* robot R_j would remain static $\forall t \geq t_1$. All the robots can uniquely identify the *pivot* robot. First, all the fixed points belonging to the half-plane containing the *pivot* robot would be selected for circle formation. This is because an *unsaturated* fixed point that has the highest rank is selected as a *target* fixed point. This would ensure that R_j remain asymmetric about the y-axis. Therefore, $C(t) \notin \mathcal{U}_4$ for $t \geq t_1$. Next, we need to show that $C(t) \notin \mathcal{U}_4$ for $t \in [0, t_1]$. The following cases are to be considered:

Case 1. A robot moves horizontally one unit away from the y-axis. Let R_a be such a robot. There are two possible subcases:

Subcase 1. $R_a(0) \in R_y(0)$. First, consider that $R_a(0) = R_a(t)$. Since $R_a(t) \in R_y(t)$, $R_y(t) \neq \emptyset$. Therefore, $C(t) \notin \mathcal{U}_4$. Next, consider that $R_a(0) \neq R_a(t)$. Since R_a has moved to one of the half-planes, the configuration has become asymmetric about the y-axis. Therefore, $C(t) \notin \mathcal{U}_4$.

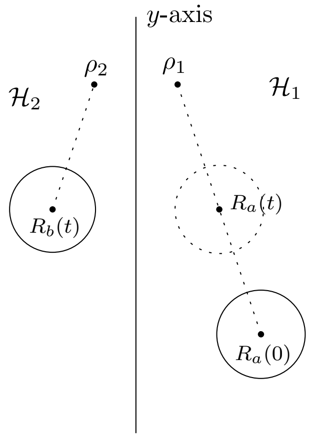


FIGURE 6.8: Since $d(R_b(0), \rho_2) < d(R_a(0), \rho_1)$, R_b would get selected for moving towards the *pivot* position ρ_2.

Subcase 2. $R_a(0) \notin R_y(0)$. It has been selected for horizontal movement to create space for some robot that lies on the y-axis. Thus, $R_y(t) \neq \emptyset$ during R_a's horizontal movement. Therefore, $C(t) \notin \mathcal{U}_4$.

Case 2. A robot moves towards one of the *pivot* positions. Without loss of generality, assume that $R_a \in \mathcal{H}_1$ has been selected for moving towards ρ_1. If possible, let $R_a(t)$ become symmetric with $R_b(t)$ (Figure 6.8). In the time interval $[0, t]$, R_a is the only robot that has been selected for movement. Thus, $R_b(0) = R_b(t)$. This contradicts $d(R_a(0), \rho_1) = \min_{\rho_i \in \{\rho_1,\, \rho_2\},\, R_j \in R(t)} d(R_j(0), \rho_i)$. Therefore, $C(t) \notin \mathcal{U}_4$.

Hence, if $C(0) \in \mathcal{J}_3$ and $C(0) \notin \mathcal{U}_4$, then at any arbitrary point of time $t > 0$ during an execution of *AlgorithmFatRobot*, $C(t) \notin \mathcal{U}_4$. □

6.5.2 Progress

During an execution of *AlgorithmFatRobot*, a robot will move by the following procedures: (i) *PivotSelection*, (ii) *DownwardMovement*, (iii) *CircleFormation*.

6.5.2.1 Progress during *DownwardMovement*

Progress of first kind: For some $R_j \in C(t)$, consider an execution of procedure $DownwardMovement(R_j)$. Define $k_1(t) = d(R_j(t), p_j(t))$ and $k_2(t) = d(R_j(t), q_j(t))$. In case $|M_j(t)| > 0$, let $R_a \in M_j(t)$ be a robot that has the minimum rank. Define $d_1(t) = d(R_a(t), p_a(t))$. If $|M_j(t)| = 0$, then assume that $d_1(t) = 0$. Similarly, if $|N_j(t)| > 0$ then assume that $R_b \in N_j(t)$ be a robot that lies at the farthest distance from $R_j(t)$. Define $d_2(t) = d(R_b(t), s_4)$. If $|N_j(t)| = 0$, then assume that $d_2(t) = 0$. Define $Z_7(t) = (k_1(t), |M_j(t)|, d_1(t))$ and $Z_8(t) = (k_2(t), |N_j(t)|, d_2(t))$. In the time interval t to t', $Z_i(t') < Z_i(t)$ where $i \in \{7, 8\}$ if $Z_i(t')$ is lexicographically smaller than $Z_i(t)$. During an execution of *DownwardMovement*, the configuration is said to have *progress* of first kind in the time interval t to t' if either $Z_7(t') < Z_7(t)$ or $Z_8(t') < Z_8(t)$.

Lemma 6.5.3. *During the execution of $DownwardMovement(R_j)$ for some $R_j \in C(t)$, let $t' > t$ be the point of time at which each robot has completed at least one LCM cycle. Progress of first kind is ensured in the time interval t to t'.*

Proof. The following cases are to be considered:

Case 1. R_j can move vertically one unit downwards. Since it would move by at least δ amount, $k_1(t') + \delta \leq k_1(t)$. Thus, $Z_7(t') < Z_7(t)$.

Case 2. R_j can not be moved vertically one unit downwards and $M_j(t)$ does not contain the *pivot* robot. Let $R_a \in M_j(t)$ be a robot that has the minimum rank. It would start moving towards $p_j(t)$. If $R_j(t') = p_j(t)$, then $|M_j(t')| = |M_j(t)| - 1$. Otherwise, since it would move by at least δ amount, $d_1(t') + \delta \leq d_1(t)$. Thus, $Z_7(t') < Z_7(t)$.

Case 3. R_j can not be moved vertically one unit downwards and $M_j(t)$ contains the *pivot* robot. There are two possible subcases:

Subcase 1. $N_j(t) = \emptyset$. Since it would move by at least δ amount, $k_2(t') + \delta \leq k_2(t)$. Thus, $Z_8(t') < Z_8(t)$.

Subcase 2. $N_j(t) \neq \emptyset$. Let $R_b \in N_j(t)$ be a robot that lies at the farthest distance from R_j. It would start moving towards $q_b(t)$. If $R_b(t') = q_b(t)$, then $|N_j(t')| = |N_j(t)| - 1$. Otherwise, since it would move by at least δ amount, $d_2(t') + \delta \leq d_2(t)$. Thus, $Z_8(t') < Z_8(t)$.

Hence, during the execution of $DownwardMovement(R_j)$ for some $R_j \in C(t)$ *progress* of first kind is ensured in the time interval t to t'. □

6.5.2.2 Progress during $PivotSelection$

Lemma 6.5.4. *Let $C(t) \in \mathcal{J}_3$ with odd values of k. The pivot robot would be placed within finite time by the execution of $PivotSelection(C(t))$.*

Proof. Let $t' > t$ be an arbitrary point of time at which each robot has completed at least one LCM cycle during an execution of $PivotSelection$. The following cases are to be considered:

Case 1. R_j moves towards the *pivot* position ρ_1. Since R_j is guaranteed to move by at least δ amount towards ρ_1, $d(R_j(t'), \rho_1) + \delta \leq d(R_j(t), \rho_1)$. As $d(R_j(t), \rho_1)$ is finite, R_j would eventually reach the *pivot* position.

Case 2. $R_j \in R_y(t)$ moves horizontally half unit away from the y-axis. Such a movement is performed when a unique robot can not be selected for moving towards one of the *pivot* positions. Since R_j would move by at least δ amount, the configuration is guaranteed to become asymmteric at t'. Next, a unique robot can be selected for moving towards one of the *pivot* positions.

Case 3. R_j executes $DownwardMovement(R_j)$. Such a movement is performed when $\nexists R_a \in R_y(t)$ that can be moved horizontally half unit away from the y-axis. From Lemma 6.5.3, it follows that *progress* of first kind is ensured. Thus, within finite time $M_j(t)$ would become empty. Next, $R_a \in R_y(t)$ can be moved horizontally half unit away from the y-axis.

Hence, by the execution of $PivotSelection(C(t))$, the *pivot* robot would be placed within finite time. □

6.5.2.3 Progress during $CircleFormation$

Progress of second kind: Suppose R_j has been selected for movement towards a vertex $\beta_k \in \mathcal{P}_i$ during an execution $CircleFormation(C(t), f_i)$. For the configurations without any global x-axis agreement, there might be two such moving robots. In that case, both the robots would move towards different vertices of $\mathcal{P}_i$. First, consider the case when there is only one such robot. Recall that $D_i(t) = k - |C(f_i, \rho) \cap R(t)|$ and $n_k(t)$ denotes the number of *unsaturated* fixed points. Let

$$E_j(t) = \begin{cases} d(R_j(t), \beta_k) & \mathscr{R}(R_j(t)\beta_k) \text{ is empty} \\ d(R_j(t), a_j) & \mathscr{R}(R_j(t)\beta_k) \text{ is non-empty} \end{cases}$$

Let $V_j(t) = (n(t), n_i(t), E_j(t))$. Next, consider the case when there are two such moving robots. Let R_a be the other robot that starts moving towards a vertex $\beta_b \in \mathcal{P}_i$. Similarly, define $E_a(t)$ and $V_a(t) = (n_k(t), D_i(t), E_a(t))$. In the time interval t to t', $V_i(t') < V_i(t)$, where $i \in \{j, a\}$ if $V_i(t')$ is lexicographically smaller than $V_i(t)$. During an execution of $AlgorithmFatRobot$, the configuration is said to have *progress* of second kind in the time interval t to t', if either $V_j(t') < V_j(t)$ or $V_a(t') < V_a(t)$.

Lemma 6.5.5. *Let $C(t)$ be a given configuration. During the execution of the procedure $CircleFormation$, let $t' > t$ be an arbitrary point of time at which all the robots have completed at least one LCM cycle. Either progress of first kind or progress of second kind is ensured in the time interval between t and t'.*

Proof. Let f_i be the *target* fixed point. First, consider the case when the robots have a global x-axis agreement or $f_i \notin F_y$. Consider the following cases:

Case 1. $|B_i(t)| > 1$. Let $R_j \in B_i(t)$ be the robot that has the highest rank. The procedure $DownwardMovement(R_j)$ would be executed. Lemma 6.5.3 ensures *progress* of first kind. Since $|B_i(t)| \leq n$, $|B_i(t)| = 1$ would be satisfied within finite time. If $\exists R_a \in R(t)$ such that $R_a \in M_j(t) \cap A_i(t)$ or $R_a \in N_j(t) \cap A_i(t)$, then $|B_i(t)|$ would increase by the execution of $DownwardMovement$. Since $|B_i(t)| \leq n$, Lemma 6.5.3 ensures that $|B_i(t)| = 1$ would be satisfied within finite time.

Case 2. $|B_i(t)| = 1$. Let $R_j \in B_i(t)$. Suppose β_a be the empty vertex of $\mathcal{P}_i$ that has the highest rank. There are two possible subcases:

Subcase 1. $\mathscr{R}(R_j(t)\beta_a)$ is empty. R_j would start moving towards β_a along $\overline{R_j(t)\beta_a}$. If $\beta_a = R_j(t')$, then $D_j(t') = D_j(t) - 1$. The configuration would satisfy $|B_i(t)| = 0$ and

$C(f_i, \rho)$ is *perfect*. Else, either $R_j(t') \in C(f_i, \rho)$ or $R_j(t') \in B_i(t)$. If $R_j(t') \in C(f_i, \rho)$, then $C(t')$ would satisfy $|B_i(t)| = 0$ and $C(f_i, \rho)$ is *imperfect*. Otherwise, $C(t')$ would still satisfy $|B_i(t)| = 1$. However, R_j has moved by at least δ amount, $d(R_j(t'), \beta_a) + \delta \leq d(R_j(t), \beta_a)$ is satisfied. Thus, *progress* of second kind is ensured.

Subcase 2. $\mathscr{R}(R_j(t)\beta_a)$ is not empty. Procedure $DownwardMovement(R_j)$ is executed. Lemma 6.5.3 ensures *progress* of first kind. If $M_j(t) \neq \emptyset$, then $|B_i(t)| > 1$ would be satisfied. In case $M_j(t) = \emptyset$, then either $\mathscr{R}(R_j(t)\beta_a)$ would become empty or, $|B_i(t)| = 0$ would be satisfied.

Case 3. $|B_i(t)| = 0$ and $C(f_i, \rho)$ is *imperfect*. Suppose β_a be the empty vertex of $\mathcal{P}_i$ that has the highest rank. Let R_j be the robot that starts moving towards β_a. Since δ is the minimum distance traveled by a robot, $d(R_j(t'), \beta_a) + \delta \leq d(R_j(t), \beta_a)$. Thus, *progress* of second kind is ensured.

Case 4. $|B_i(t)| = 0$ and $C(f_i, \rho)$ is *perfect*. Suppose β_a be the empty vertex of $\mathcal{P}_i$ that has the highest rank. Let $R_j \in R(t) \setminus A_i(t)$ be the robot that lies at the closest distance from β_a. If there is a tie, the robot that has the maximum rank is selected. If $\mathscr{R}(R_j(t)\beta_a)$ is non-empty, then R_j would start moving towards a_j. Else, it would start moving towards β_a. In both the cases, *progress* of second kind is ensured.

Next, consider the case when the robots do not have any global x-axis agreement. In such a case, there may be two moving robots in the plane. Such robots would be delimited by the y-axis. Additionally, their destinations would also lie in their respective half-planes. From the above cases (Case 1, 2, 3 and 4), it follows that either *progress* of first kind or *progress* of second kind is ensured for both the moving robots. Hence, during an execution of $CircleFormation$ either *progress* of first kind or *progress* of second kind is ensured in the time interval $[t, t']$. □

Lemma 6.5.6. *Let $C(0)$ be a given initial configuration. During the execution of algorithm $AlgorithmFatRobot$ collision-free movement is ensured by the robots.*

Proof. During the execution of the $AlgorithmFatRobot$, the robots would move sequentially. A robot would start moving towards its destination point only if a free path exists. Thus, during its movement, it would avoid any collisions with other robots. In case the robots do not have any global x-axis agreement, there may be two moving robots in the plane. However, they would lie in different half-planes delimited by the y-axis, avoiding any possible collisions. Therefore, during the execution of the $AlgorithmFatRobot$, collision-free movement is ensured. □

Theorem 6.5.7. *If $C(0) \notin \mathcal{U}_4$, then the uniform k-circle formation problem is deterministically solvable by the execution of $AlgorithmFatRobot$.*

Proof. During an execution of $AlgorithmFatRobot$, Lemma 6.5.1 and Lemma 6.5.2 ensure that $C(t) \notin \mathcal{U}_4$ at any arbitrary point of time $t > 0$. Collision-free movement is guaranteed by Lemma 6.5.6. If $C(0) \in \mathcal{J}_3$ and $C(0)$ is an *unsafe* configuration, then Lemma 6.5.4 ensures that the *pivot* robot would be placed within finite time by the execution of $PivotSelection$. Lemma 6.5.5 ensures that within finite time the configuration $C(t)$ for some $t \geq 0$ would satisfy the condition $|B_i(t)| = 1$. If the robots do not have a global x-axis agreement, then Lemma 6.5.5 ensures that within finite time the configuration $C(t)$ for some $t \geq 0$ would satisfy the condition $0 < |B_i(t)| \leq 2$. Since $|F|$ is finite, Lemma 6.5.5 guarantees that the robots would accomplish the formation of circles by the procedure $CircleFormation$. Hence, the robots would deterministically solve the *uniform k-circle formation* problem by the execution of $AlgorithmFatRobot$. □

6.6 Conclusion

In this chapter, the *uniform k-circle formation* problem is studied for ASYNC fat robots. The robots have an agreement on the direction and orientation of the y-axis. The following results have been proved:

Result 1: If $C(0) \in \mathcal{U}_4$, then the *uniform k-circle formation* problem is deterministically *unsolvable*.

Result 2: If $C(0) \notin \mathcal{U}_4$, then the *uniform k-circle formation* problem is deterministically *solvable*.

Chapter 7

Conclusions

7.1 Contributions of the Thesis

This thesis is primarily focused on the theoretical aspects of solving the *k-circle formation* problem by a swarm of mobile robots. The *k-circle formation* problem is a hybrid problem that connects the well studied problems: the *partitioning* problem, the *circle formation* problem and *embedded pattern formation* problem. Our aim is to identify different sets of computational assumptions under which the *k-circle formation* problem is solvable. All the studied problems have been considered under an ASYNC scheduler with non-rigid motion.

In Chapter 3, the *k-circle formation* problem has been investigated under one axis agreement. First, we have assumed that $n = km$. All the *initial* configurations and values of k for which the *k-circle formation* problem is deterministically *unsolvable* have been characterized. A deterministic distributed algorithm is proposed that solves the *k-circle formation* problem within finite time. Next, the solvability of the problem is discussed for the cases when $n \neq km$. Finally, it has been shown that if the *k-circle formation* problem is deterministically *solvable* then the *k-EPF* problem is also deterministically *solvable*.

Chapter 4 addresses the relaxation of the assumption of one axis agreement among the robots. In this chapter, the *k-circle formation* problem is considered for completely *disoriented* robots. When the robots have one axis agreement, all the robots and fixed points can be ordered with respect to the axis of agreement. Thus, the presence of rotational symmetries can be handled successfully. In this current setting, rotational symmetries must be considered in addition to reflectional symmetries. All the *initial* configurations and values of k for which the k-*circle formation* problem is deterministically *unsolvable* have been characterized. As a consequence, the set of unsolvable cases is larger compared to the set of unsolvable cases under one axis agreement. A deterministic distributed algorithm is proposed that solves the k-*circle formation* problem within finite time for *disoriented* robots.

The assumption of *unlimited* visibility for the robots has a significant influence on the results presented in Chapter 3 and Chapter 4. Chapter 5 investigates the k-*circle formation* problem under *obstructed* visibility model. Based upon the visibility of fixed points, the k-*circle formation* problem under *obstructed* visibility is studied for two different settings, namely (a) complete knowledge of the fixed points and (b) zero knowledge of the fixed points. In case where the robots have complete knowledge of the fixed points, a deterministic distributed algorithm is proposed that solves the k-*circle formation* problem for *oblivious* and *silent* robots. In the setting where the robots do not have any knowledge of the fixed points, a deterministic distributed algorithm is proposed that solves the k-*circle formation* problem for robots equipped with lights.

While point robots are easy to handle, in a more realistic model, the robots would have dimensions. In Chapter 6, the *uniform* k-*circle formation* is investigated for robots with dimensional extent under one axis agreement. The robots have *unlimited* visibility. All the *initial* configurations and values of k for which the *uniform* k-*circle formation* problem is deterministically *unsolvable* have been characterized. A deterministic distributed algorithm is proposed that solves the *uniform* k-*circle formation* problem within finite time.

A summary of the contributions of this thesis is presented in Table 7.1.

Agreement	**Visibility**	**Knowledge of Fixed points**	**Dimension**	**Light Color**	**Status**
One-Axis	Unlimited	Complete	Point	1	Solved-Chapter 3
No-Axis	Unlimited	Complete	Point	1	Solved-Chapter 4
No-Axis	Obstructed	Complete	Point	1	Solved-Chapter 5
No-Axis	Obstructed	Zero	Point	1	Solved-Chapter 5
One-Axis	Unlimited	Complete	Fat	1	Solved-Chapter 6
No-Axis	Unlimited	Complete	Fat	1	Unsolved
No-Axis	Obstructed	Complete or Zero	Fat	1	Unsolved

TABLE 7.1: Results related to the k-Circle Formation

7.2 Future Directions

The *k-circle formation* problem has a wide range of potential extensions for future research. For example, a solution for the *k-circle formation* problem where the circles may have different radii can be investigated. The following are some of the potential future research directions:

1. **Partial Knowledge of Fixed Points under Obstructed Visibility:** As a future direction the problem can be considered in a setting where the robots have the *partial* knowledge of the fixed points. For example, one may consider the case where the robots have the knowledge of the total number of fixed points but have no knowledge of the positions of them.

2. **Limited Visibility:** The *k-circle formation* problem can be considered under *limited* visibility. Depending on the visibility of the fixed points different settings can be considered, namely (a) the fixed points are only visible when they lie within the visibility range of a robot, (b) the robots have the knowledge of the positions of all the fixed points.

3. **Fat Robots:** The *k-circle formation* problem for fat robots has been investigated under the one axis agreement. However, the necessity of one axis agreement has not been discussed. The problem can be considered in the future for completely *disoriented* fat robots. Also, it has been assumed that the robots have *unlimited*

visibility. Another direction of future work would be to consider the *k-circle formation* problem for fat robots under restricted visibility models, namely *obstructed* visibility and *limited* visibility.

4. **Objective Functions:** The problem can also be considered with different objective functions, namely, minimizing the total distance traveled by all robots or the maximum distance traveled by an individual robot.

5. **Randomization:** Some of the symmetric configurations have remain deterministically *unsolvable*. In future, a randomized solution for the *k-circle formation* problem can be investigated.

6. **Fault-tolerant algorithms:** Since the robots may become faulty, one of the future directions would be to consider fault-tolerant algorithms, namely crash-fault-tolerant and byzantine-fault-tolerant.

Bibliography

[1] Edison Pignaton de Freitas, Maik Basso, Antonio Arlis Santos da Silva, Marcos Rodrigues Vizzotto, and Mateus Schein Cavalheiro Corrêa. A distributed task allocation protocol for cooperative multi-uav search and rescue systems. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 909–917. IEEE, 2021.

[2] Truong Duy Dinh, Rustam Pirmagomedov, Van Dai Pham, Aram A Ahmed, Ruslan Kirichek, Ruslan Glushakov, and Andrei Vladyko. Unmanned aerial system–assisted wilderness search and rescue mission. *International Journal of Distributed Sensor Networks*, 15(6):1550147719850719, 2019.

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

[4] Marc Steinberg. Intelligent autonomy for unmanned naval systems. In *Unmanned Systems Technology VIII*, volume 6230, pages 359–370. SPIE, 2006.

[5] Moulay A Akhloufi, Nicolás A Castro, and Andy Couturier. Uavs for wildland fires. In *Autonomous systems: Sensors, vehicles, security, and the Internet of Everything*, volume 10643, pages 134–147. SPIE, 2018.

[6] Khaled A Ghamry, Mohamed A Kamel, and Youmin Zhang. Multiple uavs in forest fire fighting mission using particle swarm optimization. In *2017 International conference on unmanned aircraft systems (ICUAS)*, pages 1404–1409. IEEE, 2017.

[7] Ayan Dutta, Swapnoneel Roy, O Patrick Kreidl, and Ladislau Bölöni. Multi-robot information gathering for precision agriculture: Current state, scope, and challenges. *IEEE Access*, 9:161416–161430, 2021.

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

[9] 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.

[10] Subhash Bhagat and Krishnendu Mukhopadyaya. Fault-tolerant gathering of semi-synchronous robots. In *Proceedings of the 18th International Conference on Distributed Computing and Networking*, pages 1–10, 2017.

[11] Sruti Gan Chaudhuri and Krishnendu Mukhopadhyaya. Leader election and gathering for asynchronous fat robots without common chirality. *Journal of Discrete Algorithms*, 33:171–192, 2015.

[12] Serafino Cicerone, Gabriele Di Stefano, and Alfredo Navarra. Gathering of robots on meeting-points: feasibility and optimal resolution algorithms. *Distributed Computing*, 31(1):1–50, Feb 2018.

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

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

[15] Jurek Czyzowicz, Leszek Gasieniec, and Andrzej Pelc. Gathering few fat mobile robots in the plane. *Theoretical Computer Science*, 410(6-7):481–499, 2009.

[16] Gianlorenzo d'Angelo, Gabriele Di Stefano, Ralf Klasing, and Alfredo Navarra. Gathering of robots on anonymous grids and trees without multiplicity detection. *Theoretical Computer Science*, 610:158–168, 2016.

[17] Xavier Défago, Maria Gradinariu, Stéphane Messika, and Philippe Raipin-Parvédy. Fault-tolerant and self-stabilizing mobile robots gathering. In *International Symposium on Distributed Computing*, pages 46–60. Springer, 2006.

[18] Anthony Honorat, Maria Potop-Butucaru, and Sébastien Tixeuil. Gathering fat mobile robots with slim omnidirectional cameras. *Theoretical Computer Science*, 557:1–27, 2014.

[19] Taisuke Izumi, Samia Souissi, Yoshiaki Katayama, Nobuhiro Inuzuka, Xavier Défago, Koichi Wada, and Masafumi Yamashita. The gathering problem for two oblivious robots with unreliable compasses. *SIAM Journal on Computing*, 41(1):26–46, 2012.

[20] Ralf Klasing, Euripides Markou, and Andrzej Pelc. Gathering asynchronous oblivious mobile robots in a ring. *Theoretical Computer Science*, 390(1):27–39, 2008.

[21] Giuseppe Prencipe. Impossibility of gathering by a set of autonomous mobile robots. *Theoretical Computer Science*, 384(2-3):222–231, 2007.

[22] Giovanni Viglietta. Rendezvous of two robots with visible bits. In *International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics*, pages 291–306. Springer, 2014.

[23] Davide Canepa, Xavier Defago, Taisuke Izumi, and Maria Potop-Butucaru. Flocking with oblivious robots. In *International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pages 94–108. Springer, 2016.

[24] Davide Canepa and Maria Gradinariu Potop-Butucaru. Stabilizing flocking via leader election in robot networks. In *Symposium on Self-Stabilizing Systems*, pages 52–66. Springer, 2007.

[25] Vincenzo Gervasi and Giuseppe Prencipe. Coordination without communication: The case of the flocking problem. *Discrete Applied Mathematics*, 144(3):324–344, 2004.

[26] Samia Souissi, Yan Yang, and Xavier Défago. Fault-tolerant flocking in a k-bounded asynchronous system. In *International Conference On Principles Of Distributed Systems*, pages 145–163. Springer, 2008.

[27] Yan Yang, Samia Souissi, Xavier Défago, and Makoto Takizawa. Fault-tolerant flocking for a group of autonomous mobile robots. *Journal of systems and Software*, 84(1):29–36, 2011.

[28] Subhash Bhagat and Krishnendu Mukhopadhyaya. Optimum circle formation by autonomous robots. In *Advanced Computing and Systems for Security*, pages 153–165. Springer, 2018.

[29] Quentin Bramas and Sébastien Tixeuil. Brief announcement: Probabilistic asynchronous arbitrary pattern formation. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, pages 443–445, 2016.

[30] Serafino Cicerone, Gabriele Di Stefano, and Alfredo Navarra. Asynchronous arbitrary pattern formation: the effects of a rigorous approach. *Distributed Computing*, 32(2):91–132, 2019.

[31] Serafino Cicerone, Gabriele Di Stefano, and Alfredo Navarra. Embedded pattern formation by asynchronous robots without chirality. *Distributed Computing*, 32(4):291–315, 2019.

[32] Yoann Dieudonné, Ouiddad Labbani-Igbida, and Franck Petit. Circle formation of weak mobile robots. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 3(4):1–20, 2008.

[33] Caterina Feletti, Carlo Mereghetti, and Beatrice Palano. Uniform circle formation for swarms of opaque robots with lights. In *Stabilization, Safety, and Security of Distributed Systems*, pages 317–332. Springer International Publishing, 2018.

[34] Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Giovanni Viglietta. Distributed computing by mobile robots: uniform circle formation. *Distributed Computing*, 30(6):413–457, 2017.

[35] Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Peter Widmayer. Arbitrary pattern formation by asynchronous, anonymous, oblivious robots. *Theoretical Computer Science*, 407(1):412 – 447, 2008.

[36] Nao Fujinaga, Yukiko Yamauchi, Shuji Kijima, and Masafumi Yamashita. Asynchronous pattern formation by anonymous oblivious mobile robots. In *International Symposium on Distributed Computing*, pages 312–325. Springer, 2012.

[37] Masafumi Yamashita and Ichiro Suzuki. Characterizing geometric patterns formable by oblivious anonymous mobile robots. *Theoretical Computer Science*, 411(26):2433 – 2453, 2010.

[38] John Augustine and William K Moses Jr. Dispersion of mobile robots: A study of memory-time trade-offs. In *Proceedings of the 19th International Conference on Distributed Computing and Networking*, pages 1–10, 2018.

[39] Lali Barriere, Paola Flocchini, Eduardo Mesa-Barrameda, and Nicola Santoro. Uniform scattering of autonomous mobile robots in a grid. *International Journal of Foundations of Computer Science*, 22(03):679–697, 2011.

[40] Reuven Cohen and David Peleg. Local spreading algorithms for autonomous robot systems. *Theoretical Computer Science*, 399(1-2):71–82, 2008.

[41] Ajay D Kshemkalyani, Anisur Rahaman Molla, and Gokarna Sharma. Fast dispersion of mobile robots on arbitrary graphs. In *International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics*, pages 23–40. Springer, 2019.

[42] Pavan Poudel and Gokarna Sharma. Time-optimal uniform scattering in a grid. In *Proceedings of the 20th International Conference on Distributed Computing and Networking*, pages 228–237, 2019.

[43] Subhash Bhagat, Paola Flocchini, Krishnendu Mukhopadyaya, and Nicola Santoro. Weak robots performing conflicting tasks without knowing who is in their team. In *Proceedings of the 21st International Conference on Distributed Computing and Networking*, pages 1–6, 2020.

[44] Zhiqiang Liu, Yukiko Yamauchi, Shuji Kijima, and Masafumi Yamashita. Team assembling problem for asynchronous heterogeneous mobile robots. *Theoretical Computer Science*, 721:27–41, 2018.

[45] Kaustav Bose, Manash Kumar Kundu, Ranendu Adhikary, and Buddhadeb Sau. Arbitrary pattern formation by asynchronous opaque robots with lights. *Theoretical Computer Science*, 849:138–158, 2021.

[46] Giuseppe Antonio Di Luna, Paola Flocchini, S Gan Chaudhuri, Federico Poloni, Nicola Santoro, and Giovanni Viglietta. Mutual visibility by luminous robots without collisions. *Information and Computation*, 254:392–418, 2017.

[47] David Peleg. Distributed coordination algorithms for mobile robot swarms: New directions and challenges. In *International Workshop on Distributed Computing*, pages 1–12. Springer, 2005.

[48] Quentin Bramas, Anissa Lamani, and Sébastien Tixeuil. The agreement power of disagreement. In *International Symposium on Stabilizing, Safety, and Security of Distributed Systems*, pages 273–288. Springer, 2021.

[49] Suparno Datta, Ayan Dutta, Sruti Gan Chaudhuri, and Krishnendu Mukhopadhyaya. Circle formation by asynchronous transparent fat robots. In *Distributed Computing and Internet Technology, 9th International Conference, ICDCIT 2013, Bhubaneswar, India, February 5-8, 2013. Proceedings*, pages 195–207, 2013.

[50] Xavier Défago and Samia Souissi. Non-uniform circle formation algorithm for oblivious mobile robots with convergence toward uniformity. *Theoretical Computer Science*, 396(1-3):97–112, 2008.

[51] Ayan Dutta, Sruti Gan Chaudhuri, Suparno Datta, and Krishnendu Mukhopadhyaya. Circle formation by asynchronous fat robots with limited visibility. In *In proc. 8th International Conference on Distributed Computing and Internet Technology (ICDCIT-2012)*, pages 83–93, 2012.

[52] Moumita Mondal and Sruti Gan Chaudhuri. Uniform circle formation by swarm robots under limited visibility. In *International Conference on Distributed Computing and Internet Technology*, pages 420–428. Springer, 2020.

[53] Ranendu Adhikary, Manash Kumar Kundu, and Buddhadeb Sau. Circle formation by asynchronous opaque robots on infinite grid. *Computer Science*, 22(1), 2021.

[54] Arijit Sil and Sruti Gan Chaudhuri. Formation of straight line by swarm robots. In *Computational Intelligence and Machine Learning: Proceedings of the 7th International Conference on Advanced Computing, Networking, and Informatics (ICACNI 2019)*, pages 99–111. Springer, 2021.

[55] Yusaku Tomita, Yukiko Yamauchi, Shuji Kijima, and Masafumi Yamashita. Plane formation by synchronous mobile robots without chirality. In *21st International Conference on Principles of Distributed Systems*, 2018.

[56] Taichi Uehara, Yukiko Yamauchi, Shuji Kijima, and Masafumi Yamashita. Plane formation by semi-synchronous robots in the three dimensional euclidean space. In *International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pages 383–398. Springer, 2016.

[57] Yukiko Yamauchi, Taichi Uehara, Shuji Kijima, and Masafumi Yamashita. Plane formation by synchronous mobile robots in the three-dimensional euclidean space. *Journal of the ACM (JACM)*, 64(3):1–43, 2017.

[58] Ranendu Adhikary, Kaustav Bose, Manash Kumar Kundu, and Buddhadeb Sau. Mutual visibility on grid by asynchronous luminous robots. *Theoretical Computer Science*, 2022.

[59] Aisha Aljohani and Gokarna Sharma. Complete visibility for mobile robots with lights tolerating faults. *International Journal of Networking and Computing*, 8(1):32–52, 2018.

[60] Subhash Bhagat and Krishnendu Mukhopadhyaya. Optimum algorithm for mutual visibility among asynchronous robots with lights. In *International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pages 341–355. Springer, 2017.

[61] Giuseppe Antonio Di Luna, Paola Flocchini, Federico Poloni, Nicola Santoro, and Giovanni Viglietta. The mutual visibility problem for oblivious robots. In *CCCG*, 2014.

[62] Giuseppe Antonio Di Luna, Paola Flocchini, Sruti Gan Chaudhuri, Nicola Santoro, and Giovanni Viglietta. Robots with lights: Overcoming obstructed visibility without colliding. In *Symposium on Self-Stabilizing Systems*, pages 150–164. Springer, 2014.

[63] Pavan Poudel, Aisha Aljohani, and Gokarna Sharma. Fault-tolerant complete visibility for asynchronous robots with lights under one-axis agreement. *Theoretical Computer Science*, 850:116–134, 2021.

[64] Gokarna Sharma, Costas Busch, and Supratik Mukhopadhyay. Bounds on mutual visibility algorithms. In *CCCG*, pages 268–274, 2015.

[65] Gokarna Sharma, Costas Busch, and Supratik Mukhopadhyay. Mutual visibility with an optimal number of colors. In *International Symposium on Algorithms and Experiments for Wireless Sensor Networks*, pages 196–210. Springer, 2015.

[66] Gokarna Sharma, Ramachandran Vaidyanathan, and Jerry L Trahan. Constant-time complete visibility for asynchronous robots with lights. In *International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pages 265–281. Springer, 2017.

[67] Gokarna Sharma, Ramachandran Vaidyanathan, Jerry L Trahan, Costas Busch, and Suresh Rai. O (log n)-time complete visibility for asynchronous robots with lights. In *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 513–522. IEEE, 2017.

[68] Ramachandran Vaidyanathan, Costas Busch, Jerry L Trahan, Gokarna Sharma, and Suresh Rai. Logarithmic-time complete visibility for robots with lights. In *2015 IEEE International Parallel and Distributed Processing Symposium*, pages 375–384. IEEE, 2015.

[69] Asaf Efrima and David Peleg. Distributed algorithms for partitioning a swarm of autonomous mobile robots. *Theoretical Computer Science*, 410(14):1355 – 1368, 2009.

[70] Chrysovalandis Agathangelou, Chryssis Georgiou, and Marios Mavronicolas. A distributed algorithm for gathering many fat mobile robots in the plane. In *Proceedings*

of the 2013 ACM symposium on Principles of distributed computing, pages 250–259, 2013.

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

[72] Subhash Bhagat and Anisur Rahaman Molla. Min-max gathering of oblivious robots. In *Proceedings of the 33rd ACM Symposium on Parallelism in Algorithms and Architectures*, pages 420–422, 2021.

[73] Subhash Bhagat and Krishnendu Mukhopadhyaya. Gathering asynchronous robots in the presence of obstacles. In *International Workshop on Algorithms and Computation*, pages 279–291. Springer, 2017.

[74] Subhash Bhagat and Krishnendu Mukhopadhyaya. Optimum gathering of asynchronous robots. In *Conference on Algorithms and Discrete Applied Mathematics*, pages 37–49. Springer, 2017.

[75] Zohir Bouzid, Shantanu Das, and Sébastien Tixeuil. Gathering of mobile robots tolerating multiple crash faults. In *2013 IEEE 33rd International Conference on Distributed Computing Systems*, pages 337–346. IEEE, 2013.

[76] Quentin Bramas and Sébastien Tixeuil. Wait-free gathering without chirality. In *International Colloquium on Structural Information and Communication Complexity*, pages 313–327. Springer, 2015.

[77] Reuven Cohen and David Peleg. Convergence of autonomous mobile robots with inaccurate sensors and movements. *SIAM Journal on Computing*, 38(1):276–302, 2008.

[78] Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Peter Widmayer. Gathering of asynchronous robots with limited visibility. *Theoretical Computer Science*, 337(1-3):147–168, 2005.

[79] Paola Flocchini, Nicola Santoro, Giovanni Viglietta, and Masafumi Yamashita. Rendezvous with constant memory. *Theoretical Computer Science*, 621:57–72, 2016.

[80] Takashi Okumura, Koichi Wada, and Yoshiaki Katayama. Brief announcement: Optimal asynchronous rendezvous for mobile robots with lights. In *International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pages 484–488. Springer, 2017.

[81] Pavan Poudel and Gokarna Sharma. Universally optimal gathering under limited visibility. In *International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pages 323–340. Springer, 2017.

[82] Subhash Bhagat, Abhinav Chakraborty, Bibhuti Das, and Krishnendu Mukhopadhyaya. Gathering over meeting nodes in infinite grid. *Fundamenta Informaticae*, 187(1):1–30, 2022.

[83] Subhash Bhagat, Abhinav Chakraborty, Bibhuti Das, and Krishnendu Mukhopadhyaya. Optimal gathering over weber meeting nodes in infinite grid. *International Journal of Foundations of Computer Science*, pages 1–25, 2022.

[84] François Bonnet, Maria Potop-Butucaru, and Sébastien Tixeuil. Asynchronous gathering in rings with 4 robots. In *International Conference on Ad-Hoc Networks and Wireless*, pages 311–324. Springer, 2016.

[85] Marjorie Bournat, Swan Dubois, and Franck Petit. Gracefully degrading gathering in dynamic rings. In *International Symposium on Stabilizing, Safety, and Security of Distributed Systems*, pages 349–364. Springer, 2018.

[86] Gianlorenzo D'Angelo, Gabriele Di Stefano, and Alfredo Navarra. Gathering six oblivious robots on anonymous symmetric rings. *Journal of Discrete Algorithms*, 26:16–27, 2014.

[87] Shantanu Das, Riccardo Focardi, Flaminia L Luccio, Euripides Markou, and Marco Squarcina. Gathering of robots in a ring with mobile faults. *Theoretical Computer Science*, 764:42–60, 2019.

[88] Giuseppe Antonio Di Luna, Paola Flocchini, Linda Pagli, Giuseppe Prencipe, Nicola Santoro, and Giovanni Viglietta. Gathering in dynamic rings. *theoretical computer science*, 811:79–98, 2020.

[89] Tomoko Izumi, Taisuke Izumi, Sayaka Kamei, and Fukuhito Ooshita. Mobile robots gathering algorithm with local weak multiplicity in rings. In *International Colloquium on Structural Information and Communication Complexity*, pages 101–113. Springer, 2010.

[90] Sayaka Kamei, Anissa Lamani, Fukuhito Ooshita, and Sébastien Tixeuil. Gathering an even number of robots in an odd ring without global multiplicity detection. In *International Symposium on Mathematical Foundations of Computer Science*, pages 542–553. Springer, 2012.

[91] Ralf Klasing, Adrian Kosowski, and Alfredo Navarra. Taking advantage of symmetries: Gathering of asynchronous oblivious robots on a ring. In *International Conference On Principles Of Distributed Systems*, pages 446–462. Springer, 2008.

[92] Giuseppe A Di Luna, Ryuhei Uehara, Giovanni Viglietta, and Yukiko Yamauchi. Gathering on a circle with limited visibility by anonymous oblivious robots. In *34th International Symposium on Distributed Computing*, page 1, 2020.

[93] Subhash Bhagat, Sruti Gan Chaudhuri, and Krishnendu Mukhopadyaya. Gathering of opaque robots in 3d space. In *Proceedings of the 19th International Conference on Distributed Computing and Networking*, pages 1–10, 2018.

[94] Subhash Bhagat, S Gan Chaudhuri, and Krishnendu Mukhopadhyaya. Fault-tolerant gathering of asynchronous oblivious mobile robots under one-axis agreement. *Journal of Discrete Algorithms*, 36:50–62, 2016.

[95] Gabriele Di Stefano and Alfredo Navarra. Gathering of oblivious robots on infinite grids with minimum traveled distance. *Information and Computation*, 254:377–391, 2017.

[96] Ichiro Suzuki and Masafumi Yamashita. Distributed anonymous mobile robots—formation and agreement problems. In *Proc. 3rd International Colloquium on Structural Information and Communication Complexity (SIROCCO'96)*, pages 313–330, 1994.

[97] 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*, pages 267–281, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[98] Quentin Bramas and Sébastien Tixeuil. Arbitrary pattern formation with four robots. In *International Symposium on Stabilizing, Safety, and Security of Distributed Systems*, pages 333–348. Springer, 2018.

[99] Kaustav Bose, Ranendu Adhikary, Manash Kumar Kundu, and Buddhadeb Sau. Arbitrary pattern formation on infinite grid by asynchronous oblivious robots. *Theoretical Computer Science*, 815:213–227, 2020.

[100] Serafino Cicerone, Alessia Di Fonso, Gabriele Di Stefano, and Alfredo Navarra. Arbitrary pattern formation on infinite regular tessellation graphs. *Theoretical Computer Science*, 942:1–20, 2023.

[101] Shantanu Das, Paola Flocchini, Nicola Santoro, and Masafumi Yamashita. Forming sequences of geometric patterns with oblivious mobile robots. *Distributed Computing*, 28(2):131–145, 2015.

[102] Yukiko Yamauchi and Masafumi Yamashita. Pattern formation by mobile robots with limited visibility. In *International Colloquium on Structural Information and Communication Complexity*, pages 201–212. Springer, 2013.

[103] Kaustav Bose, Ranendu Adhikary, Manash Kumar Kundu, and Buddhadeb Sau. Arbitrary pattern formation by opaque fat robots with lights. In *Conference on Algorithms and Discrete Applied Mathematics*, pages 347–359. Springer, 2020.

[104] Tamás Lukovszki and Friedhelm Meyer auf der Heide. Fast collisionless pattern formation by anonymous, position-aware robots. In *International Conference on Principles of Distributed Systems*, pages 248–262. Springer, 2014.

[105] Yukiko Yamauchi and Masafumi Yamashita. Randomized pattern formation algorithm for asynchronous oblivious mobile robots. In *International Symposium on Distributed Computing*, pages 137–151. Springer, 2014.

[106] Ramachandran Vaidyanathan, Gokarna Sharma, and Jerry Trahan. On fast pattern formation by autonomous robots. *Information and Computation*, 285:104699, 2022.

[107] 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)*, pages 1151–1161. IEEE, 2022.

[108] Xavier Défago and Akihiko Konagaya. Circle formation for oblivious anonymous mobile robots with no common sense of orientation. In *Proceedings of the second ACM international workshop on Principles of mobile computing*, pages 97–104, 2002.

[109] Nao Fujinaga, Hirotaka Ono, Shuji Kijima, and Masafumi Yamashita. Pattern formation through optimum matching by oblivious corda robots. In *International Conference On Principles Of Distributed Systems*, pages 1–15. Springer, 2010.

[110] Subhash Bhagat, Sruti Gan Chaudhuri, and Krishnendu Mukhopadhyaya. Mutual visibility for asynchronous robots. In *International Colloquium on Structural Information and Communication Complexity*, pages 336–339. Springer, 2019.

[111] Subhash Bhagat and Krishnendu Mukhopadhyaya. Mutual visibility by robots with persistent memory. In *International Workshop on Frontiers in Algorithmics*, pages 144–155. Springer, 2019.

[112] Gokarna Sharma, Ramachandran Vaidyanathan, Jerry L Trahan, Costas Busch, and Suresh Rai. Complete visibility for robots with lights in o (1) time. In *International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pages 327–345. Springer, 2016.

[113] Subhash Bhagat. Optimum algorithm for the mutual visibility problem. In *International Workshop on Algorithms and Computation*, pages 31–42. Springer, 2020.

[114] Gokarna Sharma, Rusul Alsaedi, Costas Busch, and Supratik Mukhopadhyay. The complete visibility problem for fat robots with lights. In *Proceedings of the 19th International Conference on Distributed Computing and Networking*, pages 1–4, 2018.

[115] Pavan Poudel, Gokarna Sharma, and Aisha Aljohani. Sublinear-time mutual visibility for fat oblivious robots. In *Proceedings of the 20th International Conference on Distributed Computing and Networking*, pages 238–247, 2019.

[116] Gokarna Sharma, Ramachandran Vaidyanathan, and Jerry L Trahan. Optimal randomized complete visibility on a grid for asynchronous robots with lights. *International Journal of Networking and Computing*, 11(1):50–77, 2021.

[117] Rory Hector, Ramachandran Vaidyanathan, Gokarna Sharma, and Jerry L Trahan. Optimal convex hull formation on a grid by asynchronous robots with lights. *IEEE Transactions on Parallel and Distributed Systems*, 2022.

[118] Andreas Cord-Landwehr, Bastian Degener, Matthias Fischer, Martina Hüllmann, Barbara Kempkes, Alexander Klaas, Peter Kling, Sven Kurras, Marcus Märtens, Friedhelm Meyer auf der Heide, et al. A new approach for analyzing convergence algorithms for mobile robots. In *International Colloquium on Automata, Languages, and Programming*, pages 650–661. Springer, 2011.

[119] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. Computational geometry. In *Computational geometry*, pages 1–17. Springer, 1997.