Network localization and Pattern formation in some Network topologies under Discrete domain

Doctoral dissertation submitted by

Manash Kumar Kundu

for the award of the PhD degree of

JADAVPUR UNIVERSITY

Kolkata, India

Prof. Buddhadeb Sau

Department of Mathematics

Jadavpur University

Kolkata, India

JANUARY 2023

Certificate from the Supervisor

This is to certify that this thesis entitled "*Network localization and Pattern formation in some Network topologies under Discrete domain*" submitted by **Mr. Manash Kumar Kundu** who got his name registered on **29th January, 2016** for the award of Ph.D (Science) degree of Jadavpur University, is absolutely based upon his own work under the supervision of **Dr. Buddhadeb Sau, Professor, Department of Mathematics, Jadavpur University**, and that neither this thesis nor any part of it has been submitted for either any degree/ diploma or any other academic award anywhere before under my knowledge.

.....05/07/2023

(Signature of the Supervisor with date and official seal)

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

Dedicated To

MY PARENTS

Preface

This thesis is submitted at Jadavpur University, Kolkata 700032, India for the degree "*Doctor of Philosophy*" in science. The research described herein is conducted under the supervision of Prof. Buddhadeb Sau at the Department of Mathematics, Jadavpur University, in the time period between January, 2016 and January, 2023.

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

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

- Manash Kumar Kundu, Pritam Goswami, Satakshi Ghosh and Buddhadeb Sau. **Arbitrary pattern formation by opaque fat robots on infinite grid**. International Journal of Parallel, Emergent and Distributed Systems, Vol. 37, pages 542-570, 2022. https://doi.org/10.1080/17445760.2022.2088750

- Kaustav Bose, Manash Kumar Kundu, Ranendu Adhikary and Buddhadeb Sau. **Optimal Gathering by Asynchronous Oblivious Robots in Hypercubes**. Algorithms for Sensor Systems. ALGOSENSORS 2018. Lecture Notes in Computer Science(), vol 11410. Springer, Cham. https://doi.org/10.1007/978-3-030-14094-6_7

- Kaustav Bose, Manash Kumar Kundu, Ranendu Adhikary and Buddhadeb Sau. **Distributed Localization of Wireless Sensor Network Using Communication Wheel**.Information and Computation, 2022,104962,ISSN 0890-5401, https://doi.org/10.1016/j.ic.2022.104962.

  An earlier version of the paper appeared in Algorithms for Sensor Systems. ALGOSENSORS 2020. Lecture Notes in Computer Science(), vol 12503. Springer, Cham. https://doi.org/10.1007/978-3-030-62401-9_2

- Manash Kumar Kundu, Pritam Goswami, Satakshi Ghosh and Buddhadeb Sau. **Arbitrary pattern formation by asynchronous opaque robots on infinite grid**. Arxiv e-prints, May. 2022. arXiv:2205.03053. https://doi.org/10.48550/arXiv.2205.03053 (Communicated)

Manash Kumar Kundu

Date: 06/07/2023

.....

Manash Kumar Kundu

Acknowledgment

First I would like to express my sincere gratitude to Prof. Buddhadeb Sau whom I had the privilege to have as my supervisor. For the past five years or so, he has been a fatherly figure in my life. I am immensely grateful to him for his support, advice and encouragement. I will cherish the numerous insightful conversations that I have had with him related to research and beyond.

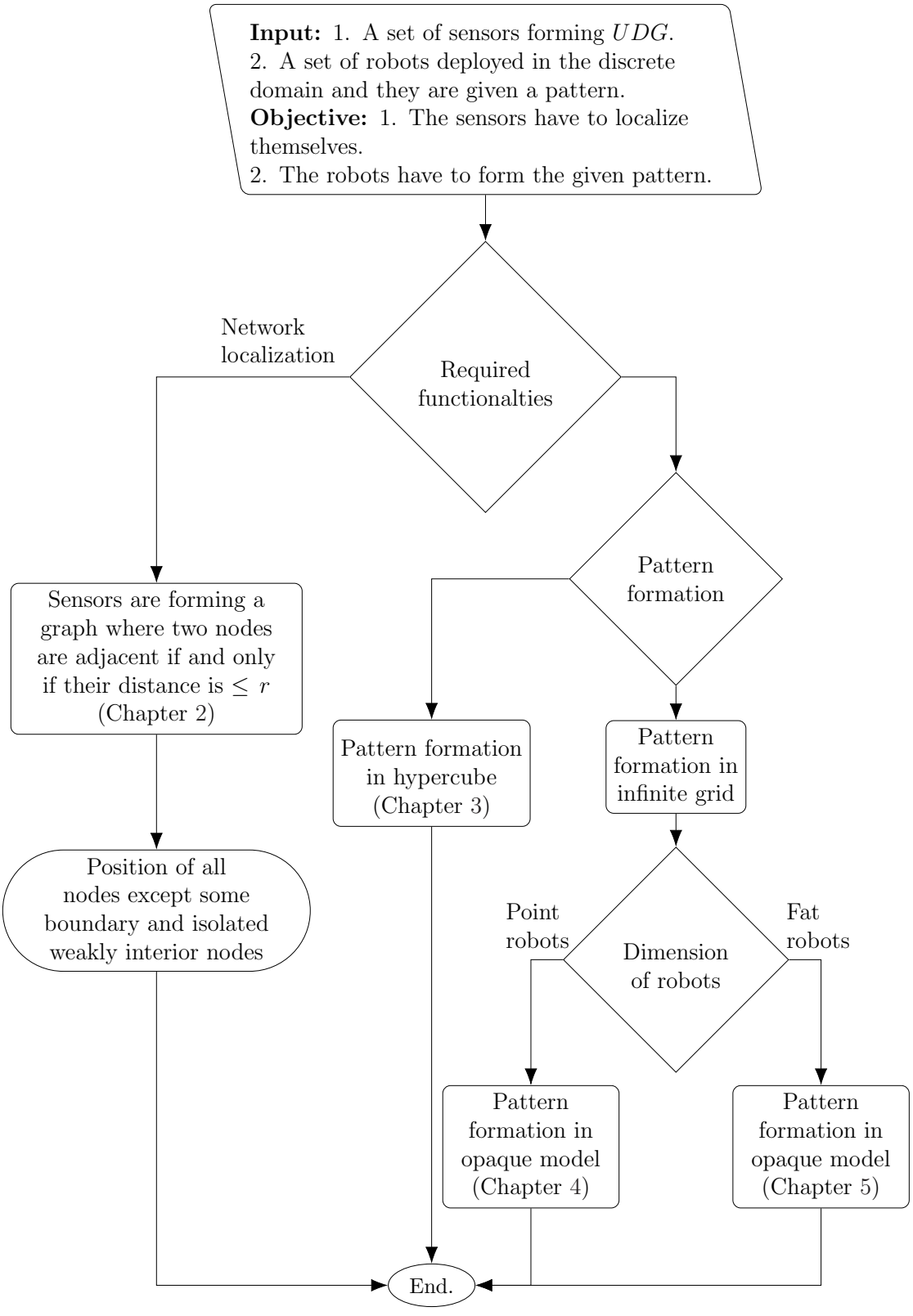
I am grateful to all the teachers of Department of Mathematics, Jadavpur University who have taught me. I would also like to thank all the members and office staffs of Jadavpur University for all their help.

Next I would like to thank Kaustav Bose, Ranendu Adhikary, Pritam Goswami and Satakshi Ghosh, with whom I have coauthored several papers in the past few years. I am extremely fortunate to have as labmates Sangita Patra, Ananya Saha, Arpita Dey, Suvankar Barai, Debajyoti Biswas, Archak Das, Raja Das, Avisek Sharma and Brati Mondal.

I would like to thank my family members, especially my parents and my sister for their unconditional love, blessings and many sacrifices. Without them none of this would have been possible.

Finally, I would like to acknowledge Jadavpur University, Kolkata for their assistance and UGC, Government of India for their financial support in the form of research fellowship to carry out this work.

Flowchart of the thesis



Contents

Chapter 1

Introduction

A *wireless sensor network* (WSN) is a wireless network consisting of a large number of small autonomous sensors spatially distributed in a region to monitor physical or environmental parameters. The sensor nodes are low-cost, low-power, autonomous, multi-functional devices equipped with sensing, processing, and communication capabilities. The knowledge of the physical location of sensor nodes is essential in many applications where the geographical information of the sensed data is important, for example, event detection, environment and habitat monitoring, target tracking, pervasive medical care, etc. The positional information of the nodes also supports many fundamental location-aware protocols, like geographic routing, topology control, coverage, etc. One method of determining the location of the nodes is by equipping the sensor nodes with Global Positioning System (GPS). However, the installation of GPS on each node of a large scale WSN is expensive and the power consumption of GPS reduces the battery life of the sensor nodes. Moreover, it is not suitable in dense forests, underground or indoor environment where GPS signals are unavailable. Therefore, novel schemes have been proposed to determine the positions of the nodes in a network where only some special nodes called *anchors* are aware of their positions with respect to some global coordinate system (e.g., [3, 7, 8, 19, 80, 93]). In these schemes, the nodes can measure the distances to their neighboring nodes and then try to determine their positions by using the distance information. This process of computing the positions of the nodes is called *range based network localization* or

simply *network localization*. In centralized setting, there is a central unit or base station involved. The inter-node distance information measured by the nodes is transmitted to the central unit where the locations of the nodes are calculated. In distributed setting, there is no central unit or base station. In this case, the nodes are required to find their positions based on the local information obtained from their neighbors.

The network localization problem can be abstracted as the following: given a weighted graph with edge weights equal to the distances between the respective nodes and coordinates of some nodes, called anchors, with respect to some coordinate system, we have to compute the coordinates of all other nodes in that coordinate system. A network, with the given positions of anchors and distances between adjacent nodes, is said to be *uniquely localizable* if all nodes of the network have unique positions consistent with the given data, i.e., there is a unique solution. Obviously, if the given instance corresponds to multiple feasible solutions, the actual positions of the nodes cannot be determined. The unique localizability of a network is completely determined by certain combinatorial properties of the network graph and the number of anchors. *Graph rigidity theory* [39, 55, 58] provides the following necessary and sufficient condition for unique localizability [39]: a network is uniquely localizable if and only if it has at least 3 anchors and the network graph is *globally rigid* (See Section 2.3 for definition). However, unless a network is highly dense and regular, it is unlikely that the network is globally rigid. But even if a network is not globally rigid as a whole, a large portion of the network may be globally rigid. For the remaining nodes, there are multiple feasible solutions and hence, their actual positions cannot be determined. In the decision version of the problem, also known as Graph Embedding or Graph Realization problem, given a weighted graph, we have to determine whether there is an embedding of the graph in the plane so that the distances between the adjacent vertices are equal to the edge weights. This problem has been shown to be strongly NP-hard [94]. In [39], it is shown that the problem remains NP-hard even when the graph is globally rigid. However, these results are for general graphs. In a sensor network, only nodes that are within a certain communication range, say r, can measure their relative dis-

tances. Therefore, the network can be better modeled as a unit disk graph: two nodes are adjacent if and only if their distance is $\leq r$. In this version of the problem, apart from the coordinates of the anchors and the distances between the adjacent nodes, we have a third type of information: the distances between the non-adjacent nodes are $> r$. The decision version of this problem, also known as Unit Disk Graph Reconstruction problem, is that given a weighted graph with weights $\leq r$, we have to determine whether there is an embedding of the graph in the plane so that 1) the distances between the adjacent vertices are equal to the edge weights, and 2) the distance between any pair of non-adjacent nodes is $> r$. It is shown in [4] that Unit Disk Graph Reconstruction is NP-hard. Therefore, there is no efficient algorithm that solves the localization problem in the worst case unless P = NP. It is further shown in [4] that a similar result holds even for instances that have unique reconstructions: there is no efficient randomized algorithm that solves the localization problem even for instances that have unique reconstructions unless RP = NP. So we are interested in algorithms that effciently localize the network partially.

Now if instead of autonomous sensors, they are autonomous mobile robots, then the distributed system is called robot swarm where they collaboratively execute some complex tasks. Swarms of low-cost, weak, simple robots are emerging as a viable alternative to using a single powerful and expensive robot. Before the study of swarm robotics, designing a robot to do a specific task was costly as it would have needed many strong capabilities. But designing a swarm of robots is cheaper than using such robot with many capabilities as the goal now become to design the robots with minimal capabilities such that they can do the same task autonomously. Among many applications of swarm robots military operations, border surveillance, cleaning of a large surface, rescue operations, disaster management etc. are the ones that use the swarm robots vividly in present days. So, it is evident why swarm robotics has gained such popularity in the industry and among researchers in the current scenario.

Among many problems (eg. gathering, scattering, exploration) Arbitrary Pattern Formation ($\mathcal{APF}$) is a classical problem in the field of swarm robotics.

The problem is to design an algorithm which will be used by each autonomous mobile robot of a robot swarm that will guide the robots to form any specific pattern that is given to the robots initially as input. In this problem, the robots are modeled as *autonomous* (there is no central control), *anonymous* (the robots do not have any unique identifiers), *homogeneous* (all robots execute the same algorithm) and *identical* (robots are indistinguishable by their appearance). All the robots can freely move on the plane. Each robot has sensing capability by which they can perceive the location of other robots on the plane. The robots do not have any global coordinate system (each robot has its own local coordinate system) and they operate in Look-Compute-Move (LCM) cycles. In the Look phase, a robot takes a snapshot of its surroundings. In the Compute, phase a robot process the information got from the Look phase and in the Move, phase a robot moves to another position (a robot also might stay still in this phase) depending on the output of the Compute phase. In any model, the robots can be considered as transparent or opaque. In the case of transparent robots, a robot can see another robot even if there are other robots between them. But in the case of opaque robots, a robot can not see another robot if there are other robots between them. There are many works where both these models have been considered ([1, 6, 12–14, 16, 33, 47, 72]). Opaque robots can be considered to be dimensionless (i.e point robots) ([1, 6, 13, 14, 27, 42, 47, 53, 75]) or they can have some dimension (i.e fat robots) [12]. In the literature on $\mathcal{APF}$, there are many works which have considered the robots to be dimensionless (i.e point robots) and opaque [1, 14].

1.1 Theoretical Framework

This section provides an overview of the theoretical framework under which the computational and complexity issues related to distributed computing by robot swarms are studied. The framework covers a large spectrum of settings. The different sets of assumptions regarding the capabilities of the robots and the environment in which they operate give rise to several variations of the framework.

1.1.1 Sensors

The mathematical model of wireless sensor network considered in this work is described in the following:

- A set of n sensors is arbitrarily deployed in $\mathbb{R}^2$. Each sensor node has computation and wireless communication capabilities.
- There is a constant $r > 0$, called the *communication range*, such that any two sensor nodes can directly communicate with each other if and only if the distance between them is $\leq r$. This implies that the corresponding communication network can be modeled as a *unit disk graph (UDG)*: two nodes are adjacent if and only if they are at most r distance apart. We assume that this graph is connected. Note that if the graph is not connected, then it is impossible to localize the entire network consistently.
- The distance between a pair of sensors can be measured directly and accurately if and only if they are at most r distance apart. Hence, if a sensor node can directly communicate with another node, then it also knows the distance between them.
- The sensor nodes are assumed to be in general positions, i.e., no three nodes are collinear. This is not a major assumption, as the nodes of a randomly deployed network are almost always in general positions. This assumption implies that if the distances of a node from any three other nodes in the network with known coordinates (with respect to some coordinate system) are known, then the coordinates of the former can also be computed.

1.1.2 The Robots

In robot swarm, a set of n mobile computational entities is called *robots* or *agents*. The robots can perform local computations. They are *oblivious* (they have no memory of past configurations and previous actions), *autonomous* (there is no central control), *homogeneous* (they execute the same distributed algorithm),

anonymous (they have no unique identifiers) and *identical* (they are indistinguishable by their appearance). They can be *point robots* or *fat robots*. *Point robots* are modelled as dimensionless points and *fat robots* are modelled as a disk of some positive diameter.

1.1.3 Network Topologies under Discrete Domain

In this case, they operate in a network modeled as a graph. The examples of some network topologies under discrete domain is given below:

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

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

Grid: The infinite two-dimensional grid $\mathcal{G}$ is a weighted graph $\mathcal{G} = (V, E)$ such that each node $v \in V$ has four adjacent nodes v_0, v_1, v_2 and $v_3 \in V$ and the edges $vv_{i \pmod 4} \in E$ is perpendicular to the edge $vv_{i+1 \pmod 4} \in E$. Also, the weight of each edge $e \in E$ is basically the length of the edge e.

Hypercube: The *d-dimensional hypercube* Q_d is an undirected graph with vertex set $V(Q_d) = \mathbb{Z}_2^d = \{0, 1\}^d$, and two vertices are adjacent if and only if the two binary strings differ in exactly one coordinate. An *oriented hypercube* is an edge-labeled hypercube with the so-called *dimensional labeling* $\lambda : E(Q_d) \to \{1, \ldots, d\}$ where $\lambda(uv) = i$, if u and v differ in the ith coordinate. We shall denote an oriented hypercube by $Q_d^{\mathcal{O}}$, and an unoriented hypercube by simply Q_d.

1.1.4 Multiplicity Detection Capability

Consider a set of robots placed on the vertices of a simple undirected connected graph $G = (V, E)$. Define a function $f : V \longrightarrow \mathbb{N} \cup \{0\}$, where $f(v)$ is the number

of robots on the vertex v. The pair (G, f) is called a *configuration of robots on* G, or simply a *configuration*. Given a configuration (G, f), we define the *multiplicity function* $\tilde{f}$ in the following way:

Strong multiplicity detection capability: If the model assumes robots with strong multiplicity detection capability, then $\tilde{f}(v) = f(v)$ for all $v \in V$.

Week multiplicity detection capability: If the robots have weak multiplicity detection capability, then $\tilde{f} : V \longrightarrow \{0, 1, 2\}$ is defined as,

$$\tilde{f}(v) = \begin{cases} 0 & \text{if } v \text{ is an empty vertex} \\ 1 & \text{if } v \text{ is occupied by exactly one robot} \\ 2 & \text{if } v \text{ is a multiplicity.} \end{cases}$$

No multiplicity detection capability: If the robots have no multiplicity detection capability, then $\tilde{f} : V \longrightarrow \{0, 1\}$ is defined as,

$$\tilde{f}(v) = \begin{cases} 0 & \text{if } v \text{ is an empty vertex} \\ 1 & \text{if } v \text{ is occupied by at least one robot.} \end{cases}$$

1.1.5 LOOK-COMPUTE-MOVE Cycle

The robots, when active, operate according to the so-called Look-Compute-Move cycle. In each cycle, a previously idle or inactive robot wakes up and executes the following steps. In the Look phase, the robot takes the snapshot of the positions of all the robots, represented in its own coordinate system. Based on the perceived configuration, the robot performs computations according to a deterministic algorithm to decide whether to stay idle or to move. Based on the outcome of the algorithm, the robot either remains stationary or makes an instantaneous move.

1.1.6 Activation and Synchronization

Based on the activation and timing of action of the robots, there are three models: fully synchronous ($\mathcal{FSYNC}$), semi-synchronous ($\mathcal{SSYNC}$) and asynchronous

($\mathcal{ASYNC}$). In the fully synchronous setting (FSYNC), the activation phase of all robots can be logically divided into global rounds, where all the robots are activated in each round. The semi-synchronous (SSYNC) model coincides with the FSYNC model with the only difference that not all robots are necessarily activated in each round. The most general type of scheduler is the asynchronous scheduler (ASYNC). In ASYNC, the robots are activated independently, and the amount of time spent in Look, Compute, Move and inactive states are finite but unbounded and unpredictable. As a result, the robots do not have a common notion of time.

1.1.7 Visibility

There are three types of visibility models of the robots.

Full visibility: The most commonly used model is the *full visibility* model. In this model, each robot is able to observe all robots in the team.

Limited visibility: In the *limited visibility* model, the visibility range of each robot is limited. Formally, there is a finite distance $d > 0$ such that two robots can see each other if and only if the distance between them is $\leq d$.

Obstructed visibility: In the *obstructed visibility* or *opaque robot* model, the visibility range of each robot is unlimited, but its visibility can be obstructed by the presence of other robots. Formally, for point robots, two robots are able to see each other if and only if no other robot lies on the line segment joining them. And for fat robots, a robot r_i can see another robot r_j if and only there is a point p_{r_j} on the boundary of r_j and p_{r_i} on the boundary of r_i such that the line segment $\overline{p_{r_i} p_{r_j}}$ does not intersect with any point occupied by other robots in the configuration.

1.1.8 Local Coordinate Systems

The robots do not have access to any global coordinate system. However, the local coordinate systems of the robots may have some consistency. There are four

main models based on the level of consistency:

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

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

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

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

1.1.9 Models regarding communication capabilities

There are mainly four models of robots depending upon their capabilities. These models are $\mathcal{OBLOT}$, $\mathcal{FSTA}$, $\mathcal{FCOM}$ and $\mathcal{LUMI}$. In all of these models, robots are considered to be autonomous, homogeneous, identical and anonymous (i.e the robots do not have any unique identifiers). In the $\mathcal{OBLOT}$ model, the robots are considered to be oblivious (i.e the robots do not have any persistent memory to remember any previous state) and silent (i.e the robots do not have any means of communication among themselves). In the $\mathcal{FSTA}$ model, the robots are silent but not oblivious. In the $\mathcal{FCOM}$ model, the robots are oblivious but not silent. And in the $\mathcal{LUMI}$ model, the robots are neither silent nor oblivious. There are many works of $\mathcal{APF}$ which has considered the $\mathcal{OBLOT}$ model in literature [1, 12–14, 47].

.

1.2 Related Works

There is a large body of work concerning network localization and pattern formation. In this section, we give a brief survey of these results. The readers are

also referred to the surveys [20, 74, 78, 110] for more details.

The localization algorithms can be classified with respect to various criteria, e.g., centralized vs distributed, range based vs range free, anchor based vs anchor free etc. In centralized setting, there is a central unit or base station involved. The data collected from the whole network is transmitted to the central unit where the locations of the nodes are calculated. In distributed setting, the nodes are required to find their positions based on the local information obtained from their neighbors. In range based algorithms, the nodes have hardware capabilities that allow them to measure their distances from neighboring nodes. Algorithms that do not rely on distance information belong to the class of range free algorithms. Anchor based algorithms assume that there are some special nodes in the network that know their coordinates with respect to some global coordinate system (with the help of GPS modules or by manual pre-programming at the time of deployment). Anchor free algorithms, on the other hand, do not require the existence of any anchor nodes.

Range based localization: Perhaps the most widely used technique in the range based settings is trilateration. In trilateration, a node calculates its position from its distances from three nodes with known coordinates. The popularity of trilateration is due to its simplicity and applicability in a wide range of scenarios. For example, trilateration can be implemented in both centralized and distributed settings. It can be also implemented in the anchor free case where some three mutually adjacent nodes of the network will play the role of virtual anchors by fixing their coordinates (respecting their mutual distances) in some virtual coordinate system. If the distance measurements are accurate, then the position of a node can be accurately calculated from its distances from three nodes with known coordinates by computing the point of intersection of three circles. However, distance measurements, usually obtained through received signal strength (RSS) based methods or time-of-arrival (ToA) based methods, are inaccurate. In this case, a more accurate estimation of location may be obtained if distance estimations from more than three localized nodes can be used. In

other words, even if the distances from the localized nodes are not accurate, a larger number of reference nodes should be able to reduce the influence of the errors. This generalization of trilateration is called *multilateration*. Multilateration estimates the location of the node by minimizing the differences between the measured distances and estimated distances. This can be done by solving a minimum mean square estimate (MMSE) [93] or using Taylor Series Expansion to transform the non-linear least squares problem to a linear problem [49, 106]. One phenomenon that may occur due to the measurement errors while attempting trilateration is *flip ambiguity*. In flip ambiguities, a sensor node having a set of neighbors that are almost collinear, may lead to the possibility of the neighbors forming a mirror through which the sensor node can be reflected, thereby causing a large localization error [62]. As the algorithm progresses, its impact can propagate in an avalanche fashion affecting the location estimates of a large portion of the network. Techniques to resolve flip ambiguities have been studied extensively in [62, 81, 98]. In order to handle flip ambiguity, trilateration or multilateration is applied after choosing a subset of neighboring localized nodes based on different robustness criteria proposed in these papers. Use of multilateration in case of three dimensional sensor networks has also been studied in [3].

A network can be fully localized using trilateration iff it has a *trilateration ordering*, i.e., there exists an ordering of the nodes as $v_1, v_2, \ldots, v_n$ so that v_1, v_2 and v_3 induces K_3 and each v_j, $j > 3$ is adjacent to at least three nodes of $v_1, v_2, \ldots, v_{j-1}$. A generalization of trilateration ordering is *bilateration ordering*, i.e., an ordering of the nodes as $v_1, v_2, \ldots, v_n$ so that v_1, v_2 and v_3 induces K_3 and each v_j, $j > 3$ is adjacent to at least two nodes of $v_1, v_2, \ldots, v_{j-1}$. There exist uniquely localizable networks that do not have trilateration ordering, but have bilateration ordering. However, not all uniquely localizable networks have a bilateration ordering. In [40], an algorithm called Sweeps was proposed that localizes uniquely localizable networks having bilateration ordering. However, the algorithm is centralized and takes exponential time in the worst case. An adaptation of the Sweeps algorithm for the case of inaccurate distance measurements was presented in [41].

There are a large number of network localization heuristics based on *multidimensional scaling* (MDS). MDS is a dimensionality reduction technique used for exploring similarities or dissimilarities in data. In centralized MDS based algorithms, the central unit collects the distance matrix of the network which contains inter-node distance of each pair of nodes (with distances between non-neighboring nodes estimated by their shortest path distances obtained by Dijkstra or Floyd Warshall algorithm). Then MDS method is applied to the distance matrix to obtain a relative global map of the network which is then transformed to an absolute map based on the positions of anchor nodes. MDS can be also used in range free setting (e.g., [96]), i.e., using only connectivity information, by estimating the distances between the pair of nodes based on shortest path hop-distances between the nodes. Due to the high computational overhead in centralized MDS based schemes, several semi-centralized or clustered MDS based schemes have proposed (e.g., [95, 112]). In these schemes, the network is divided into clusters in which local maps are built by cluster heads and then merged together to form a global one by a central unit. In [28], a distributed MDS based strategy was proposed where the non-anchor nodes have imperfect a-priori knowledge of their locations which they update over time based on data obtained from neighboring sensors. Readers are referred to [91] for a comprehensive survey on MDS based localization techniques.

Another popular approach is to formulate the network localization problem as an optimization problem and solve the problem using nonlinear optimization techniques. In particular, the original non-convex problem can be relaxed to a Quadratic Programming (QP) problem and then solve it by transforming it into a standard Semi-Definite Programming (SDP) [7, 9] or a Second-Order Cone Programming (SOCP) [105] problem. Although these approaches are primarily centralized, some distributed implementations have been suggested as well (e.g., [8, 99]). The original nonlinear, non-convex optimization problem can also be solved using popular randomized search heuristics such as Evolutionary Algorithms (EA), Simulated Annealing (SA), Particle Swarm Optimization (PSO) etc (c.f. [21, 63, 84, 109]).

Another class of localization techniques is based on angle of arrival (AoA) information between neighbor nodes. Here, it is assumed that the nodes are capable of detecting the angles of incoming signals. AoA is defined as the angle between the propagation direction of an incoming signal and some reference direction. When the information of the reference direction of a node is known with respect to some global coordinate system, then location of the node can be found when it receives signals from at least two localized nodes. When no information about the reference direction is available, signals from at least three localized nodes are needed to localize the node. This basic technique is called *triangulation*. However, more advanced techniques are required for localization from noisy AoA measurements. Interested readers are referred to [71, 82, 83, 89] more on AoA based localization.

There are also a number of studies [39, 52, 116] on testing network localizability, i.e., checking if a network is uniquely localizable or not, or finding a uniquely localizable portion of a network. Recall however that a uniquely localizable network may not be efficiently localizable due to the hardness results. There is also a considerable body of work on mobile anchor assisted localization schemes (e.g., [56, 59, 67, 73, 85, 97, 100, 113, 113]). In these settings, there is a mobile robot (or group of mobile robots) equipped with a GPS unit. The robot is required to move through the sensor field and localize the sensor nodes or assist the sensor nodes to localize themselves.

Pattern formation: The problem of Arbitrary Pattern Formation was first introduced in [103] and after that this problem has been studied many times in the literature ([11, 15, 17, 18, 22, 23, 34, 37, 43, 48, 50, 76, 108, 114, 115]). In a plane, the robots can move freely in any direction. So collision can be avoided by the robots by comparably easy techniques. So $\mathcal{APF}$ in specific network topologies under discrete domain(eg. grid, ring, hypercube etc.) is quite interesting itself where it is not so easy to avoid collisions. In [12], the authors have considered this problem and produced an algorithm with robots having full visibility on an infinite grid in $\mathcal{OBLOT}$ model.

Initially the pattern formation problem has been studied only assuming that the robots do not have obstructed visibility. But in a more practical setting, when more than two robots are in a straight line, a robot with camera sensors can only see its adjacent robots (at most two robots). These robots are known as opaque robots. In [15], $\mathcal{APF}$ has been solved considering opaque robots with visible lights which can assume 6 persistent colors. They have also assumed one axis agreement for each robots. This model of luminous robots has first been introduced in [88] by Peleg et al. The visible lights can be used by robots as a means of communication and persistent memory.

In most of these works, the basic assumption was that the robots are points and they do not have obstructed visibility. But in a practical application-based scenario designing a point robot is impossible because every physical object has a certain dimension. So in [12], authors have considered a swarm of fat and opaque robots and shown that this swarm can form any given pattern from any asymmetric configuration without collision under the $\mathcal{LUMI}$ model using 10 colours in a plane. But again it has been studied in plane. In [13], an algorithm for $\mathcal{APF}$ has been provided for a swarm of point robots on an infinite grid but considering full and Unobstructed visibility. Now in [1], considering obstructed visibility model the authors have shown that a circle can be formed on an infinite grid from any initial configuration if the opaque point robots in the swarm have one-axis agreement and 7 colours. Then, in [72] authors have presented another algorithm where a swarm of opaque point robots on an infinite grid can form the given pattern in finite time using one-axis agreement and 8 colours. But none of these works considered fat robots on infinite grid and solve the problem of $\mathcal{APF}$. To the best of our knowledge, there is no work till now which has considered fat robots on infinite grid and provided any algorithm for arbitrary pattern formation on the grid. Since designing a collision-free algorithm in plane is easier than handling collision in discrete domain, many researchers became interested to study the problem of $\mathcal{APF}$ in discrete domain.

Gathering: The gathering (a specific pattern, namely point formation) problem has been extensively studied in continuous domain under various assumptions [2, 24–26, 29, 46, 86, 90]. This problem has been studied in different network topologies under discrete domain [5, 30, 35, 38, 44, 45, 64, 65, 68, 77, 79, 102]. The problem of gathering two robots on an anonymous ring was studied in [36, 69, 79]. The problem for more than two robots was studied in [45]. In [45], the robots had memory and used tokens to break symmetry. In [65], the problem was first considered in a very minimal setting with identical, asynchronous, memoryless robots without tokens or any kind of communication capability. They proved that without multiplicity detection, gathering is impossible on rings for $n \geq 2$ robots. With weak multiplicity detection capability, they solved the problem for all configurations with an odd number of robots, and all the asymmetric configurations with an even number of robots by different algorithms. In [64], symmetric configurations with an even number of robots were studied, and the problem was solved for more than 18 robots. Some of the remaining configurations were solved in [31, 54, 66] in separate algorithms. In [32], a single unified algorithm was proposed, that achieves gathering for all gatherable initial configurations except some potentially gatherable configurations with 4 robots. The problem was studied with weak local multiplicity detection in [57, 60, 61]. A full characterization of gatherable configurations for finite grids and trees with weak multiplicity detection was provided in [30]. Gathering in finite grids in presence of crash-faults was studied in [10]. Optimal gathering in infinite grid with strong multiplicity detection was studied in [101].

1.3 Overview of the Thesis

The main focus of study of the thesis is network localization problem and pattern formation problem in some network topologies under discrete domain. The problems have been considered under the distributed environment.

In Chapter 2, we study the *network localization problem*, i.e., the problem of determining node positions of a wireless sensor network modeled as a unit disk

graph. In an arbitrarily deployed network, positions of all nodes of the network may not be uniquely determined. Computational complexity results suggest that even if the network corresponds to a unique solution, a polynomial-time algorithm is unlikely to exist. So we are interested in algorithms that efficiently localize the network partially. A widely used technique that can efficiently localize a uniquely localizable portion of the network is *trilateration*: starting from three *anchors* (nodes with known positions), nodes having at least three localized neighbors are sequentially localized. However, the performance of trilateration can substantially differ for different choices of the initial three anchors. In this chapter, we proposed a distributed localization scheme with a theoretical characterization of nodes that are guaranteed to be localized. In particular, our proposed distributed algorithm starts localization from a *strongly interior node* and provided that the subgraph induced by the strongly interior nodes is connected, it localizes all nodes of the network except some *boundary nodes* and *isolated weakly interior nodes*.

In Chapter 3, we consider the problem of gathering a set of autonomous, identical, oblivious, asynchronous, mobile robots at a vertex of an anonymous hypercube. The robots operate in Look-Compute-Move cycles. In each cycle, a robot takes a snapshot of the current configuration (*Look*), then based on the perceived configuration, decides whether to stay idle or to move to an adjacent vertex (*Compute*), and in the later case makes an instantaneous move accordingly (*Move*). We have shown that the problem is unsolvable if the robots do not have multiplicity detection capability. With weak multiplicity detection capability, the problem is solvable in an oriented hypercube for any initial configuration of $2k + 1 (k > 0)$ number of robots. For $4k (k > 0)$ number of robots, the problem is solvable under the same assumptions if and only if the group of automorphism of the configuration is trivial. Our proposed algorithms are optimal with respect to the total number of moves executed by the robots.

In Chapter 4, we are solving $\mathcal{APF}$ on infinite grid with asynchronous opaque robots with lights. Arbitrary pattern formation ($\mathcal{APF}$) by mobile robots is studied by many in literature under different conditions and environment. Recently it has been studied on an infinite grid network but with full visibility. In opaque

robot model, circle formation on infinite grid has also been studied. In this chapter, we considered arbitrary pattern formation on infinite grid in the same opaque model. The robots do not share any global co-ordinate system. The main challenge in this problem is to elect a leader to agree upon a global co-ordinate where the vision of the robots are obstructed by other robots. Since the robots are on a grid, their movements are also restricted to avoid collisions. In this chapter, the aforementioned hardness are overcome to produce an algorithm that solves the problem.

In Chapter 5, we are solving $\mathcal{APF}$ on infinite grid with opaque fat robots with lights. Many works regarding $\mathcal{APF}$ have been proposed on plane and infinite grid by point robots. But in practical application, it is impossible to design point robots. In [12], the robots are assumed opaque fat robots but the environment is plane. To the best of our knowledge, no work till now ever considered the $\mathcal{APF}$ problem assuming opaque fat robots on infinite grid where movements are restricted. In this chapter, we have provided a collisionless distributed algorithm and solved $\mathcal{APF}$ using 9 colors.

Finally in Chapter 6, we discuss some directions for future research.

Chapter 2

Distributed Localization of Wireless Sensor Network Using Communication Wheel

In this chapter[1], we are interested in distributed, anchor free, range based localization schemes. Since a real life instance may not have unique solution and even if it has, it is unlikely that there is an efficient algorithm that solves the problem, we look for an efficient algorithm that partially localize the network. A very popular technique is *trilateration* which efficiently localizes a globally rigid subgraph of the network. It is based on the simple fact that the position of a node can be determined from its distance from three non-collinear nodes with known coordinates. The algorithm starts with at least three anchor nodes and then nodes adjacent to at least three nodes with known coordinates are sequentially localized. It is computationally efficient and very easy to implement in distributed setting, thus widely used in practice. In this chapter, we are interested in *anchor-free localization*, i.e., there are no anchor nodes. Since at least three anchor nodes are necessary for localization, in the anchor-free case, some three mutually adjacent nodes of the network fix their coordinates (respecting

[1]Based on this chapter, the following paper has been published:
Kaustav Bose, Manash Kumar Kundu, Ranendu Adhikary and Buddhadeb Sau. **Distributed Localization of Wireless Sensor Network Using Communication Wheel**. Information and Computation, 2022, 104962, ISSN 0890-5401. https://doi.org/10.1016/j.ic.2022.104962

their mutual distances) in some virtual coordinate system. These three nodes play the role of anchors. However, in case of trilateration, the performance of the algorithm can drastically differ for different choices of the initial three nodes.

The example in Fig. 2.1 shows how substantially the performance of trilateration varies with different choices of the three anchor nodes. An even more extreme situation is shown in Fig. 2.2 where there is a choice of anchors from which

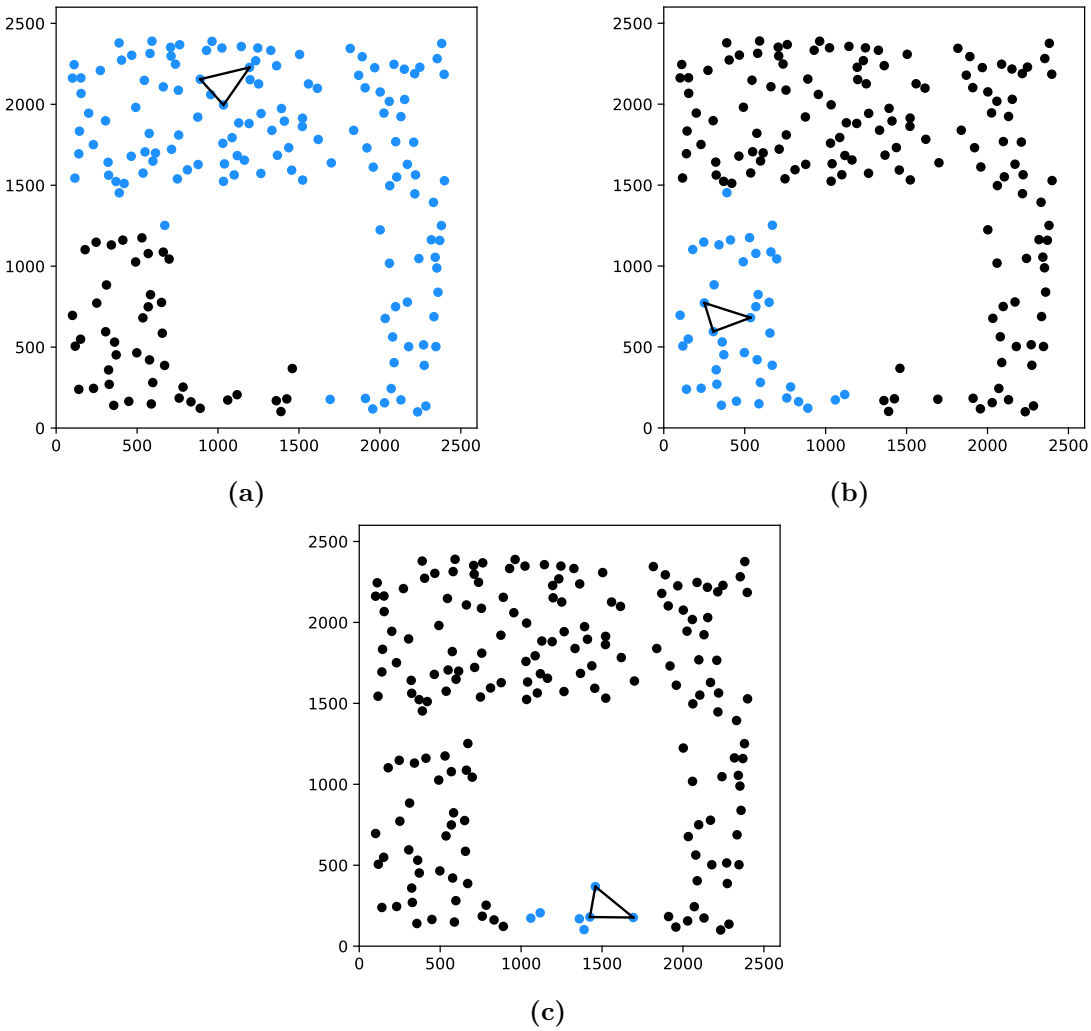


Figure 2.1: Trilateration is attempted from different initial triangles of anchor nodes. The light blue nodes are successfully localized, and the black nodes are not localized. a) In the best result, 137 nodes out of 180 are localized. b) In this case, only 41 nodes out of 180 are localized. c) In the worst case, only 7 nodes out of 180 are localized.

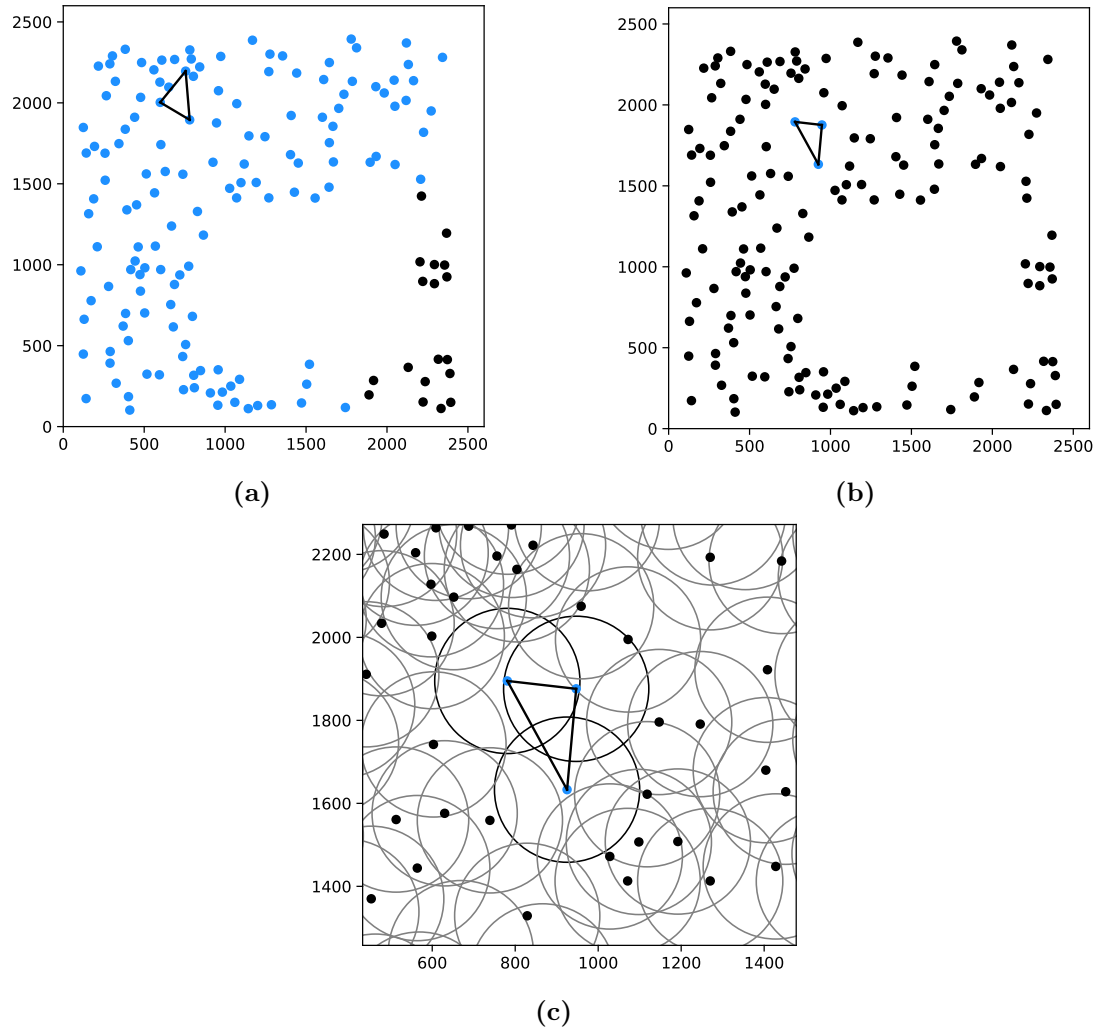


Figure 2.2: Trilateration is attempted from different initial triangles of anchor nodes. The light blue nodes are successfully localized, and the black nodes are not localized. a) In the best result, 142 nodes out of 160 are localized. b) In the worst case, trilateration does not progress beyond the initial triangle. c) A closer view of (b), where the disks centered at the nodes have radii equal to half of the communication range. Hence, two nodes are adjacent in the network iff the corresponding disks intersect. One can see that no node is adjacent to all three anchor nodes.

the algorithm does not progress at all. We address this issue in this chapter. Of course, there is a brute-force method of overcoming this problem: all inter-node distance information is sent to a leader node (elected by a leader election protocol) which will act as a central unit, i.e., it will calculate the positions of

the nodes using trilateration. In particular, since it has the entire network graph, it can try trilateration from different triangles and find one that performs best or reasonably well. However, this creates a huge computational overhead at that node. This makes the approach impractical since the sensor nodes usually have limited computational resources (as compared to a powerful central unit in a centralized system). So instead we propose a distributed anchor-free localization scheme with a theoretical characterization of nodes that are guaranteed to be localized. We define three types of nodes in the network: *strongly interior nodes*, *weakly interior nodes* and *boundary nodes*. Intuitively, boundary nodes are on the fringe of the network. The rest are called interior nodes. An interior node is called a strongly interior node if all its neighbors are also interior nodes. A weakly interior node is an interior node that has at least one boundary node as neighbour. We first give an algorithm that allows a node to determine its type. Provided that the *strong interior* (i.e., the subgraph induced by the set of strongly interior nodes) is connected, one strongly interior node is then chosen by a leader election protocol. Our localization algorithm then starts from that strongly interior node. We prove that our algorithm is guaranteed to localize all nodes in the network except some boundary nodes and isolated weakly interior nodes (i.e., weakly interior nodes that are not adjacent to any strongly interior node).

Organization In Section 2.1, 2.2 and 2.3, we describe the model and present some definitions, notations and basic results. In Section 2.4, we introduce some constructions and prove some results which are the main building blocks of our algorithm. Then in Section 2.5, we present our main algorithm.

2.1 Basic Model and Assumptions

The mathematical model of wireless sensor network considered in this work is described in the following:

- A set of n sensors is arbitrarily deployed in $\mathbb{R}^2$. Each sensor node has

computation and wireless communication capabilities.

- There is a constant $r > 0$, called the *communication range*, such that any two sensor nodes can directly communicate with each other if and only if the distance between them is $\leq r$. This implies that the corresponding communication network can be modeled as a *unit disk graph (UDG)*: two nodes are adjacent if and only if they are at most r distance apart. We assume that this graph is connected. Note that if the graph is not connected, then it is impossible to localize the entire network consistently.

- The euclidean distance between a pair of sensors can be measured directly and accurately if and only if they are at most r distance apart. Hence, if a sensor node can directly communicate with another node, then it also knows the distance between them.

- The sensor nodes are assumed to be in general positions, i.e., no three points are collinear. This is not a major assumption, as the nodes of a randomly deployed network are almost always in general positions.

2.2 Definitions and Notations

Let $\mathcal{V}$ be the set of n sensors at distinct positions in $\mathbb{R}^2$. The corresponding wireless sensor network can be modeled as an undirected edge-weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$, where

- $\mathcal{V} = \{v_1, \ldots, v_n\}$ is the set of sensors,

- $(v_i, v_j) \in \mathcal{E}$, i.e., v_i is adjacent to v_j, if and only if $d(v_i, v_j) \leq r$, where r is the communication range of the sensors,

- the edge-weight $w : \mathcal{E} \longrightarrow \mathbb{R}$ is given by $w(v_i, v_j) = d(v_i, v_j)$.

We call $\mathcal{G}$ the *underlying network graph* of the wireless sensor network. As mentioned previously, we assume that the graph $\mathcal{G}$ is connected.

For any $v \in \mathcal{V}$, $\mathcal{Z}(v)$ is the disk $\{x \in \mathbb{R}^2 \mid d(x, v) \leq \frac{r}{2}\}$. We shall say that a node $v \in \mathcal{V}$ *covers* a point $p \in \mathbb{R}^2$, if $p \in \mathcal{Z}(v)$. For $v \in \mathcal{V}$, its *neighborhood* is the set $\mathcal{N}(v) = \{v' \in \mathcal{V} \setminus \{v\} \mid d(v', v) \leq r\} = \{v' \in \mathcal{V} \setminus \{v\} \mid \mathcal{Z}(v') \cap \mathcal{Z}(v) \neq \emptyset\}$. If $v, v' \in \mathcal{V}$ are adjacent to each other, then we shall refer to the intersections of $\partial(\mathcal{Z}(v))$ and $\partial(\mathcal{Z}(v'))$ as their *boundary intersections*. We shall denote these boundary intersections as $CW(v, v')$ and $CCW(v, v')$ according to the following rule: if one traverses from $CCW(v, v')$ to $CW(v, v')$ along $\partial(\mathcal{Z}(v))$ in clockwise direction, it sweeps an angle $< \pi$ about the center v. Given a node v, we define a partial order relation $\preceq_v$ on $\mathcal{N}(v)$ as following: for $u, u' \in \mathcal{N}(v)$, $u \preceq_v u'$ if and only if $\mathcal{Z}(u) \cap \partial(\mathcal{Z}(v)) \subseteq \mathcal{Z}(u') \cap \partial(\mathcal{Z}(v))$. See Fig. 2.3. A node $u \in \mathcal{N}(v)$ is said to be a *maximal* neighbor of v if it is a maximal element in $\mathcal{N}(v)$ with respect to $\preceq_v$, i.e., there is no $u' \in \mathcal{N}(v) \setminus \{u\}$, such that $u \preceq_v u'$.

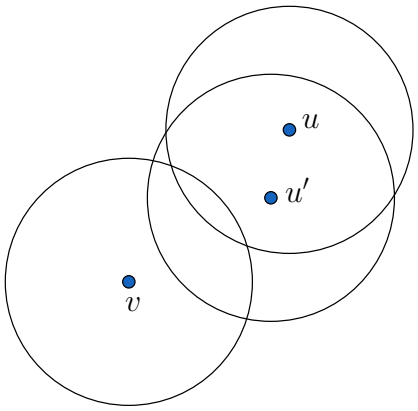


Figure 2.3: u is not a maximal neighbor of v as $u \preceq_v u'$.

A sensor node $v \in \mathcal{V}$ is called an *interior node* if for every point $z \in \partial(\mathcal{Z}(v))$, where $\partial(\mathcal{Z}(v))$ is the boundary of $\mathcal{Z}(v)$, we have $z \in \mathcal{Z}(v')$ for some $v' \in \mathcal{V} \setminus \{v\}$. If $v \in \mathcal{V}$ is not an interior node, then it is called a *boundary node*. An interior node $v \in \mathcal{V}$ is said to be *a strongly interior node* if every node in $\mathcal{N}(v)$ is an interior node. An interior node $v \in \mathcal{V}$ is said to be *a weakly interior node* if at least one node in $\mathcal{N}(v)$ is a boundary node. A weakly interior node is said to be *isolated* if it is not adjacent to any strongly interior node. The subgraph of $\mathcal{G}$ induced by the set of all interior nodes is called the *interior* of $\mathcal{G}$. Similarly, the subgraph of $\mathcal{G}$ induced by the set of all strongly interior nodes is called the *strong*

interior of $\mathcal{G}$.

2.3 Some Results from Graph Rigidity Theory

In this section, we present some basic definitions and results in graph rigidity. For a detailed exposition on graph rigidity, the readers are referred to [58].

A d-dimensional *framework* is a pair (G, ρ), where $G = (V, E)$ is a connected simple graph and the *realization* ρ is a map $\rho : V \longrightarrow \mathbb{R}^d$. Two frameworks (G, ρ_1) and (G, ρ_2) are said to be *equivalent* if $d(\rho_1(u), \rho_1(v)) = d(\rho_2(u), \rho_2(v))$, for all $(u, v) \in E$. Frameworks (G, ρ_1) and (G, ρ_2) are said to be *congruent* if $d(\rho_1(u), \rho_1(v)) = d(\rho_2(u), \rho_2(v))$, for all $u, v \in V$. In other words, two frameworks are said to be congruent if one can be obtained from another by an isometry of $\mathbb{R}^d$. A realization is *generic* if the vertex coordinates are algebraically independent over rationals. The framework (G, ρ) is *rigid* if $\exists$ an $\varepsilon > 0$ such that if (G, ρ') is equivalent to (G, ρ) and $d(\rho(u), \rho'(u)) < \varepsilon$ for all $u \in V$, then (G, ρ') is congruent to (G, ρ). Intuitively, it means that the framework cannot be continuously deformed. (G, ρ) is said to be *globally rigid* if every framework which is equivalent to (G, ρ) is congruent to (G, ρ). It is known [111] that rigidity is a *generic property*, that is, the rigidity of (G, ρ) depends only on the graph G, if (G, ρ) is generic. The set of generic realizations is dense in the realization space and thus almost all realizations of a graph are generic. So, we say that a graph G is rigid in $\mathbb{R}^2$ if every generic realization of G in $\mathbb{R}^2$ is rigid.

Theorem 2.1. *[58] A graph G is* globally rigid *in $\mathbb{R}^2$ if and only if either G is a complete graph on at most three vertices or G is 3-connected, rigid and remains rigid even after deleting an edge.*

Theorem 2.2. *[39] If a network has at least 3 anchors and the underlying network graph is globally rigid, then it is uniquely localizable.*

The condition of having at least 3 anchors is also necessary for unique localizability in order to rule out the trivial transformations. Since we are considering the case where there are no pre-existing anchors, some three mutually adjacent nodes

of the network will play the role anchors by fixing their coordinates (respecting their mutual distances) in some virtual coordinate system. The remaining nodes of the network have to find their position according to this coordinate system. It should be noted here that for networks that do not satisfy the condition that two nodes are adjacent if and only if they are within some fixed distance, the condition of having globally rigid underlying network graph is also necessary. In our model, where two nodes are adjacent if and only if the distance between them is at most r, the network can be uniquely localizable even if its underlying network graph is not globally rigid.

2.4 Construction of a Globally Rigid Subgraph Using Communication Wheels

In this section, we shall show that if the strong interior is connected, then the network has a globally rigid subgraph containing all strongly interior nodes, and all non-isolated weakly interior nodes. The proof is constructive and will lead to our localization algorithm presented in Section 2.5.

We first present some results that will be frequently used in the chapter. Lemmas 2.2-2.5 follow from elementary geometric arguments.

Lemma 2.1. *Let v_1 be an interior node and $v_2 \in \mathcal{N}(v_1)$. Then*

1. *$CCW(v_1, v_2) \in \mathcal{Z}(v_3)$ for some $v_3 \in \mathcal{N}(v_1) \setminus \{v_2\}$, such that $CCW(v_1, v_3) \notin \mathcal{Z}(v_2)$,*

2. *$CW(v_1, v_2) \in \mathcal{Z}(v_4)$ for some $v_4 \in \mathcal{N}(v_1) \setminus \{v_2\}$, such that $CW(v_1, v_4) \notin \mathcal{Z}(v_2)$*

Proof. It is sufficient to prove only the first part. We shall prove by contradiction. So, assume that there is no such node in $\mathcal{N}(v_1) \setminus \{v_2\}$. Let $P = CCW(v_1, v_2)$. Let us partition the set of neighbors of v_1 into two sets as: $A = \{v \in \mathcal{N}(v_1) \mid P \in \mathcal{Z}(v)\}$ and $B = \{v \in \mathcal{N}(v_1) \mid P \notin \mathcal{Z}(v)\}$. $A \neq \emptyset$, since $v_2 \in A$. $B \neq \emptyset$, because

the diametrically opposite point of P on $\partial(\mathcal{Z}(v_1))$ must be covered by some node which does not cover P.

Fix the ray $\overrightarrow{v_1P}$ as a reference axis. Now for each $v \in \mathcal{N}(v_1)$, shoot rays from v_1 passing through $CCW(v_1, v)$ for $v \in A$ and $CW(v_1, v)$ for $v \in B$. For each $v \in \mathcal{N}(v_1)$, let θ_v be the angle formed by the corresponding ray measured counterclockwise from the reference axis $\overrightarrow{v_1P}$. Let $\theta = min\{\theta_v \mid v \in B\}$. We must have $\theta > 0$, since for any $v \in B$, $\theta_v > 0$. Also, it implies from our hypothesis that $max\{\theta_v \mid v \in A\} = 0$. Then clearly any point on $\partial(\mathcal{Z}(v_1))$ making an angle in between $(0, \theta)$ with the ray $\overrightarrow{v_1P}$ is not covered by any neighbor of v_1 (See Fig. 2.4). This contradicts the fact that v_1 is an interior node. □

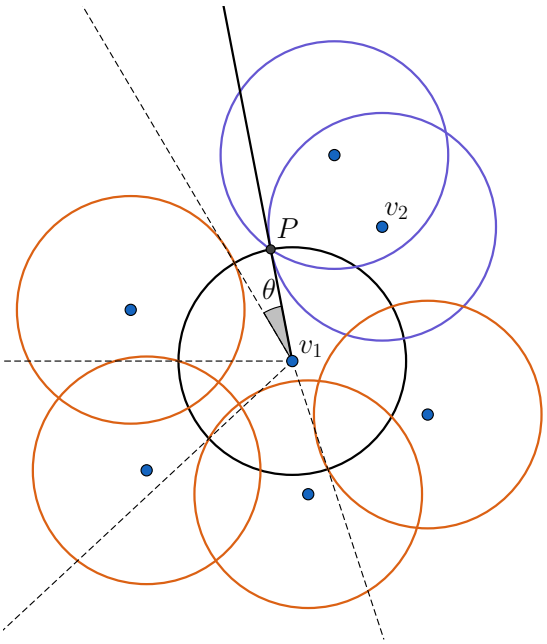


Figure 2.4: Illustration of the constructions in the proof of Lemma 2.1. The purple and the orange circles correspond to $\mathcal{Z}()$ of the nodes in the set A and B respectively.

Lemma 2.2. *If u and u' are two distinct neighbors of $v \in \mathcal{V}$ such that $u \preceq_v u'$, then $d(u, v) > d(u', v)$.*

Proof. It follows from the fact that the length of the intercepted arcs of two intersecting circles increases if the distance between their centers decreases. □

Lemma 2.3. *For distinct* $v, u, u' \in \mathcal{V}$, $u \preceq_v u' \Leftrightarrow v \preceq_u u'$.

Proof. Suppose that $\partial(\mathcal{Z}(u))$ and $\partial(\mathcal{Z}(v))$ intersect each other at points A and B. Then $u \preceq_v u' \Leftrightarrow A, B \in \mathcal{Z}(u') \Leftrightarrow v \preceq_u u'$. □

Lemma 2.4. *For distinct* $v, u \in \mathcal{V}$, u *is a maximal neighbor of* v *if and only if* v *is a maximal neighbor of* u.

Proof. u is a maximal neighbor of $v \Leftrightarrow$ there is no node u' such that $u \preceq_v u' \Leftrightarrow$ there is no node u' such that $v \preceq_u u'$ (by Lemma 2.3) $\Leftrightarrow v$ is a maximal neighbor of u. □

Lemma 2.5. *For distinct* $v, u, u' \in \mathcal{V}$, $u \preceq_v u' \Rightarrow u \not\preceq_{u'} v$.

Proof. Suppose that $\partial(\mathcal{Z}(u))$, $\partial(\mathcal{Z}(v))$ intersect each other at points A and B, and $\partial(\mathcal{Z}(u))$, $\partial(\mathcal{Z}(u'))$ intersect each other at points C and D. Since $u \preceq_v u'$, we have $A, B \in \mathcal{Z}(u')$. For the sake of contradiction, assume that $u \preceq_{u'} v$. Then by Lemma 2.3, $u' \preceq_u v$. So, on $\partial(\mathcal{Z}(u))$, the arc between C, D is contained by the arc between A, B. But this implies that $A, B \notin \mathcal{Z}(u')$. So we have a contradiction. □

A *wheel graph* [107] of order n or simply an *n-wheel*, $n \geq 3$, is a simple graph which consists of cycle of order n and another vertex called the *hub* such that every vertex of the cycle is connected to the hub. The vertices on the cycle are called the *rim vertices*. An edge joining a rim vertex and the hub is called a *spoke*, and an edge joining two consecutive rim vertices is called a *rim edge*. By Theorem 2.1, it follows that a wheel is globally rigid.

The most crucial part of our algorithm is the construction of a special structure called the *communication wheel*. The definition of communication wheel closely resembles to that of *sensing wheel* used in [92], where the authors devised a wheel based centralized sequential localization algorithm for a restricted class of sensing covered networks over a convex region.

Communication wheel: For any interior node $v \in \mathcal{V}$, we define a *communication wheel* of v (see Fig. 2.5) as a subgraph W of $\mathcal{G}$ such that

1. W is a wheel graph with v as the hub and the rim nodes $\{v_1, \ldots, v_m\}$ being maximal neighbors of v

2. $CCW(v, v_i) \in \mathcal{Z}(v_{i+1})$ and $CW(v, v_i) \in \mathcal{Z}(v_{i-1})$, for $i = 1, \ldots, m$, where v_{m+1} means v_1 and v_0 means v_m.

For a rim node v' of a communication wheel W of v, we can denote the two neighboring rim nodes of v' as $CCW_W(v')$ and $CW_W(v')$ so that $CCW(v, v') \in \mathcal{Z}(CCW_W(v'))$ and $CW(v, v') \in \mathcal{Z}(CW_W(v'))$.

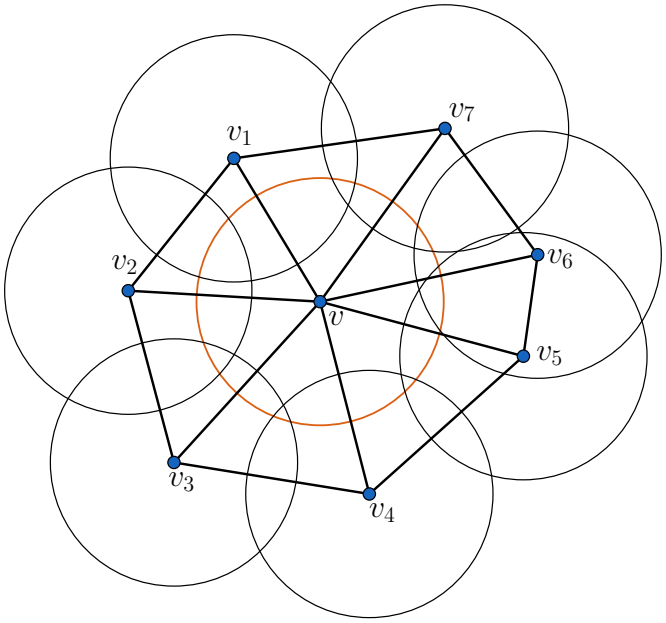


Figure 2.5: A communication wheel of v with rim nodes $v_1, v_2, v_3, v_4, v_5, v_6$ and v_7. The nodes v_5 and v_7 are also but adjacent, but the edge is not shown here.

Lemma 2.6. *If W is a communication wheel of v, then $\partial(\mathcal{Z}(v)) \subset \bigcup_{u \in \mathcal{V}(W) \setminus \{v\}} \mathcal{Z}(u)$.*

Proof. Follows immediately from the definition of communication wheel. □

Theorem 2.3. *If $v \in \mathcal{V}$ is an interior node and v_1 a maximal neighbor of v, then v has a communication wheel W having v_1 as a rim node.*

Proof. First observe that for any maximal neighbor v' of v, $|\partial(\mathcal{Z}(v)) \cap \partial(\mathcal{Z}(v'))| = 2$, i.e., $CW(v, v')$ and $CCW(v, v')$ are distinct points. If not, then suppose that v' is a maximal neighbor of v such that $\partial(\mathcal{Z}(v))$ and $\partial(\mathcal{Z}(v'))$ intersect at a single point, say P. Then by Lemma 2.1, there is another neighbor of v, say v'', such that $P \in \mathcal{Z}(v'')$. Hence we have $v'' \neq v'$, such that $\partial(\mathcal{Z}(v)) \cap \mathcal{Z}(v') = \{P\} \subset \partial(\mathcal{Z}(v)) \cap \mathcal{Z}(v'')$. This contradicts the fact that v' is a maximal neighbor of v.

Now take any maximal neighbor v_1 of v. By Lemma 2.1, choose a maximal $v_2 \in \mathcal{N}(v) \setminus \{v_1\}$, such that $CCW(v, v_1) \in \mathcal{Z}(v_2)$ and $CCW(v, v_2) \notin \mathcal{Z}(v_1)$. Notice that $CW(v, v_1) \notin \mathcal{Z}(v_2)$, because otherwise $v_1 \preceq_v v_2$. Since $CCW(v, v_2) \notin \mathcal{Z}(v_1)$, by again invoking Lemma 2.1, we can choose a maximal $v_3 \in \mathcal{N}(v) \setminus \{v_1, v_2\}$, such that $CCW(v, v_2) \in \mathcal{Z}(v_3)$ and $CCW(v, v_3) \notin \mathcal{Z}(v_2)$. Continuing in this manner, after some m steps we shall find $v_m \in \mathcal{N}(v) \setminus \{v_1, \ldots, v_{m-1}\}$, such that $CCW(v, v_{m-1}) \in \mathcal{Z}(v_m)$ and $CW(v, v_1) \in \mathcal{Z}(v_m)$. It is easy to see that a communication wheel of v can be formed with $\{v_1, \ldots, v_m\}$ as rim nodes. □

Corollary 2.3.1. *$v \in \mathcal{V}$ is an interior node if and only if it has a communication wheel.*

Lemma 2.7. *Let $v \in \mathcal{V}$ be an interior node and W be a communication wheel of v. If $u \in \mathcal{V}$ is a neighbor of v, then u is either a rim node of W or adjacent to some rim node of W.*

Proof. If u is a rim node of W, then we are done. Otherwise, take a point $P \in \partial(\mathcal{Z}(v)) \cap \mathcal{Z}(u)$. Now there is some rim node w of W such that $P \in \mathcal{Z}(w)$. Hence $\mathcal{Z}(u) \cap \mathcal{Z}(w) \neq \emptyset$, which means that u and w are adjacent. □

Lemma 2.8. *Let $v \in \mathcal{V}$ be an interior node and W be a communication wheel of v. If $u \in \mathcal{V}$ is a neighbor of v, which is adjacent to exactly one rim node of W, say u', then $u \preceq_v u'$.*

Proof. If u itself is a rim node of W, then it is adjacent to two other rim nodes of W. But u is adjacent to exactly one rim node of W. So, we can conclude that u is not a rim node of W. Assume for the sake of contradiction that $u \not\preceq_v u'$. So there is a point $P \in \partial(\mathcal{Z}(v))$ such that $P \in \mathcal{Z}(u) \cap \mathcal{Z}(u')^c$. Now there is some

rim node w ($\neq u'$) of W such that $P \in \mathcal{Z}(w)$. Hence $\mathcal{Z}(u) \cap \mathcal{Z}(w) \neq \emptyset$, which means that u and w are adjacent. This contradicts the fact that u is adjacent to exactly one rim node of W. So we have $u \preceq_v u'$. □

Lemma 2.9. *Let $v \in \mathcal{V}$ be a strongly interior node and u a neighbor of v. If W_1 is a communication wheel of v, then there is a globally rigid subgraph of $\mathcal{G}$ containing v, u and W_1.*

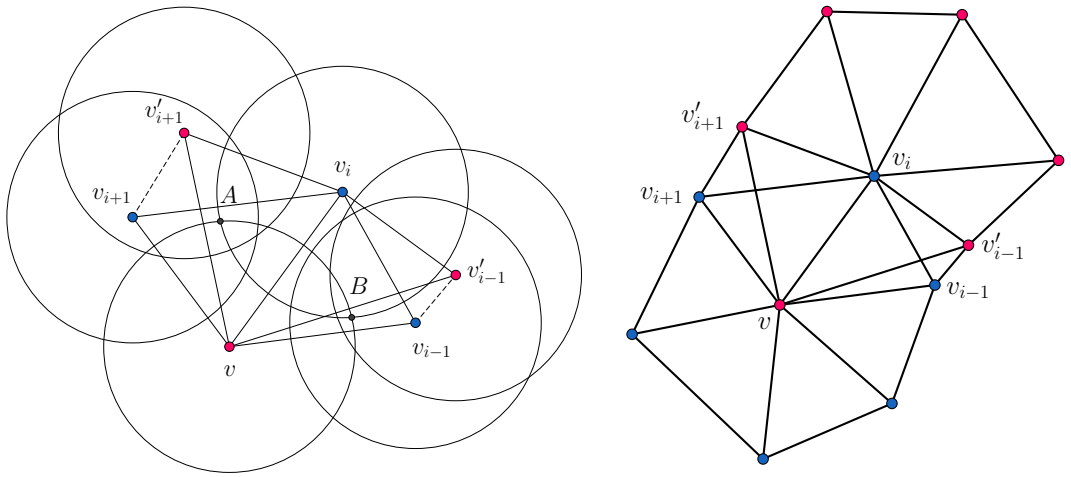
Proof. If u is a rim node of W_1, then we are done, since a wheel graph is globally rigid. So, suppose that u is not a rim node of W_1.

Then by Lemma 2.7, u is adjacent to a rim node of W_1, say v_i. If u is adjacent to another rim node, then u can be added to W_1 to form a globally rigid graph. Hence, we assume that u is adjacent to only one rim node of W_1, i.e., v_i. Then by Lemma 2.8, we have $u \preceq_v v_i$.

Since v is a strongly interior node and v_i is a neighbor of v, v_i must be an interior node. Also, since v_i is a maximal neighbor of v, v is also a maximal neighbor of v_i, by Lemma 2.4. Hence, by Theorem 2.3, v_i has a communication wheel W_2 having v as a rim node.

See Fig. 2.6a. Let $v_{i-1} = CW_{W_1}(v_i)$ and $v_{i+1} = CCW_{W_1}(v_i)$. Also, let $v'_{i+1} = CW_{W_2}(v)$ and $v'_{i-1} = CCW_{W_2}(v)$. Now let $A = CCW(v, v_i) = CW(v_i, v)$ and $B = CW(v, v_i) = CCW(v_i, v)$. So we must have $A \in \mathcal{Z}(v_{i+1}) \cap \mathcal{Z}(v'_{i+1})$ and $B \in \mathcal{Z}(v_{i-1}) \cap \mathcal{Z}(v'_{i-1})$. This implies that v_{i-1}, v'_{i-1} and v_{i+1}, v'_{i+1} are adjacent. So v_{i-1} and v_{i+1} can be added to the list of rim nodes of W_2 and construct a wheel W_3 with hub v_i and having v_{i-1}, v, v_{i+1} as rim nodes. Then the two globally rigid graphs W_1 and W_3 have at least three nodes common. In fact, they have at least four common nodes, namely v, v_{i-1}, v_i and v_{i+1}. Hence $W_1 \cup W_3$ is globally rigid. See Fig. 2.6b.

Now it is sufficient to prove that u is adjacent to at least three nodes of $W_1 \cup W_3$. Let $\mathcal{R}im(W_3) = \mathcal{V}(W_3) \setminus \{v_i\}$ be the set of rim nodes of W_3. We have $\partial(\mathcal{Z}(v_i)) \subset \bigcup_{w \in \mathcal{R}im(W_3)} \mathcal{Z}(w)$. Since $u \preceq_v v_i$, we have $u \not\preceq_{v_i} v$, by Lemma 2.5. In other words, $\partial(\mathcal{Z}(v_i)) \cap \mathcal{Z}(u) \not\subseteq \partial(\mathcal{Z}(v_i)) \cap \mathcal{Z}(v)$. Therefore, we have $\partial(\mathcal{Z}(v_i)) \cap \mathcal{Z}(u) \subset$



(a) **(b)**

Figure 2.6: Illustrations supporting the proof of Lemma 2.9.

$\partial(\mathcal{Z}(v_i)) \subset \bigcup_{w \in \mathcal{R}im(W_3)} \mathcal{Z}(w) \Rightarrow (\partial(\mathcal{Z}(v_i)) \cap Z(u)) \bigcap (\bigcup_{w \in \mathcal{R}im(W_3) \setminus \{v\}} \mathcal{Z}(w)) \neq \emptyset$. This implies that u is adjacent to some rim node of W_3 other than v. We already have assumed that u is adjacent to v and v_i. Hence, u is adjacent to at least three nodes of $W_1 \cup W_3$. □

We also have the following two observations from the proof of Lemma 2.9.

Lemma 2.10. *Let $v \in \mathcal{V}$ be a strongly interior node and W_1 a communication wheel of v. If v_i, v_{i+1} be two neighboring rim nodes of W_1 and W_2 be a communication wheel of v_i having v as rim node, then v_{i+1} is in W_2 or adjacent to at least three nodes of W_2.*

Proof. Follows from the proof of Lemma 2.9. □

Lemma 2.11. *Let $v \in \mathcal{V}$ be a strongly interior node and W_1 a communication wheel of v. Suppose that u is a neighbor of v which is adjacent to only one rim node v_i of W_1. If W_2 is a communication wheel of v_i having v as rim node, then u is in W_2 or adjacent to at least three nodes of W_2.*

Proof. Follows from the proof of Lemma 2.9. □

Theorem 2.4. *If $v \in \mathcal{V}$ is a strongly interior node, then there is a subgraph $\mathcal{H}_v$ of $\mathcal{G}$ containing v such that 1) $\mathcal{H}_v$ contains all neighbors of v, 2) $\mathcal{H}_v$ is globally rigid.*

Proof. Let W_1 be a communication wheel of v. Then by Lemma 2.9, for each neighbor of v not in W_1, we obtain a globally rigid subgraph of $\mathcal{G}$ containing the neighbor, v and W_1. So any two of these globally rigid subgraphs have at least three nodes in common. Hence, these graphs constitute to form the desired globally rigid graph. □

Theorem 2.5. *If the strong interior of $\mathcal{G}$ is connected, then $\mathcal{G}$ has a globally rigid subgraph $\mathcal{R}$ which contains 1) all strongly interior nodes, 2) all non-isolated weakly interior nodes.*

Proof. Choose any strongly interior node v. Denote the subgraph of $\mathcal{G}$ consisting of only the node v as $\mathcal{R}_0$. Since the strong interior of $\mathcal{G}$ is connected, every strongly interior node of $\mathcal{G}$ is connected to v by a path consisting strongly interior nodes. The distance of a strongly interior node from v is defined as the smallest length of such a path. Let m be the maximum distance of a strongly interior node from v. We shall prove the theorem by inductively constructing globally rigid subgraphs $\mathcal{R}_0, \mathcal{R}_1, \ldots, \mathcal{R}_m$, where $\mathcal{R}_j$ contains all strongly interior nodes at a distance at most j from v. $\mathcal{R}_0$ is globally rigid as it is only a singleton node. $\mathcal{R}_1$ is constructed using Theorem 2.4. Suppose that $\mathcal{R}_0, \mathcal{R}_1, \ldots, \mathcal{R}_j$, $1 \leq j < m$, are already constructed. Now consider a strongly interior node v' at a distance $j+1$ from v. In a smallest path from v to v', let v' be adjacent to v''. Clearly v'' is in $\mathcal{R}_j$. Since $\mathcal{R}_j$ is globally rigid, v'' is adjacent to at least three nodes v_1, v_2, v_3 in $\mathcal{R}_j$ ($\mathcal{R}_j$ has at least four nodes as it contains the communication wheel of v). By Theorem 2.4, there is a globally rigid graph containing v'' and its neighbors v', v_1, v_2, v_3. Union of this graph and $\mathcal{R}_j$ is globally rigid as there are at least three nodes in common, namely v'', v_1, v_2, v_3, etc. Similarly for each strongly interior node at a distance $j+1$ from v, we extend the subgraph $\mathcal{R}_j$ preserving global rigidity to eventually obtain a globally rigid graph $\mathcal{R}_{j+1}$ containing all the strongly interior nodes at a distance at most $j+1$ from v. The inductive argument leads to the

globally rigid subgraph $\mathcal{R}_m$ which contains all strongly interior nodes in $\mathcal{G}$. Each non-isolated weakly interior node v''' is adjacent to some strongly interior node in $\mathcal{R}_m$. For each such v''', again by the same construction, we can extend $\mathcal{R}_m$ preserving global rigidity to include v''', if it is not already in $\mathcal{R}_m$. The resulting graph is the desired globally rigid subgraph $\mathcal{R}$. □

2.5 The Localization Algorithm

In the beginning, each node messages its neighbor list along their distances from itself to all its neighbors. Therefore, every node $u \in \mathcal{V}$ knows the neighbors of all its neighbors and also if v is a neighbor of u and w is a neighbor of v, then u knows $d(v, w)$ as well. The three main stages of our algorithm are 1) construction of communication wheel, 2) leader election, and 3) propagation. They are discussed in detail in the following subsections.

2.5.1 Construction of Communication Wheel

Each sensor node v starts off computations by executing the Communication-Wheel algorithm. The algorithm finds if the node is interior or boundary, and also constructs a communication wheel if it is interior. A pseudocode description of the procedure is presented in Algorithm 1. The algorithm Communication-Wheel is similar to the constructions used in the proof of the Theorem 2.3. To construct a communication wheel of a node v, if it exists, we first need to find a maximal neighbor. In view of Lemma 2.2, the closest neighbor of a node is

Algorithm 1: CommunicationWheel

```
1  {The node v constructs a communication wheel. If it successfully constructs a
     communication wheel, it declares itself as an interior node, or otherwise a
     boundary node.}
2  Procedure CommunicationWheel(v)
3      w = [ ] ;
4      v.position ⟵ origin;
5      w_0 ⟵ closest neighbor of v ;
6      w_0.position ⟵ on the X-axis according to the distance between v and w_0;
7      w_1 ⟵ the common neighbor of v and w_0 closest to v that covers a
         boundary-intersection of v and w_0, such that w_1 ⋠_v w_0 ;
8      if (no such w_1 is found) then
9          v.type ⟵ boundary;
10         break;
11     else
12         w_1.position ⟵ one of the two possible positions preserving the
             distances from v and w_0 such that w_1 has positive Y-coordinate;
13         CCW(v, w_0) ⟵ the boundary-intersection of v and w_0 covered w_1
14         i = 1 ;
15         do
16             i ⟵ i + 1 ;
17             NextRim(v,w_{i−1},w_{i−2});
18         while (w_i is found & w_i does not cover CW(v, w_0));
19         if (no such w_i is found) then
20             v.type ⟵ boundary;
21         else
22             v.type ⟵ interior;
23         end
24     end
```

guaranteed to be a maximal neighbor. After finding the closest neighbor, call it w_0, v assigns its position on the X-axis and itself at the origin. Then it searches for a common neighbor of v and w_0 that covers a boundary-intersection of v and w_0. If no such node is found, then v is a boundary node. If more than one of such nodes are found, the one closest to v is to be taken. Let us call this node w_1. Now the distance of w_1 from v and w_0 is known. From this data, there are two possible coordinates for w_1, one with positive Y-coordinate and one with negative Y-coordinate. Choose the position for w_1 so that its Y-coordinate is positive. In other words, v sets its local coordinate system in such a way that w_1 gets positive Y-coordinate. See Fig. 2.7a and 2.7b.

Algorithm 2: NEXTRIM

```
1  {Given two consecutive rim nodes w_{i-1} and w_{i-2}, the node v searches for the
     next rim node.}
2  Function NEXTRIM(v,w_{i-1},w_{i-2})
3      D ⟵ r ;
4      for (u ∈ N(w_{i-1}) ∩ N(v)) do
5          if (distance between u and v ≤ D) then
6              if (u is adjacent to w_{i-2}) then
7                  Find the unique position of u using its distances from v, w_{i-1}
                     and w_{i-2};
8                  if (u covers CCW(v, w_{i-1}) and CCW(v, u) is not covered by
                     w_{i-1}) then
9                      w_i ⟵ u;
10                     w_i.position ⟵ the unique position of u determined using
                         its distances from v, w_{i-1} and w_{i-2};
11                     D ⟵ the distance between u and v;
12             else
13                 Find two possible positions of u using its distances from v and
                     w_{i-1};
14                 if (for any of the two possible positions, u covers
                     CCW(v, w_{i-1}) and CCW(v, u) is not covered by w_{i-1}) then
15                     w_i ⟵ u;
16                     w_i.position ⟵ the one of the two possible positions of u
                         for which it covers CCW(v, w_{i-1});
17                     D ⟵ the distance between u and v;
```

Also set the boundary-intersection of v and w_0 that is covered by w_1 as $CCW(v, w_0)$. In other words v sets 'counterclockwise' to be the direction in which if one rotates a ray, from the origin towards the positive direction of the X-axis, by $\frac{\pi}{2}$, it coincides with the positive direction of the Y-axis. While discussing Algorithm 1, 'counterclockwise' and 'clockwise' will always be with respect to the local coordinate system of node executing the algorithm. Note that since w_0 is a maximal neighbor of v, $CW(v, w_0)$ is not covered by w_1. After fixing the positions of w_0 and w_1, the subroutine NEXTRIM (pseudocode presented in Algorithm 2) is recursively called to find the subsequent rim nodes of the communication wheel. Given two consecutive rim nodes w_{i-1} and w_{i-2}, having positions fixed, NEXTRIM(v,w_{i-1},w_{i-2}) finds the next rim node w_i. The program terminates

when either NextRim reports a failure or returns a node that covers $CW(v, w_0)$.

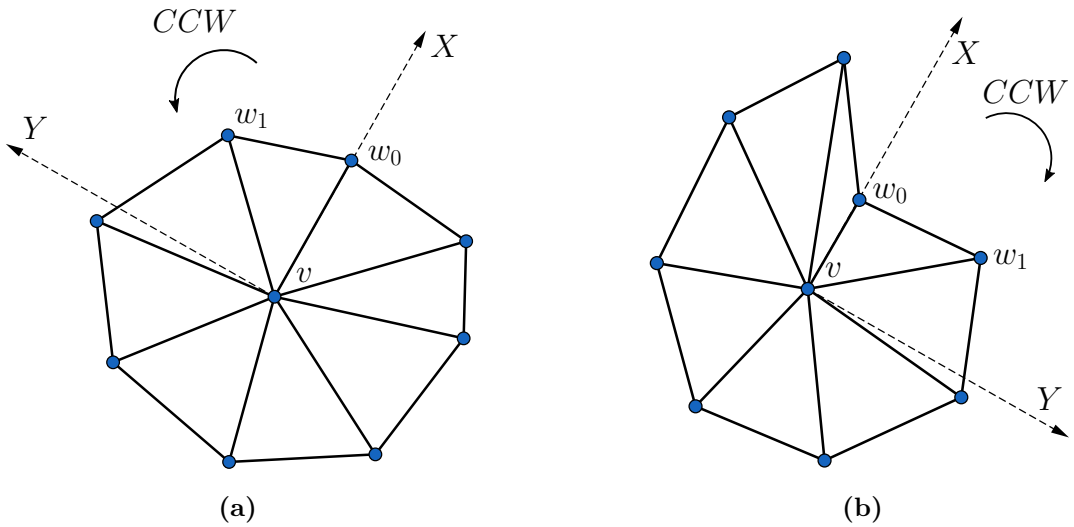


Figure 2.7: A node v executing Algorithm 1 sets its local coordinate system in such a way that w_1 gets positive Y-coordinate.

Theorem 2.6. *The algorithm* CommunicationWheel *is correct, i.e., if v is an interior node then* CommunicationWheel(v) *constructs a communication wheel of v and declares it as an interior node; and otherwise declares it as a boundary node.*

Proof. The algorithm replicates the proof of Theorem 2.3. The algorithm starts off with fixing a maximal neighbor of v as the first rim node w_0. Then it recursively finds the rim nodes w_i such that w_i covers $CCW(v, w_{i-1})$ and $CCW(v, w_i)$ is not covered by w_{i-1}. The algorithm terminates when there is no such w_i or when w_i covers $CW(v, w_0)$. Thus in view of the proof of Theorem 2.3, we only need to show that these steps are correctly executed.

Initialization: The closest neighbor of v is set as w_0. Hence, w_0 is a maximal neighbor of v by Lemma 2.2. Note that in order to compute the communication wheel of v, if it exists, first we need to fix the positions of (i.e., assign virtual coordinates to) at least three nodes of the communication wheel, preserving their

mutual distances. So, first v is assigned with virtual coordinates $(0, 0)$. If the distance between v and w_0 is d, then the coordinates of w_0 are set as $(d, 0)$.

Now we have to check if there is a common neighbor w of v and w_0 that covers a boundary intersection of v and w_0, and such that $w \not\preceq_v w_0$. For any common neighbor w of v and w_0, this can be easily checked from $d(v, w_0), d(v, w)$ and $d(w, w_0)$. If no such w is found, then v is obviously a boundary node. Otherwise one such node that is closest to v is set as the next rim node w_1. From $d(v, w_1)$ and $d(w_0, w_1)$, two possible coordinates of w_1 can be found, one with positive Y-coordinate and one with negative Y-coordinate. Then v sets its local coordinate system in such a way that w_1 gets positive Y-coordinate, and hence w_1 covers $CCW(v, w_0)$. Recall that here counterclockwise and clockwise is defined with respect to the local coordinate system of v as described in Section 2.5.1.

Recursion: After w_0 and w_1 are fixed, the algorithm will recursively call NextRim(v,w_{i-1},w_{i-2}) to find the next rim node w_i, if it exists. In the for loop (line 2 in Algorithm 2), the common neighbors of v and w_{i-1} are scanned through to find nodes $u \in \mathcal{N}(w_{i-1}) \cap \mathcal{N}(v)$ such that u covers $CCW(v, w_{i-1})$ and $CCW(v, u)$ is not covered by w_{i-1}. Among these nodes, the one closest to v is set as the next rim node w_i. If no such node is found, then v is a boundary node. Notice that in order to check whether u covers $CCW(v, w_{i-1})$ or not, the exact position of u needs to be known. As we scan through $\mathcal{N}(w_{i-1}) \cap \mathcal{N}(v)$, there are two cases to consider:

Case 1. Suppose that u is adjacent to w_{i-2}. Now the positions of v, w_{i-1} and w_{i-2} are known. Hence the position of u can ascertained from its distances from v, w_{i-1} and w_{i-2}. Once the position of u is found, it can be checked whether it covers $CCW(v, w_{i-1})$ and also whether w_{i-1} covers $CCW(v, u)$.

Case 2. Suppose that u is not adjacent to w_{i-2}. Two possible positions of u can be found from its distance from v and w_{i-1}. Call these two possible positions U_1 and U_2. U_1 and U_2 are mirror images of each other with respect to the line joining v and w_{i-1}. $CCW(v, w_{i-1})$ and $CW(v, w_{i-1})$ are also mirror images of each other with respect to the line joining v and w_{i-1}. Hence if a node at U_1

covers $CCW(v, w_{i-1})$, then a node at U_2 covers $CW(v, w_{i-1})$ as well. Similarly if a node at U_1 covers neither $CCW(v, w_{i-1})$ nor $CW(v, w_{i-1})$, then the same is true for a node at U_2. So consider the following three possibilities:

Case 2a. If for both positions U_1 and U_2, no boundary intersection between v and w_{i-1} is covered, then u does not meet the desired criteria that it is to cover $CCW(v, w_{i-1})$.

Case 2b. Suppose that for both positions U_1 and U_2, both of the boundary intersections between v and w_{i-1} are covered. This cannot happen as this would imply that $w_{i-i} \preceq_v u$ and hence is adjacent to w_{i-2}, contradicting our assumption.

Case 2c. Suppose that u, if situated at U_1, covers $CCW(v, w_{i-1})$ (and not $CW(v, w_{i-1})$). Hence u, if it is at U_2, would cover $CW(v, w_{i-1})$ and would not cover $CCW(v, w_{i-1})$. In this case, the algorithm determines the position of u to be U_1. We shall prove that U_1 is indeed the correct position of u. Suppose on the contrary that the actual position of u is U_2. First observe that $CW(v, w_{i-1}) \in \mathcal{Z}(w_{i-2}) \cap \partial(\mathcal{Z}(v))$. Otherwise, it implies that $w_{i-2} \preceq_v w_{i-1}$. We argue that this is impossible. For $i = 2$, it is obvious since $w_0 \not\preceq_v w_1$. Recall that w_0 is the closest neighbor, and hence is a maximal neighbor, of v. For $i > 2$, we assume as induction hypothesis that rim nodes $w_{i-1}, \ldots, w_1, w_0$ are successfully found by the algorithm. Also each w_j, for $j = 1, \ldots, i-1$, must satisfy the two criteria: 1) w_j covers $CCW(v, w_{j-1})$ and 2) $CCW(v, w_j)$ is not covered by w_{j-1}. In fact, as mentioned earlier, the algorithm chooses as w_j the closest among the nodes that satisfy these two criteria. So in particular, w_{i-2} is the closest neighbor of v such that 1) w_{i-2} covers $CCW(v, w_{i-3})$ and 2) $CCW(v, w_{i-2})$ is not covered by w_{i-3}. Clearly if $w_{i-2} \preceq_v w_{i-1}$, w_{i-1} also satisfies the two aforesaid conditions. But by Lemma 2.2, w_{i-1} is closer to v than w_{i-2}. This contradicts the fact that w_{i-2} is the closest node satisfying the two aforesaid conditions. So we have $CW(v, w_{i-1}) \in \mathcal{Z}(w_{i-2})$. Also $CW(v, w_{i-1}) \in \mathcal{Z}(u)$ as u is assumed to be at U_2. Hence $\mathcal{Z}(w_{i-2}) \cap \mathcal{Z}(u) \neq \emptyset$. This is a contradiction as u and w_{i-2} are not adjacent.

Termination: If v is a boundary node, then the algorithm will terminate when NextRim will fail to find the next rim node. If v is an interior node, then the algorithm will terminate as eventually NextRim will find a rim node that covers $CCW(v, w_0)$. □

2.5.2 Leader Election

Once a node identifies itself as interior or boundary, it announces the result to all its neighbors. Hence, every node can determine if it is a strongly interior node or not. Since the strong interior is connected and the nodes have unique id's, the strongly interior nodes can elect a leader among themselves. For this, each strongly interior node will collect the id's of all strongly interior nodes in the network using convergecast (see [87], Chapter 3 for details). Then the one with the largest id will elect itself as the leader.

2.5.3 Propagation

Starting from the leader, different nodes will gradually get localized via message passing. The correctness of the process will follow from the discussions in this subsection and the proofs of Theorem 2.4 and 2.5. There are five types of messages that a sensor node can send to another node:

1. "*I am at ...*"
2. "*You are at ...*"
3. "*Construct wheel with me at ... and v at ...*"
4. "*Construct wheel with me at ..., you at ..., v at ... and find u*"
5. "*u is at ...*".

The nodes of the network will be localized in the local coordinate system of the leader v_l set during its execution of Algorithm 1. Henceforth, this coordinate

system will be referred to as the *global coordinate system*. So the leader first localizes itself by setting its coordinates to $(0, 0)$. Any non-leader node u is localized by either receiving a "*You are at ...*" message or receiving at least three "*I am at ...*" messages. In the first case, some node has calculated the coordinates of u and has sent it to u. In the second case, u receives the coordinates of at least three neighbors and therefore, can calculate its own coordinates. When a node is localized, it announces its coordinates to all its neighbors. After setting its coordinates to $(0, 0)$, v_l initiates the localization of $\mathcal{H}_{v_l}$ (See Theorem 2.4). It first announces its coordinates to all its neighbors via the message "*I am at* $(0, 0)$". During the construction of its communication wheel, v_l had assigned coordinates to the rim nodes. So v_l sends these coordinates to the corresponding rim nodes via the message "*You are at ...*". Let us denote the communication wheel of v_l as $\mathcal{W}(v_l)$ and the set of all rim nodes as $\mathcal{R}im(v_l)$. When a rim node receives this message, it sets its coordinates accordingly and announces it to all its neighbors via the message "*I am at ...*". Now if a neighbor of v_l is adjacent to at least two nodes of $\mathcal{R}im(v_l)$, then it can localize itself, since it will receive "*I am at ...*" messages from at least three nodes, i.e., one from v_l and at least two from $\mathcal{R}im(v_l)$. But if a neighbor of v_l is adjacent to only one vertex from $\mathcal{R}im(v_l)$, then it may not be localized. To resolve this, v_l computes $|\mathcal{N}(u) \cap \mathcal{R}im(v_l)|$ for all $u \in \mathcal{N}(v_l)$. If it finds a $u \in \mathcal{N}(v_l)$ with $\mathcal{N}(u) \cap \mathcal{R}im(v_l) = \{v_i\}$, it sends the message "*Construct wheel with me at ... and* v_{i+1} *at ...*" to v_i, where v_{i+1} is a neighboring rim node of v_i in $\mathcal{W}(v_l)$. When v_i receives this message from v_l, it does the following. Since v_l is a strongly interior node, v_i must be an interior node. Therefore, v_i has already computed the communication wheel $\mathcal{W}(v_i)$ and coordinates of each of its nodes with respect to its local coordinate system. Since v_l is a maximal neighbor of v_i (by Lemma 2.4), it is adjacent to at least two nodes of $\mathcal{R}im(v_i)$ (by Lemma 2.8). Hence, v_i can compute the coordinates of v_l with respect to its local coordinate system. Let $\mathcal{W}'$ be the globally rigid graph $v_l \cup \mathcal{W}(v_i)$. $\mathcal{W}'$ contains as subgraph a communication wheel of v_i with v_l as a rim node. Hence from the Lemma 2.10, it is known that v_{i+1} is adjacent to at least three nodes of $\mathcal{W}'$. Hence, v_i can also compute the coordinates of v_{i+1} with respect to its local coordinate system. So, v_i has the coordinates of all nodes of

$\mathcal{W}'' = v_{i+1} \cup \mathcal{W}'$ with respect to its local coordinate system. Now, v_i will compute the positions of all nodes of $\mathcal{W}''$ with respect to the global coordinate system set by v_l. Let us call them the *true positions* of the nodes. Note that v_i knows the true positions of at least three nodes of $\mathcal{W}''$, namely, itself, v_l, and v_{i+1}. With this information, v_i can determine the formula that transforms its local coordinate system to the global coordinate system. Hence, v_i computes the true positions of all nodes in $\mathcal{W}''$ and informs them via "*You are at ...*" messages. Hence, all nodes in $\mathcal{W}''$ will be localized and will announce their locations to all their neighbors. Since u is adjacent to at least three nodes in $\mathcal{W}''$ (from the Lemma 2.11), it will also get localized. Therefore, we see that every neighbor of v_l eventually gets localized.

The localization propagates as each strongly interior node localizes its neighbors. However, a strongly interior node v can compute the positions of its neighbors only with respect to its local coordinate system. Hence, in order to compute the true positions (i.e., to perform coordinate transformation), it needs to know its true position and that of at least two neighbors. Hence, when a localized strongly interior node v receives at least two "*I am at ...*" messages, it starts to localize its neighbors in the following way. Let u be a neighbor of v. If u is adjacent at least two nodes of $\mathcal{R}im(v)$, then v can compute the position of u in terms of its local coordinate system. Otherwise, if $\mathcal{N}(u) \cap \mathcal{R}im(v) = \{v_i\}$, then v sends the message "*Construct wheel with me at ..., you at ..., v_{i+1} at ... and find u*" to v_i, where v_{i+1} is a neighboring rim node to v_i in $\mathcal{W}(v)$, and positions mentioned in the message are given in local coordinates of v. Again, as v_i is a maximal neighbor of v, by Lemma 2.4 and 2.8, v is either in $\mathcal{W}(v_i)$ or adjacent to at least three nodes in $\mathcal{W}(v_i)$. Also, v_{i+1} is either in $\mathcal{W}' = \mathcal{W}(v_i) \cup v$ or adjacent to at least three nodes in $\mathcal{W}'$ (from the Lemma 2.10). Hence, from the message received from v, v_i knows the positions of three nodes (i.e., v, v_i and v_{i+1}) of the globally rigid graph $\mathcal{W}'' = \mathcal{W}' \cup v_{i+1}$ in terms of the local coordinates of v. Hence, v_i can compute the positions of all the nodes in $\mathcal{W}''$ in terms of the local coordinates of v. Now, u is either in $\mathcal{W}''$ or adjacent to at least three nodes in $\mathcal{W}''$ (from Lemma 2.11). So v_i can compute the position of u in terms of the local coordinate system of v, and then sends the information back to v via the message

"*u is at ...*". Hence, v computes the positions of all its neighbors in its local coordinate system. So, v now knows the positions of three nodes (namely, itself and the two nodes from which it has received "*I am at ...*" message) with respect to both its local coordinate system and the global coordinate system set by the leader. Hence, v can find the formula that transformations its local coordinate system to the global coordinate system. Hence, v computes the true positions of all its neighbors and then informs them via "*You are at ...*" messages. However, it still remains to prove that every non-leader localized strongly interior node v always receives at least two "*I am at ...*" messages, that triggers the propagation. Since v is localized, either it has received three "*I am at ...*" messages or one "*You are at ...*" message. If it is the first case, then we are done. In the later case, v receives the "*You are at ...*" message from a localized interior node, say v'. Then by Lemma 2.7, v is either in the communication wheel of v', or adjacent to at least one of its rim nodes. Observe that v' is localized and has also localized all its rim nodes. So, v will get least two "*I am at ...*" messages, as all localized nodes announce their positions. Therefore, all strongly interior nodes and their neighbors, i.e., all non-isolated weakly interior nodes, get localized. Also, if a localized weakly interior node receives "*I am at ...*" messages from at least two nodes, it can localize all the rim nodes of its wheel and also neighbors that are adjacent to at least two rim nodes. However, some boundary nodes and some isolated weakly interior nodes may not get localized. Hence, we have the following main result of this chapter.

Theorem 2.7. *If the strong interior of the network is connected then there is a distributed anchor-free localization algorithm that localizes all nodes of the network except some boundary nodes and isolated weakly interior nodes.*

2.6 Overhead of the algorithm

It is shown in [4] that Unit Disk Graph Reconstruction is NP-hard. Therefore, there is no efficient algorithm that solves the localization problem in the worst case unless P = NP. So we are interested in algorithms that efficiently localize the

network partially. We considered the problem when exact range measurements are available. Localization when the distance measurements are imprecise, is NP-complete. In our algorithm the range measurements are exact and the algorithm solves the localization problem in polynomial time. Now in our work, each node send messages to all its neighbours. The number of messages require to solves the localization problem is total $\mathcal{O}(kD)$ in the worst case, where k is the number of robots and D is the diameter of the network.

2.7 Discussion

In this work, we studied the distributed range based localization problem of wireless sensor networks in the anchor-less setting. The performance of the popular trilateration algorithm substantially varies for different choices of the initial triangle as exhibited in Fig. 2.1 and 2.2. This motivates us to seek an algorithm with some theoretical guarantee. We define three types of nodes in the network: strongly interior nodes, weakly interior nodes and boundary nodes. Intuitively, boundary nodes are on the fringe of the network and the weakly interior nodes make up a layer next to the boundary (see Fig. 2.8a). Provided that the subgraph induced by the set of strongly interior nodes is connected, one strongly interior node is then chosen by a leader election protocol. Our localization algorithm then starts from that strongly interior node and is guaranteed to localize all nodes in the network except some boundary nodes and isolated weakly interior nodes (i.e., weakly interior nodes that are not adjacent to any strongly interior node). In Fig. 2.8 we show an instance where our algorithm is able to localize all nodes in the network.

In Fig. 2.9, we show the performance of our algorithm in the relatively sparse and irregular shaped networks from Fig. 2.1 and 2.2. In both these instances, most nodes of the networks are isolated weakly interior nodes and boundary nodes. Yet our algorithm manages to localize 137 out of 180 nodes and 142 out of 160 nodes respectively. Notice that the performance of our algorithm in both instances match the best performance by trilateration (see Fig. 2.1 and

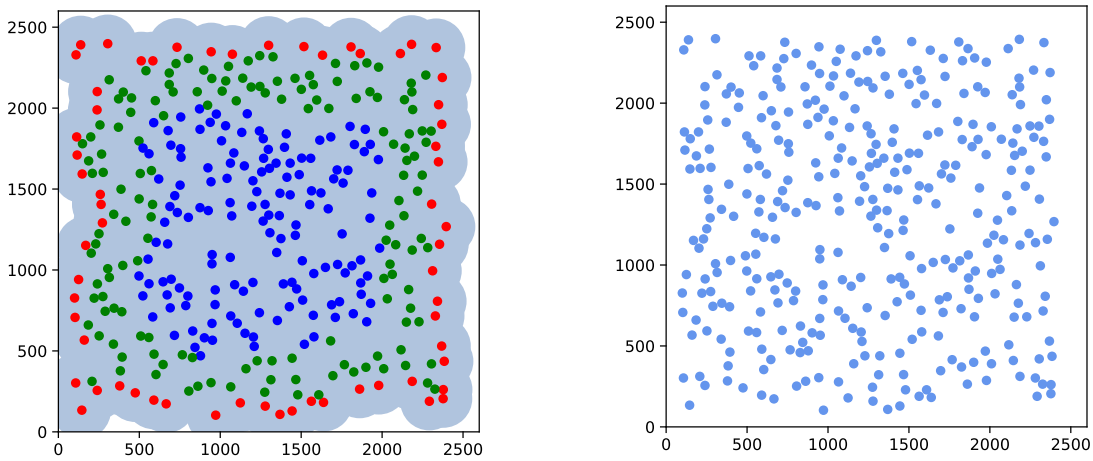


Figure 2.8: a) A random network of 350 sensor nodes. The red, green and deep blue nodes are respectively boundary, weakly interior and strongly interior nodes. Disks of radii $\frac{r}{2}$ around each node is shown in pale blue. b) All 350 nodes are successfully localized by our algorithm.

2.2). Notice that in our algorithm, a node is localized whenever at least three of its neighbors are localized. This is because whenever a node is localized, it announces its coordinates to all its neighbors using "*I am at ...*" messages, and a node localizes itself if it receives "*I am at ...*" messages from at least three neighbors. Hence, in some sense, our algorithm uses trilateration as a subroutine. The fact that the performance of our algorithm in Fig. 2.9 matches exactly with the best performance by trilateration is not surprising as a large number of nodes may be localized by the 'trilateration subroutine' of our algorithm. Since our algorithm uses a 'trilateration subroutine', it may be seen as a strengthening of the plain trilateration and a question that naturally arises is whether it is true that our algorithm always performs at least as good as the plain trilateration. However, this is not true as we can see in this interesting instance shown in Fig. 2.10. When trilateration starts from the upper-right portion of the network, it is not able to progress to the upper-left portion of the network. As a result, only 69 out of 150 nodes are localized. On the other hand, when trilateration starts from the upper-left portion of the network, it is able to progress to the upper-right portion of the network and as a result, it is able to localize 106 out of 150 nodes.

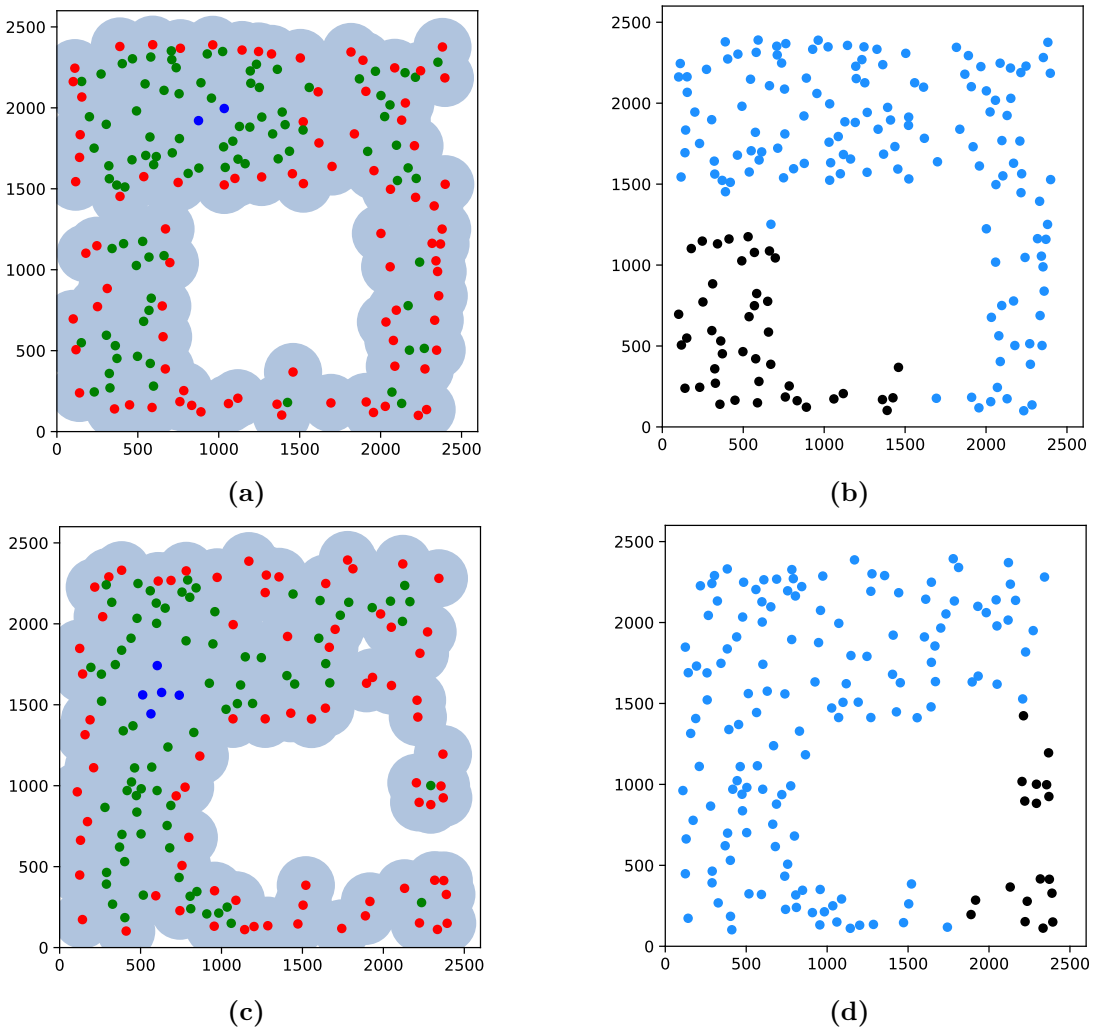


Figure 2.9: a)-b) The network from Fig. 2.1. In 137 nodes out of 180 are localized by our algorithm. c)-d) The network from Fig. 2.2. 142 nodes out of 160 are localized by our algorithm.

To understand why this happens, see in Fig 2.11 the links between the left and right part of the network. Our localization starts from the upper-right portion of the network and also localizes only 69 nodes.

Notice that in the 'irregular shaped' networks of Fig. 2.9 and 2.10, there are only a few strongly interior and non-isolated weakly interior nodes. Hence, the theoretical guarantee of our algorithm in such instances is not very impressive. In nicer instances such as in Fig. 2.8, strongly interior and non-isolated weakly

interior nodes constitute a large portion of the network. The theoretical guarantee of our algorithm is particularly appealing is such networks. However, one may think that trilateration might also perform well in such 'nice' networks if the initial triangle is carefully chosen. In particular, one may think that in case of such networks with large number of strongly interior nodes, trilateration might also perform well if the initial triangle is chosen from the strong interior. In other words, it may appear that a good heuristic could be to first apply Algorithm 1 to

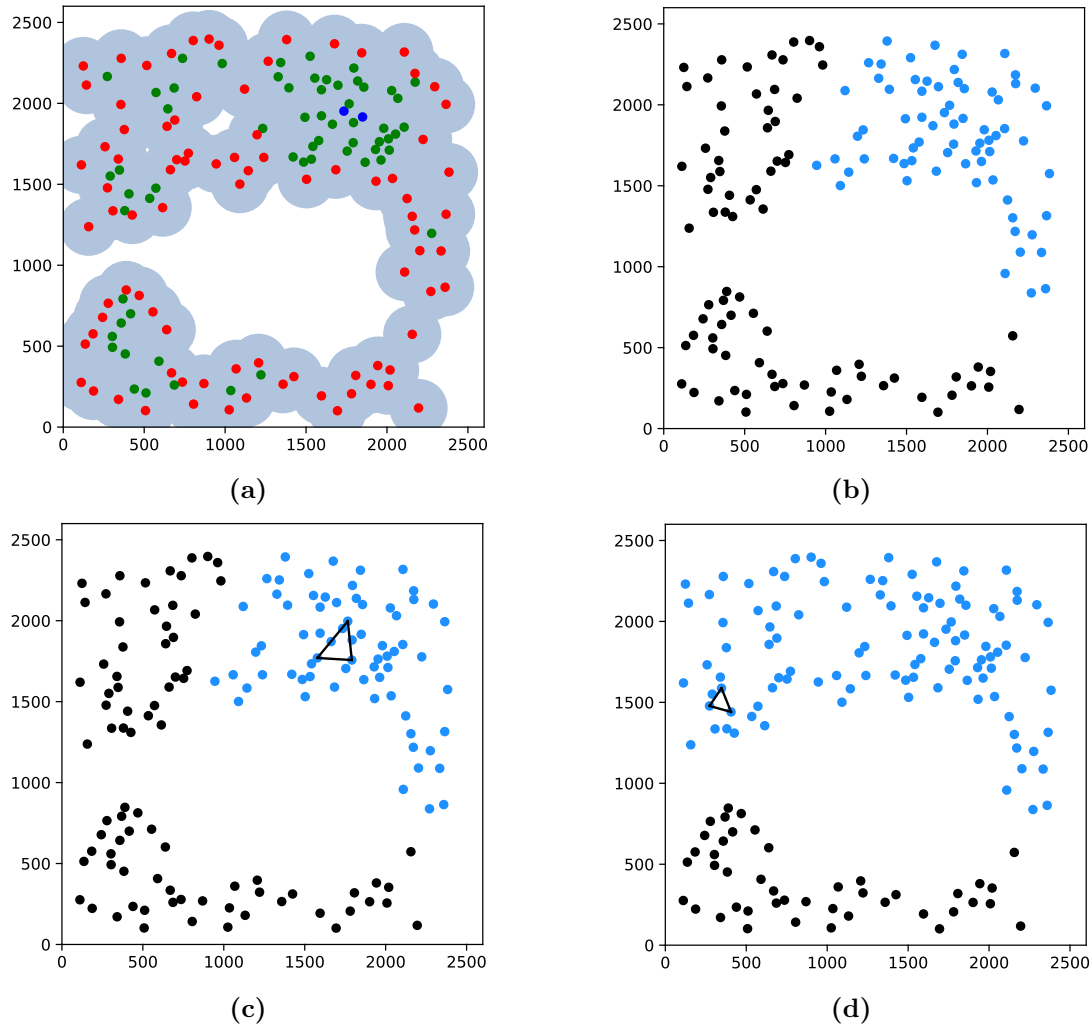
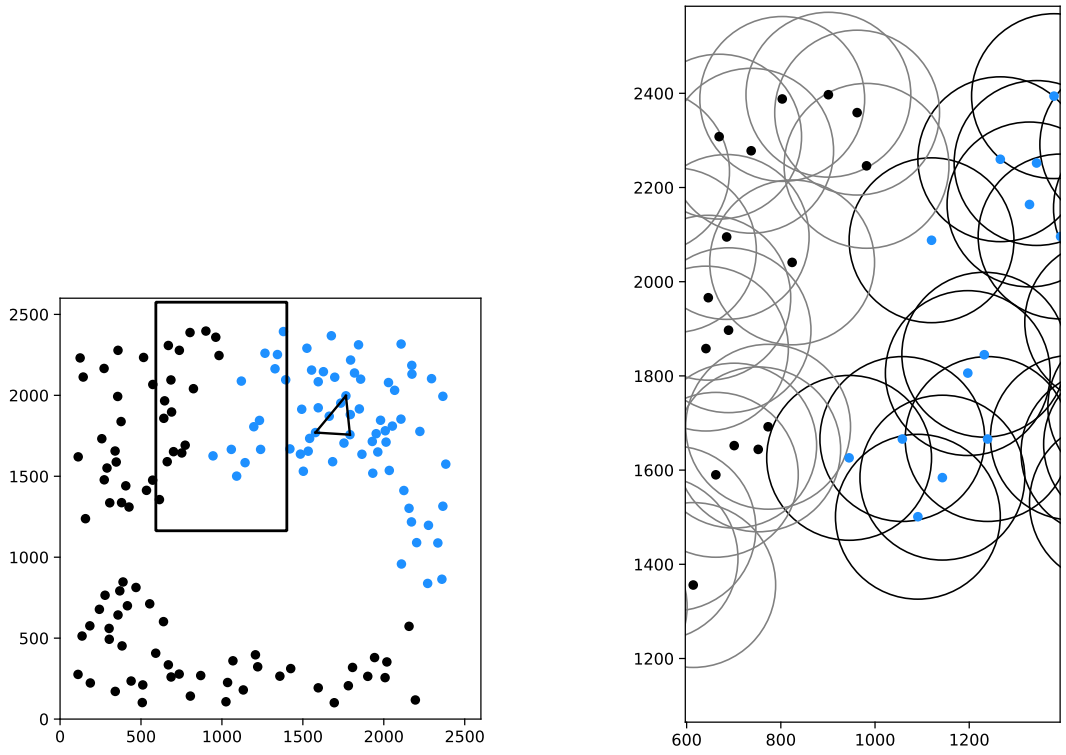


Figure 2.10: a)-b) 69 nodes out of 150 are localized by our algorithm. c)-d) 69 and 106 nodes out of 150 are localized respectively by trilateration from two different choices of the initial triangle.



(a) **(b)**

Figure 2.11: There is no node in the left part that is adjacent to at least three nodes of the right part. Therefore, trilateration does not propagate from right to left. However, there are nodes in the right part that are adjacent to at least three nodes of the left part. Hence, trilateration will propagate from left to right.

characterize the node types and then start trilateration from a triangle consisting of strongly interior nodes. However, it is not difficult to see that the situation shown in Fig. 2.2c can also occur in the strong interior of a network. In fact, we can use this idea to construct a class of networks as shown in Fig. 2.12 in which trilateration does not progress beyond the base step for any choice of the initial triangle, but our algorithm always localizes all the nodes from any initial strongly interior node.

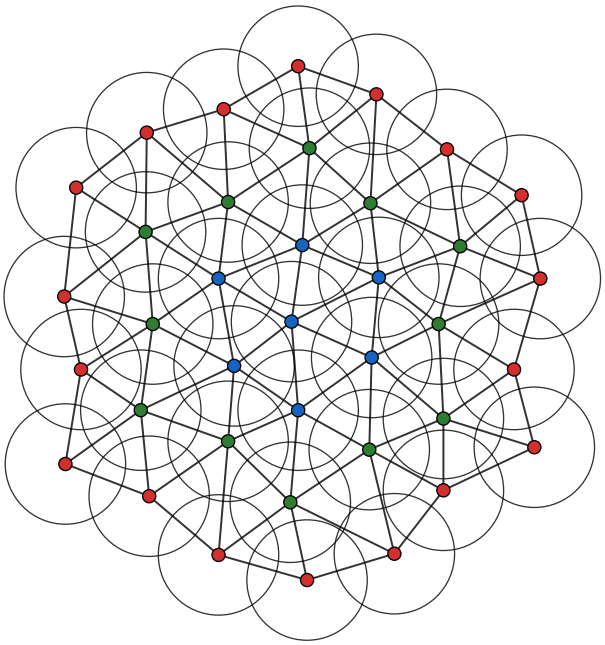


Figure 2.12: The red, green and blue nodes are respectively boundary, weakly interior and strongly interior nodes. It is easy to see that trilateration does not progress beyond the base step for any choice of the initial triangle. However, our algorithm always localizes all the nodes of the network from any initial strongly interior node. This structure can be extended to arbitrarily large sizes. Hence, for any $n \in \mathbb{N}$, we have a network of size $\geq n$, such that 1) it is always entirely localized by our algorithm, 2) but trilateration fails to localize more than 3 nodes for any choice of the initial triangle.

The sensor network model adopted in this work has some idealistic assumptions. We assumed that neighboring nodes can accurately measure the distance between them. Distance measurement techniques such as Received Signal Strength Indication (RSSI), Time of Arrival (ToA), etc do not give accurate results. It would be interesting to study the impact of noisy distance measurement on our algorithm. Our algorithm also works under the strong assumption of uniform communication range. Another important direction of future research would be to see if our approach can be extend to networks with sensors having irregular communication range, e.g., quasi unit disk networks [70].

2.8 Concluding Remarks

Our algorithm works under the condition that the strong interior of the network is connected. Relaxing this condition, it would be interesting to characterize the conditions under which localization starting from different components of the strong interior can be stitched together. It would be also interesting to study impact noisy distance measurement on our algorithm. Our algorithm also works under the strong assumption of uniform communication range. An important direction of future research would be to see if our approach can be extend to networks with sensors having irregular communication range, e.g., quasi unit disk networks. Another problem is to compare the class of networks that are fully localized by our algorithm to those that are fully localized by trilateration.

Chapter 3

Gathering in Hypercubes by Asynchronous Oblivious Robots

The *gathering* problem requires a set of n mobile computational entities, usually called *robots* or *agents*, initially situated at different locations in a spatial universe, to gather at some unspecified location within finite time. When only two robots are involved, the problem is usually referred to as the *rendezvous* problem. In distributed computing, gathering has been extensively studied both in continuous and in discrete domains. In the continuous setting, the robots operate in the two-dimensional Euclidean space and in the discrete case, they operate in a network modeled as a graph. In the discrete setting, the problem has been previously studied in different graph topologies, e.g. rings [32, 54, 64, 65], grids [30, 101], trees [30] etc. The problem is particularly difficult in graphs that are highly symmetric and is solvable only in very limited cases. Hence, for characterization of gatherability, it is important to investigate the problem in highly symmetric graphs. In this chapter[1], we investigate the problem in a hypercube graph.

[1]Based on this thesis, the following paper has been published:
Kaustav Bose, Manash Kumar Kundu, Ranendu Adhikary and Buddhadeb Sau. **Optimal Gathering by Asynchronous Oblivious Robots in Hypercubes**. Algorithms for Sensor Systems. ALGOSENSORS 2018. Lecture Notes in Computer Science(), vol 11410. Springer, Cham. https://doi.org/10.1007/978-3-030-14094-6_7

3.1 The Model

A set of autonomous mobile robots is randomly deployed on the vertices of a d-dimensional hypercube network. The *d-dimensional hypercube* Q_d is an undirected graph with vertex set $V(Q_d) = \mathbb{Z}_2^d = \{0,1\}^d$, and two vertices are adjacent if and only if the two binary strings differ in exactly one coordinate. An *oriented hypercube* is an edge-labeled hypercube with the so-called *dimensional labeling* $\lambda : E(Q_d) \to \{1, \ldots, d\}$ where $\lambda(uv) = i$, if u and v differ in the ith coordinate. We shall denote an oriented hypercube by $Q_d^{\mathcal{O}}$, and an unoriented hypercube by simply Q_d. The binary string labels of the vertices are for descriptive purposes, and are not known to the robots. However, in an oriented hypercube, the edge-labels are known to the robots. It is traditionally assumed that the robots can only perceive the labels of the edges adjacent to the vertex on which it resides. But since the labels of edges adjacent to a single vertex determine the dimensional labels of all the edges in a hypercube (See Theorem 3.1 in [104]), we assume without loss of generality that the robots know the labels of all the edges.

The robots are *oblivious* (they have no memory of past configurations and previous actions), *autonomous* (there is no central control), *homogeneous* (they execute the same distributed algorithm), *anonymous* (they have no unique identifiers) and *identical* (they are indistinguishable by their appearance). The robots have *global visibility*, i.e., they are able to perceive the entire graph. The robots do not agree on any global coordinate system. Furthermore, there are no means of communication between the robots.

The robots, when active, operate according to the so-called Look-Compute-Move cycle. In each cycle, a previously idle or inactive robot wakes up and executes the following steps. In the Look phase, the robot takes the snapshot of the positions of all the robots, represented in its own coordinate system. Based on the perceived configuration, the robot performs computations according to a deterministic algorithm to decide whether to stay idle or to move to an adjacent vertex. Based on the outcome of the algorithm, the robot either remains stationary or makes an instantaneous move to an adjacent vertex. Since the moves are

instantaneous, it implies that the robots are always seen on vertices, not on edges. In the fully synchronous setting (FSYNC), the activation phase of all robots can be logically divided into global rounds, where all the robots are activated in each round. The semi-synchronous (SSYNC) model coincides with the FSYNC model with the only difference that not all robots are necessarily activated in each round. The most general type of scheduler is the asynchronous scheduler (ASYNC). In ASYNC, the robots are activated independently, and the amount of time spent in Look, Compute, Move and inactive states are finite but unbounded and unpredictable. As a result, the robots do not have a common notion of time. In this chapter, we considered the asynchronous scheduler (ASYNC).

An important capability associated to the robots is multiplicity detection. During the Look phase, a robot may perceive a vertex occupied by more than one robot in different ways. In *strong multiplicity detection*, the robots perceive the actual number of robots in each vertex. In *weak multiplicity detection*, the robots are only able to detect whether a vertex is occupied by more than one robot, but not the exact number. If the robots have no multiplicity detection capability, they can only decide if a vertex is occupied or empty. In our model, robots have weak multiplicity detection capability.

3.2 Theoretical Preliminaries

In this section, we present some definitions, notations and explain some important results. Automorphisms, feasibility of gathering, weber points and leading weber points are described in the subsections 3.2.1-3.2.4.

3.2.1 Group of Automorphisms

An *automorphism* of a graph $G = (V, E)$ is a bijection $\varphi : V \longrightarrow V$ such that for all $u, v \in V$, u, v are adjacent if and only if $\varphi(u), \varphi(v)$ are adjacent. The set of all automorphisms of G forms a group, called the *automorphism group* of G and is denoted by $Aut(G)$.

The automorphism group of a hypercube is generated by two types of automorphisms, namely *translation* and *rotation*.

Translation: For $a \in \mathbb{Z}_2^d$, the map $\tau_a : V(Q_d) \longrightarrow V(Q_d)$ given by $u \mapsto u \oplus a$ is called *translation by* a. Here, $u \oplus a$ is the vertex obtained by adding the binary strings u and a componentwise. The set $T = \{\tau_a \mid a \in \mathbb{Z}_2^d\}$ of all translations forms a subgroup of $Aut(Q_d)$.

Rotation: For $\sigma \in S_d$, the map $r_\sigma : V(Q_d) \longrightarrow V(Q_d)$ given by $u \mapsto \sigma(u)$ is called *rotation by* σ, where $\sigma(u)$ is the vertex obtained by permuting the binary string u by $\sigma : \{1, \ldots, d\} \longrightarrow \{1, \ldots, d\}$. The set $R = \{r_\sigma \mid \sigma \in S_d\}$ of all rotations forms a subgroup of $Aut(Q_d)$.

Theorem 3.1. *[51]* $Aut(Q_d) = TR$.

Hence, for any automorphism $\varphi \in Aut(Q_d)$, $\exists$ a unique pair $(a, \sigma) \in (\mathbb{Z}_2^d, S_d)$, such that $\varphi : V \longrightarrow V$ can be written as $u \mapsto \sigma(u) \oplus a$.

It is easy to see that $|T| = 2^d$ and $|R| = d!$. Since $T \cap R$ is trivial, $|TR| = |T||R|/|T \cap R| = 2^d d!$. Therefore, we have the following corollary.

Corollary 3.1.1. *[51]* $|Aut(Q_d)| = 2^d d!$.

The definition of automorphism of graphs can be extended to edge-labeled graphs in a natural way. Given an edge-labeled graph $G = (V, E, \lambda)$ with edge-labeling $\lambda : E \longrightarrow \mathbb{N}$, an automorphism of G is a bijection $\varphi : V \longrightarrow V$ such that for all $u, v \in V$, $\varphi(u)\varphi(v) \in E$ if and only if *1)* $uv \in E$ and *2)* $\lambda(\varphi(u)\varphi(v)) = \lambda(uv)$. In view of this definition, it is easy to see that the dimensional labeling of a hypercube kills all rotational automorphisms. Thus the automorphism group of an oriented hypercube consists of only translations.

Theorem 3.2. $Aut(Q_d^{\mathcal{O}}) = T$.

3.2.2 Feasibility of Gathering

Consider a set of robots placed on the vertices of a simple undirected connected graph $G = (V, E)$. Define a function $f : V \longrightarrow \mathbb{N} \cup \{0\}$, where $f(v)$ is the number

of robots on the vertex v. The pair (G, f) is called a *configuration of robots on* G, or simply a *configuration*. If all the robots in a configuration reside on a single vertex, then it is called *final configuration*; otherwise it is called a *non-final configuration*. Given a configuration (G, f), we define the *multiplicity function* $\tilde{f}$ in the following way. If the model assumes robots with strong multiplicity detection capability, then $\tilde{f}(v) = f(v)$ for all $v \in V$. If the robots have weak multiplicity detection capability, then $\tilde{f} : V \longrightarrow \{0, 1, 2\}$ is defined as,

$$\tilde{f}(v) = \begin{cases} 0 & \text{if } v \text{ is an empty vertex} \\ 1 & \text{if } v \text{ is occupied by exactly one robot} \\ 2 & \text{if } v \text{ is a multiplicity.} \end{cases}$$

If the robots have no multiplicity detection capability, then $\tilde{f} : V \longrightarrow \{0, 1\}$ is defined as,

$$\tilde{f}(v) = \begin{cases} 0 & \text{if } v \text{ is an empty vertex} \\ 1 & \text{if } v \text{ is occupied by at least one robot.} \end{cases}$$

Given a configuration (G, f), the pair $(G, \tilde{f})$ is called the *perceived configuration*.

An *automorphism of a perceived configuration* $(G, \tilde{f})$ is a graph automorphism $\varphi \in Aut(G)$ such that $\tilde{f}(v) = \tilde{f}(\varphi(v))$ for all $v \in V$. The set of all automorphisms of $(G, \tilde{f})$ also forms a group that will be denoted by $Aut(G, \tilde{f})$. If $|Aut(G, \tilde{f})| = 1$, we say that $(G, \tilde{f})$ is *asymmetric*, otherwise it is said to be *symmetric*.

For an automorphism $\varphi \in Aut(G, \tilde{f})$, let $< \varphi > \subseteq Aut(G, \tilde{f})$ be the cyclic subgroup generated by φ. Elements of this group are $\{\varphi^0, \varphi^1, \varphi^2, \ldots, \varphi^{p-1}\}$, where φ^0 is the identity, $\varphi^k = \underbrace{\varphi \circ \varphi \circ \cdots \circ \varphi}_{k \text{ times}}$ and p is the order of φ.

For any subgroup H of $Aut(G, \tilde{f})$, define the equivalence relation on V given by: $x \sim y$ if and only if $x = \varphi(y)$ for some $\varphi \in H$. This equivalence relation induces a partition on V. The *orbit of a vertex* $v \in V$ *under the action of* H is the set $H_v = \{\sigma(v) | \sigma \in H\}$, which is the corresponding equivalence class containing v.

Partitive automorphism: Let $\mathcal{C} = ((V, E), \tilde{f})$ be a perceived configuration. A non-trivial automorphism $\varphi \in Aut(\mathcal{C})$ is said to be *partitive* on V if $|H_v| = p$ for all $v \in V$, where $p > 1$ is the order of φ and $H = < \varphi >$.

Lemma 3.1. *In Q_d, any non-trivial translation is partitive.*

Theorem 2 in [102], stated for configurations of robots with strong multiplicity detection capability can be easily generalized to the following theorem.

Theorem 3.3. *Let $\mathcal{C} = ((V, E), \tilde{f})$ be a non-final perceived configuration. If there exists a $\varphi \in Aut(\mathcal{C})$ partitive on V, then $\mathcal{C}$ is not gatherable.*

Theorem 3.4. *Without multiplicity detection capability, gathering in (both oriented and unoriented) hypercubes is not deterministically solvable in SSYNC.*

Proof. Assume that there exists a correct gathering algorithm $\mathcal{A}$. In the SSYNC model, time can be logically divided into discrete global rounds. So, starting from some non-final initial configuration, consider a synchronous execution of algorithm $\mathcal{A}$, in which gathering is achieved in round t.

Case 1: Suppose that in round $t-1$, exactly two vertices in Q_d are occupied. Hence, the perceived configuration in round $t-1$ is $(Q_d, \tilde{f})$ where $\tilde{f}(v) = \tilde{f}(w) = 1$ for two distinct vertices $v, w \in V(Q_d)$, and $\tilde{f}(u) = 0\ \forall u \in V(Q_d) \setminus \{v, w\}$. But, then the perceived configuration $(Q_d, \tilde{f})$ admits a partitive automorphism given by $x \mapsto x \oplus v \oplus w$. Hence by Theorem 3.3, gathering can not be deterministically achieved from this configuration.

Case 2: Assume that at least three vertices in Q_d are occupied in round $t-1$. Then algorithm $\mathcal{A}$ brings all the robots to a common vertex, say u, in one step. But the adversary can choose to activate all the robots except one that is not placed at u. Then all but one robot will reach u. This will create a configuration with exactly two vertices occupied. Since this configuration admits a partitive automorphism, gathering can not be deterministically achieved from here by Theorem 3.3.

□

Corollary 3.4.1. *Without multiplicity detection capability, gathering in (both oriented and unoriented) hypercubes is not deterministically solvable in ASYNC.*

3.2.3 Weber Point

Given a configuration (G, f), with $G = (V, E)$, the centrality of $v \in V$ is defined as $c_{G,f}(v) = \sum_{u \in V} d(u, v) \cdot f(u)$. When there is no ambiguity, we shall write $c_f(v)$, or simply $c(v)$.

Weber point: Given a configuration $\mathcal{C} = (G = (V, E), f)$, a vertex $v \in V$ is a *Weber point* of $\mathcal{C}$, if $c(v) = min\{c(u)|u \in V\}$.

By definition, a Weber point is a vertex with minimum centrality. In other words, a vertex $w \in V$ is a Weber point if the sum of the lengths of the shortest paths from all robots to w is minimum. Therefore, an algorithm that gathers all the robots at a Weber point via the shortest paths is optimal with respect to the total number of moves performed by the robots. However, a configuration may have more than one Weber point. Given a configuration (G, f), we shall denote the set of Weber points by $\mathcal{W}_{G,f}$, or simply $\mathcal{W}_f$ or $\mathcal{W}$ when there is no ambiguity.

Theorem 3.5. *[102] Let (G, f) be a configuration with Weber points $\mathcal{W}_f$. If a robot moves towards a Weber point $w \in \mathcal{W}_f$, resulting in a new configuration (G, f'), then*

1. $c_{f'}(v) = c_f(v) - 1$ *for each* $v \in \mathcal{W}_{f'}$
2. $w \in \mathcal{W}_{f'}$
3. $\mathcal{W}_{f'} \subseteq \mathcal{W}_f$.

We shall now discuss about the Weber points of configurations on a hypercube. Consider a set of n robots $\{r_1, r_2, \ldots, r_n\}$ on a d-dimensional hypercube Q_d. Suppose that the robots $r_1, r_2, \ldots, r_n$ are placed on the vertices $v_1, v_2, \ldots, v_n$ respectively. For $i = 1, 2, \ldots, n$, let the binary string representation of v_i be $b_{i1}b_{i2}\ldots b_{id}$, where $b_{ij} \in \{0, 1\}$. For $j = 1, 2, \ldots, d$, let us define sets $[b]_j \subseteq \{0, 1\}$ in the following way.

$$[b]_j = \begin{cases} \{0\}, & \text{if the number of 0's in the multiset } \{b_{1j}, b_{2j}, \ldots, b_{nj}\} \text{ is more than the number of 1's} \\ \{1\}, & \text{if the number of 1's in the multiset } \{b_{1j}, b_{2j}, \ldots, b_{nj}\} \text{ is more than the number of 0's} \\ \{0,1\}, & \text{if the multiset } \{b_{1j}, b_{2j}, \ldots, b_{nj}\} \text{ has equal number of 0's and 1's.} \end{cases}$$

In Theorem 3.6, we show that the set of Weber points of the configuration is

$$\mathcal{W} = [b]_1 \times [b]_2 \times \ldots \times [b]_d.$$

For instance, consider a configuration of a set of 8 robots $\{r_1, r_2, \ldots, r_8\}$ on a 4-dimensional hypercube Q_4. Suppose that the robots are positioned on the following vertices respectively: 0110, 0111, 1000, 1110, 0000, 0001, 1110, 1101. Then the set of Weber points of this configuration is given by

$$\begin{aligned} \mathcal{W} &= \{0,1\} \times \{1\} \times \{0,1\} \times \{0\} \\ &= \{0100, 0110, 1100, 1110\}. \end{aligned}$$

Theorem 3.6. *Let $\{r_i\}_{i=1}^n$ be a set of robots placed on the vertices of Q_d with binary string representations $\{b_{i1}b_{i2}\ldots b_{id}\}_{i=1}^n$ respectively. Then the set of Weber points of the configuration is $\mathcal{W} = [b]_1 \times [b]_2 \times \ldots \times [b]_d$.*

Proof. The distance between any two vertices in Q_d is the number of positions in which their binary string representations differ. Then it can be easily seen that 1) the centrality of all $w \in [b]_1 \times [b]_2 \times \ldots \times [b]_d$ are equal, 2) the centrality of any $v \in V(Q_d) \setminus [b]_1 \times [b]_2 \times \ldots \times [b]_d$ is strictly greater than the centrality of $w \in [b]_1 \times [b]_2 \times \ldots \times [b]_d$. □

Corollary 3.6.1. *The subgraph induced by the set of Weber points $\mathcal{W}$ of a configuration on a hypercube Q_d is also a hypercube.*

Corollary 3.6.2. *The number of Weber points of a configuration on a hypercube Q_d is 2^k, where $0 \leq k \leq d$.*

3.2.4 Leading Weber Point

A configuration of robots on a hypercube can have more than one Weber point. We want to devise an algorithm that gathers all the robots at one of the Weber points via the shortest paths. Our proposed algorithm requires to solve a subproblem called LeadingWeberPoint. Let us formally define the problem LeadingWeberPoint. Consider a configuration in which no vertex contains more than one robot, and that has no partitive automorphism. Let $\mathcal{W}$ be the set of Weber points of this configuration. The problem LeadingWeberPoint asks to devise an algorithm so that every robot that perceives this configuration in its local view, deterministically elects a unique Weber point $w_\ell \in \mathcal{W}$. We shall call the vertex w_ℓ the *leading Weber point*.

Since we have assumed that the robots are positioned at distinct vertices, there is no distinction between the configuration and the perceived configuration. In other words, given such a configuration (G, f), we have $\tilde{f} = f$. A vertex $v \in V$ is called a *fixed vertex* if $\varphi(v) = v$, $\forall \varphi \in Aut(G, f)$.

Theorem 3.7. LeadingWeberPoint *can be deterministically solved only if* $\mathcal{W}$ *has at least one fixed vertex.*

Proof. Consider a configuration (Q_d, f) that has no partitive automorphism. Since we have assumed that the robots are positioned at distinct vertices, there is no distinction between the configuration and the perceived configuration. In other words, we have $\tilde{f} = f$. Assume that the configuration has no fixed Weber point. Let us assume that there is an algorithm $\mathcal{A}$ that deterministically solves LeadingWeberPoint. Let $w_1 \in \mathcal{W}$ be the leading Weber point elected by the robots. Since w_1 is not a fixed vertex, there is a $w_2 \neq w_1$ such that $\varphi(w_1) = w_2$, for some $\varphi \in Aut(G, f)$.

Each robot observes the positions of other robots in its local coordinate system. A local coordinate system of a robot is just an assignment $\Psi : V(Q_d) \longrightarrow \{0, 1\}^d$, respecting the rule that $u, v \in V(Q_d)$ are adjacent if and only if $\Psi(u), \Psi(v)$ differ in precisely one bit. Since there is no global agreement, the local coordinate system of each robot is arbitrary, and is chosen by the adversary. Let us formally define

the view of a robot. The view of a robot is given by the triplet $\mathcal{V}_\Psi = (\Psi, \tilde{f}, me)$, where $\Psi : V(Q_d) \longrightarrow \{0,1\}^d$ is the local coordinate system, $\tilde{f} : \{0,1\}^d \longrightarrow \{0,1\}$ is the multiplicity function defined on the set of vertices expressed in local coordinates, and $me \in \{0,1\}^d$ is the coordinates of the vertex on which the robot resides. The view $\mathcal{V}_\Psi$ is the input for algorithm $\mathcal{A}$. The output $\mathcal{A}(\mathcal{V}_\Psi) \in \{0,1\}^d$ is the coordinates of the required leading Weber point, i.e., the returned leading Weber point is the vertex $\Psi^{-1}(\mathcal{A}(\mathcal{V}_\Psi))$.

Consider a robot r_1 in the configuration residing at vertex v_1. The robot r_1, using a local coordinate system $\Psi_1 : V(Q_d) \longrightarrow \{0,1\}^d$, elects w_1 as the leading Weber point. That is, given the input in the coordinate system Ψ_1, the output of $\mathcal{A}$ is $\Psi_1(w_1)$. Now consider the following cases.

Case 1: Suppose that $\varphi(v_1) = v_1$. Consider the local coordinate system $\Psi_2 = \Psi_1 \circ \varphi^{-1}$. Note that the view of r_1 in coordinate systems Ψ_1 and Ψ_2 are exactly the same, i.e., $\mathcal{V}_{\Psi_1} = \mathcal{V}_{\Psi_2}$. Since $\mathcal{A}$ is a deterministic algorithm, $\mathcal{A}(\mathcal{V}_{\Psi_1}) = \mathcal{A}(\mathcal{V}_{\Psi_2})$. Since the elected leading Weber point in local coordinate system Ψ_1 is w_1, we have $\mathcal{A}(\mathcal{V}_{\Psi_1}) = \Psi_1(w_1)$. So we have,

$$\begin{aligned} & \mathcal{A}(\mathcal{V}_{\Psi_2}) = \mathcal{A}(\mathcal{V}_{\Psi_1}) = \Psi_1(w_1) \\ \Rightarrow \quad & \Psi_2^{-1}(\mathcal{A}(\mathcal{V}_{\Psi_2})) = \Psi_2^{-1}(\Psi_1(w_1)) = \varphi \circ \Psi_1^{-1} \circ \Psi_1(w_1) = \varphi(w_1) = w_2 \end{aligned}$$

Hence, we see that in local coordinate system Ψ_1 the robot r_1 elects w_1 as the leading Weber point, while in Ψ_2 it elects w_2. This is a contradiction.

Case 2: Now assume that $\varphi(v_1) = v_2 \neq v_1$. Then there must be a robot r_2 in v_2. Suppose that the adversary sets the local coordinate system of r_2 as $\Psi_2 = \Psi_1 \circ \varphi^{-1}$. Then the view of r_1 and r_2 will be identical, i.e., $\mathcal{V}_{\Psi_1} = \mathcal{V}_{\Psi_2}$. Again we have, $\mathcal{A}(\mathcal{V}_{\Psi_2}) = \mathcal{A}(\mathcal{V}_{\Psi_1}) = \Psi_1(w_1)$ and hence, $\Psi_2^{-1}(\mathcal{A}(\mathcal{V}_{\Psi_2})) = w_2$. Therefore, r_2 will elect w_2, while r_1 elects w_1 as the leading Weber point. This is again a contradiction. □

Theorem 3.8. LeadingWeberPoint *may not be deterministically solvable in an unoriented hypercube.*

Proof. Consider a configuration (Q_5, f) of a set of 14 robots on the 5-dimensional

unoriented hypercube Q_5. The robots are placed on the following vertices: 00100, 00001, 11000, 10010, 01100, 01010, 00101, 00011, 11010, 10110, 11001, 10101, 01111, 11111. It is easy to see that $\mathcal{W}_f = V(Q_5)$. It can be shown that $Aut(Q_5, f) = \{e, \varphi_1, \varphi_2, \varphi_3\}$, with each φ_i given by $u \mapsto \sigma_i(u) \oplus a_i$, where $a_i \in \mathbb{Z}_2^5, \sigma_i \in S_5$ are the following: $a_1 = 00000$, $\sigma_1 = (1)(24)(35)$, $a_2 = 10000$, $\sigma_2 = (1)(2543)$, $a_3 = 10000$, $\sigma_3 = (1)(2345)$. Then it can be easily verified that 1) there is no partitive automorphism in $Aut(Q_5, f)$, 2) there is no fixed vertex in $\mathcal{W}_f = V(Q_5)$. So by Theorem 3.7, LeadingWeberPoint is deterministically unsolvable. □

Now we show that LeadingWeberPoint can be deterministically solved in an oriented hypercube.

Lemma 3.2. *Given a vertex u_0 in an oriented hypercube $Q_d^{\mathcal{O}}$, $\exists$ exactly one coordinate assignment (bijection) $\Psi : V(Q_d^{\mathcal{O}}) \longrightarrow \{0,1\}^d$ such that*

1. *$\Psi(u_0) = 00\ldots0 \in \{0,1\}^d$*
2. *u, v are adjacent if and only if $\Psi(u), \Psi(v)$ differ in exactly one bit*
3. *for $uv \in E(Q_d^{\mathcal{O}})$, $\lambda(uv) = i$ if and only if $\Psi(u), \Psi(v)$ differ in the ith position.*

Proof. The coordinates given to u_0 are $00\ldots0$. Then by rule 2) and 3), the coordinates of all its neighbors are uniquely determined. If the coordinates of all vertices at distance $i(< d)$ from u_0 are uniquely determined, then again by rule 2) and 3), the coordinates of all vertices at distance $i+1$ can be determined uniquely. Hence by induction, the coordinates assigned to all the vertices are unique. □

Now for any $w \in V(Q_d^{\mathcal{O}})$, we define a binary string $\zeta(w)$ of length 2^d in the following the way:

1. First, give $Q_d^{\mathcal{O}}$ the unique coordinate assignment $\Psi : V(Q_d^{\mathcal{O}}) \longrightarrow \{0,1\}^d$ with $\Psi(w) = 00\ldots0$.

2. Now we define a total ordering $\prec$ on $V(Q_d^{\mathcal{O}})$ as: $u \prec v$

$$u \prec v \Leftrightarrow \begin{cases} d(u,w) < d(v,w) \\ Or, \\ d(u,w) = d(v,w), \text{and } \Psi(u) \text{ is lexicographically larger that } \Psi(v), \end{cases}$$

where $d(u,w)$ is the distance of u from w. For example, when $d = 4$, the assigned coordinates of the vertices written in increasing order will be: $\underbrace{0000}_{\text{distance 0}}$, $\underbrace{1000, 0100, 0010, 0001}_{\text{distance 1}}$, $\underbrace{1100, 1010, 1001, 0110, 0101, 0011}_{\text{distance 2}}$, $\underbrace{1110, 1101, 1011, 0111}_{\text{distance 3}}$, $\underbrace{1111}_{\text{distance 4}}$.

3. Finally, scan the vertices of the hypercube according to the above ordering. For each vertex, put a 0 if it is empty, or 1 if it is occupied by a robot. Recall that any vertex can be occupied by at most one robot. The string of length 2^d thus obtained is $\zeta(w)$. In the previous example, if the occupied vertices are $0000, 1000, 0010, 1001, 0011, 1011, 1111$, then $\zeta(w) = 1101000100100101$.

Lemma 3.3. *For any two distinct vertices $u, v \in V(Q_d^{\mathcal{O}})$, if $\zeta(u) = \zeta(v)$, then the configuration has a partitive automorphism.*

Proof. It can be easily seen that if $\zeta(u) = \zeta(v)$, then the configuration has the automorphism (translation) given by $x \mapsto x \oplus u \oplus v$. □

Theorem 3.9. LeadingWeberPoint *is solvable in an oriented hypercube.*

Proof. Since the configuration has no partitive automorphism, $\zeta(w_1) \neq \zeta(w_2)$ for any distinct $w_1, w_2 \in \mathcal{W}$. Hence the robots can unanimously elect $w \in \mathcal{W}$ with the lexicographically (strictly) largest $\zeta(w)$ as the leading Weber point. □

3.3 The Algorithm

Our plan is to solve the problem in two stages. In stage 1, we create a multiplicity at a Weber point and then in stage 2, we sequentially bring the remaining robots to that vertex. Before describing the algorithm, we first give two definitions.

Anchor: Let $(Q_d^{\mathcal{O}}, \tilde{f})$ be a non-final perceived configuration on an oriented hypercube with at most one multiplicity and no partitive automorphism. The *anchor* of $(Q_d^{\mathcal{O}}, \tilde{f})$ is a vertex $\alpha \in V(Q_d^{\mathcal{O}})$ defined as the following. If $(Q_d^{\mathcal{O}}, \tilde{f})$ has no multiplicity, then α is the leading Weber point; otherwise α is the unique vertex with multiplicity. Note that all the robots observing the configuration $(Q_d^{\mathcal{O}}, \tilde{f})$ agree on which vertex is the anchor.

Leader: Since all the robots observing the configuration $(Q_d^{\mathcal{O}}, \tilde{f})$ agree on the anchor α, they also agree on a common coordinate system, which is the unique coordinate system Ψ described in Lemma 3.2 with $\Psi(\alpha) = 00\ldots0$. This also allows the robots to order the vertices of the hypercube as described in the previous section. In this ordering, the first robot appearing on a non-anchor vertex will be called the *leader*.

Depending on number of robots of the configuration, algorithm is described in the following subsections 3.3.1 and 3.3.2.

3.3.1 2k +1 Robots

Theorem 3.10. *Any configuration on a hypercube with odd number of robots has exactly one Weber point.*

Proof. Using the same notations as in Theorem 3.6, the set of Weber points is given by $\mathcal{W} = [b]_1 \times [b]_2 \times \ldots \times [b]_d$. Since there are odd number of robots, the multiset $\{b_{1j}, b_{2j}, \ldots, b_{nj}\}$ can never have equal number of 0's and 1's. Hence, $[b]_j = \{0\}$ or $\{1\}$, $\forall j \in \{1, \ldots, d\}$. Thus $|\mathcal{W}| = 1$.

□

Theorem 3.11. *Gathering in $Q_d^{\mathcal{O}}$ is optimally solvable in ASYNC with weak multiplicity detection for any configuration of odd number of robots.*

Proof. We simply ask only the leader to move towards the anchor. The anchor α is the unique Weber point of the configuration. As the leader moves towards it, the Weber point remains invariant by Theorem 3.5. After one or two robots

reach α, a multiplicity is created at α. Throughout stage 2, α remains the unique multiplicity in the configuration, since only the leader moves. Thus, all the remaining robots will sequentially reach α. The algorithm is clearly optimal with respect to the total number of moves executed by the robots.

□

3.3.2 4k Robots

In view of Theorem 3.2, Theorem 3.3 and Lemma 3.1, any configuration with non-trivial automorphism group is ungatherable. We show that for all the remaining configurations, gathering can be optimally solved. Again our strategy is to move the leader towards the anchor. However, in this case the leader has to judiciously choose the edge via which it should approach the anchor. Unlike the previous case, the anchor may change after a move.

Consider the first stage of the algorithm, when there is no multiplicity in the configuration. Then the anchor is the leading Weber point w_ℓ. We classify all the non-anchor vertices into two types: *type 1* and *type 2*. If the configuration has 2^m $(0 \leq m \leq d)$ Weber points, then among the d neighbors of w_ℓ, m are also Weber points. This is because of Lemma 3.6.1. Let us call these Weber points $w_1, \ldots, w_m$. Since the coordinates assigned to w_ℓ are $0 \ldots 0$, the coordinates of each $w_i \in \{w_1, \ldots, w_m\}$ have exactly one 1. For each w_i, assume that its assigned coordinates have the 1 at the p_ith place, which implies that the edge joining w_i and w_ℓ has label p_i. Also, the set of Weber points, in the assigned coordinates, is given by $\mathcal{W} = [b]_1 \times [b]_2 \times \ldots \times [b]_d$, where $[b]_l$ is $\{0, 1\}$ if $l \in \{p_1, \ldots, p_m\}$, and $\{0\}$ otherwise.

Type 1 vertex: A non-anchor vertex v will be called a *type 1 vertex* if the following holds: there is at least one $p_i \in \{p_1 \ldots p_m\}$ such that the assigned coordinates of v have 1 at p_ith place. Also the edge incident to v with label p_i will be called a *type 1 edge*.

Type 2 vertex: If a non-anchor vertex v is not type 1, then it will be called a *type 2 vertex*. This implies that the p_1th, $\ldots p_m$th terms of the assigned coordinates

Algorithm 3: Gathering for $4k$ ($k > 0$) robots

```
1  Procedure GATHER()
2      α ← anchor
3      if α is a multiplicity then
4          if I am leader then
5              Move towards α
6      else
7          if I am leader then
8              if I am on a Type 1 vertex then
9                  Move towards α via a Type 1 edge
10             else if I am on a Type 2 vertex then
11                 Move towards α
```

of v are 0. Note that if the configuration has only one Weber point, i.e., $m = 0$, then all non-anchor vertices are vacuously type 2.

Theorem 3.12. *Let $(Q_d^{\mathcal{O}}, f)$ be a configuration of $4k$ ($k > 0$) robots with no multiplicities and no partitive automorphisms. Let w_ℓ be the leading Weber point, and hence the anchor. Assume that the leader r is placed at a type 1 vertex u. Suppose that it moves via a type 1 edge with label p_i to an empty vertex v, and gives rise to configuration $(Q_d^{\mathcal{O}}, f')$. Then the following holds.*

1. *$w_\ell \in \mathcal{W}_{f'}$*
2. *If $|\mathcal{W}_f| = 2^m (m > 0)$, then $|\mathcal{W}_{f'}| = 2^{m-1}$*
3. *$(Q_d^{\mathcal{O}}, f')$ has no partitive automorphism.*

Proof. 1) Assume that r moves via a type 1 edge with label p_i. Then the assigned coordinates of u and v differ in exactly one bit at the p_ith position. At the p_ith place, u has 1, while v has 0. Then v has less 1's than u, and hence v is closer to w_ℓ than u, i.e., r has moved towards w_ℓ. Hence, by Lemma 3.5, $w_\ell \in \mathcal{W}_{f'}$.

2) It is easy to see that, as r moves from u to v, i) its distance from all Weber points whose coordinates have 1 at p_ith place (there are 2^{m-1} of them), increases by one, and ii) its distance from all Weber points whose coordinates have 0 at

p_ith place, decreases by one. Hence the move reduces the set of Weber points by half.

3) If possible, assume that $(Q_d^{\mathcal{O}}, f')$ admits a partitive automorphism, i.e., a non-trivial translation τ. Assume that the translation, in the assigned coordinate system, is given by $x \mapsto x \oplus a$, for some $a \in \{0,1\}^d$. Let $\mathcal{R}$ and $\mathcal{R}'$ be the set of vertices occupied by robots in $(Q_d^{\mathcal{O}}, f)$ and $(Q_d^{\mathcal{O}}, f')$ respectively. Since τ maps any vertex of $\mathcal{R}'$ to another vertex of $\mathcal{R}'$, the group $< \tau >$ induces an equivalence relation on $\mathcal{R}'$, partitioning it into $2k$ disjoint sets of cardinality 2: $\mathcal{R}' = \bigcup_{j=1}^{2k} \{x_j, \tau(x_j)\}$. Let $\mathcal{R}_{p_i}$ and $\mathcal{R}'_{p_i}$ be the multiset containing the p_ith terms of the assigned coordinates of vertices of $\mathcal{R}$ and $\mathcal{R}'$ respectively. Clearly $\mathcal{R}_{p_i}$ contains $2k$ number of 0's and $2k$ number of 1's. In $\mathcal{R}'_{p_i}$, we have $2k+1$ number of 0's and $2k-1$ number of 1's.

Case 1: Let the p_ith term of a be 0. Hence, if p_ith term of x is $b \in \{0,1\}$, then the p_ith term of $\tau(x) = x \oplus a$ is also b. This implies that $\mathcal{R}'_{p_i}$ has even number of 0's and 1's. This is a contradiction, as we have shown that number of 0's and 1's in $\mathcal{R}'_{p_i}$ is $2k+1$ and $2k-1$ respectively.

Case 2: Let the p_ith term of a be 1. So, if p_ith term of x is $b \in \{0,1\}$, then the p_ith term of $\tau(x) = x \oplus a$ is $\overline{b}$. This implies that $\mathcal{R}'_{p_i}$ has equal number of 0's and 1's. This is again a contradiction.

□

Theorem 3.13. *Let $(Q_d^{\mathcal{O}}, f)$ be a configuration of $4k$ $(k > 0)$ robots with no multiplicities and no partitive automorphisms. Let w_ℓ be the leading Weber point, and hence the anchor. Suppose that the leader r is placed at a type 2 vertex u. If it moves towards w_ℓ to an empty vertex, then*

1. *the new configuration $(Q_d^{\mathcal{O}}, f')$ has no partitive automorphism*
2. *$\mathcal{W}_f = \mathcal{W}_{f'}$*
3. *w_ℓ is the leading Weber point of $(Q_d^{\mathcal{O}}, f')$*
4. *r is the leader in $(Q_d^{\mathcal{O}}, f')$.*

Proof. We use the same notations as in the proof of the previous theorem. For each $w \in \mathcal{W}_f \setminus \{w_\ell\}$ the following holds: i) among the p_1th, $\ldots, p_m$th terms of the assigned coordinates, there is at least one 1, ii) all the terms except the p_1th, $\ldots, p_m$th ones are 0. Exactly the opposite is true for the type 2 vertex u. It implies that the distance of r from w_ℓ is strictly less than its distance from any other Weber point in $\mathcal{W}_f \setminus \{w_\ell\}$. Also, after the move, its distances from all Weber points reduce by exactly 1. Hence, after the move, $\zeta(w_\ell)$ remains lexicographically strictly largest among $\{\zeta(w) \mid w \in \mathcal{W}_{f'}\}$. All the statements of the theorem easily follow from these observations. □

Lemma 3.4. *Let $G = (V, E)$ be an arbitrary graph. Let $\mathcal{P} = v_0 e_0 v_1 e_1 v_2 \ldots v_{l-1} e_{l-1} v_l$ be a path from v_0 to v_l. Suppose that for any e_j, a move through it by a robot from v_j to v_{j+1} reduces its distance from v_l by 1. Then $\mathcal{P}$ is a shortest path from v_0 to v_l in G.*

Lemma 3.5. *Suppose that a robot moves from a vertex u to an adjacent vertex v in a hypercube Q_d. Then for any $w \in V(Q_d)$, either its distance from w reduces by 1 or increases by 1, i.e., its distance from w does not remain unchanged.*

Theorem 3.14. *Algorithm 3 achieves optimal gathering in ASYNC, for all asymmetric configurations of $4k$ ($k > 0$) robots with weak multiplicity detection.*

Proof. By Theorem 3.12 and 3.13, no move in the first stage creates a partitive automorphism. Since at any time, only the leader is allowed to move, multiplicity can only be created at the anchor. Since throughout stage 2, there is a unique multiplicity, a configuration with a partitive automorphism is never created. Notice that in both stages, an anchor is always a Weber point. Hence, the robots always move towards some Weber point. So, after each move, the centrality of the surviving set of Weber points is reduced by 1. Therefore, eventually the centrality of one Weber point becomes 0, which implies that gathering is accomplished.

It remains to prove that Algorithm 3 is optimal with respect to total number of moves executed by the robots. Suppose that the algorithm gathers all the robots at w, which was a Weber point of the initial configuration. In view of Lemma 3.12, it is sufficient to show that every movement executed by any robot

is towards w. Since every movement executed by a robot is towards some Weber point, according to Theorem 3.5, the set of Weber points of the configurations starting from the initial to the final configuration form the following nested series: $\mathcal{W}_0 \supseteq \mathcal{W}_1 \supseteq \ldots \supseteq \mathcal{W}_{final} = \{w\}$. In other words, w remains a Weber point throughout the progress of the algorithm. If at some step, a move by a robot is not towards w, then by Lemma 3.13, it moves away from w. Then the centrality of w is increased by 1, while the centrality of some other Weber point is decreases by 1. This means that w does not remain a Weber point after the move. This is a contradiction. □

3.4 Concluding Remarks

This is the first chapter that investigates the gathering problem on a hypercube graph. We have provided a complete characterization of all gatherable configurations in ASYNC for $2k+1$ and $4k$ ($k > 0$) number of robots with weak multiplicity detection in an oriented hypercube. This leaves unsettled only the configurations with $4k + 2$ ($k > 0$) number of robots. Note that our strategy for $4k$ robots does not work for $4k+2$ robots. To see this, consider a configuration in Q_9 of 10 robots placed on the following vertices: 000000000, 110111000, 101111000, 011111000, 000111000, 001000111, 010000111, 100000111, 111000111, 111000000. Here, the anchor, i.e., the leading Weber point is 000000000 and the leader is 111000000. It can be seen that a move by the leader towards the anchor via any edge creates a configuration with a partitive automorphism. Another challenging direction of future research would be to study the problem with limited visibility. It would also be interesting to consider randomized algorithms to bypass the impossibility results.

Chapter 4

Arbitrary Pattern Formation on Infinite Grid by Opaque Point Robots

Arbitrary Pattern Formation ($\mathcal{APF}$) Problem is a classical problem in swarm robotics which deals with fundamental coordination problem of multi-robot systems. The problem is to design an algorithm which will be used by each autonomous mobile robot of a robot swarm that will guide the robots to form any specific pattern that is given to the robots initially as input. In this problem,[1] the robots are modeled as *autonomous* (there is no central control), *anonymous* (the robots do not have any unique identifiers), *homogeneous* (all robots execute the same algorithm) and *identical* (robots are indistinguishable by their appearance). All the robots can freely move on the plane. Each robot has sensing capability by which they can perceive the location of other robots on the plane. The robots do not have any global coordinate system (each robot has its own local coordinate system) and they operate in Look-Compute-Move (LCM) cycles. In the Look phase, a robot takes a snapshot of its surroundings. In the Compute, phase a robot process the information got from the Look phase and

[1]Based on this chapter, the following paper has been published:
Manash Kumar Kundu, Pritam Goswami, Satakshi Ghosh and Buddhadeb Sau. **Arbitrary pattern formation by asynchronous opaque robots on infinite grid**. Arxiv e-prints, May. 2022. arXiv:2205.03053. https://doi.org/10.48550/arXiv.2205.03053

in the MOVE, phase a robot moves to another position (a robot also might stay still in this phase) depending on the output of the COMPUTE phase.

4.1 Problem description and our contribution

This chapter deals with the arbitrary pattern formation problem on an infinite grid using opaque luminous robots with 8 colors. The robots operate in *LCM* cycles under an adversarial asynchronous scheduler. The robots are autonomous, anonymous, identical and homogeneous. They move only through the edges of the grids and the movement is instantaneous for each robot (i.e a robot can only be seen on a grid point). Initially the robots are placed arbitrarily on the grid. From this configuration, they need to move to a target configuration or, Pattern (a set of target coordinates) without collision. The robots have one axis agreement and does not have agreement on global coordinate (each robot has its own local coordinate).

The main difficulty of the problem is visibility. As $\mathcal{APF}$ is closely related to the LEADER ELECTION problem, without seeing the whole configuration it is quite hard to elect a leader and thus design an algorithm to solve the problem. Depending on the local view of each robot, the algorithm is need to be designed. In this chapter, the described algorithm does so. Also another difficulty was to avoid collision between robots while they are moving. We removed this difficulty by using a technique where the robots will move sequentially.

The problem described in this chapter is also very practical in nature. The restricted movement and also the obstructed visibility, these practical scenarios are considered here. The algorithm described in this chapter solves the above mentioned $\mathcal{APF}$ problem in total $\mathcal{O}(kD)$ moves in the worst case, where k is the number of robots on the grid and D is $max\{m, n, M, N, k\}$ (m, n are the height and width of smallest enclosing rectangle of the initial configuration; M, N are the same of for the target configuration).

4.2 Model

Robots: Robots are autonomous, anonymous, homogeneous and identical. They are deployed on a two-dimensional infinite grid where each of them is initially positioned on distinct grid points. They do not have a common notion of direction. The robots have an agreement over the positive direction of X-axis i.e, all the robots have an agreement over left and right. They do not have any agreement over the Y-axis. Here the robots do not have access to any global coordinate system other than the agreement over the positive direction of X-axis. The total number of robots is not known to them. The robots are assumed to be dimensionless and modeled as points.

Look-Compute-Move cycles: The robot, when active, operates according to the Look-Compute-Move cycle. In the Look phase, a robot takes the snapshot of the positions of all the robots represented in its own local co-ordinate system. Then the robot performs computation and compute the next position and a light according to a deterministic algorithm i.e., the Compute phase. In the Move phase, it will either move unit length to the desired location along a straight line or make a null move.

Scheduler: We assume that the robots are controlled by an asynchronous adversarial scheduler. This implies that the amount of time spent in Look, Compute, Move, and inactive states by different robots is finite but unbounded and unpredictable. As a result, the robots do not have a common notion of time, and the configuration perceived by a robot during the Look phase may significantly change before it actually makes a move.

Movement: The movement of robots are restricted only along grid lines from one grid point to one of its four neighboring grid points. Robots' movements are assumed to be instantaneous in discrete domains. Here we assume that the movements are instantaneous. The robots are always seen on grid points, not on edges.

Visibility: The robots visibility is unlimited but by the presence of other robots it can be obstructed. A robot r_i can see another robot r_j if and only if there are

no robots on the straight line segment $\overline{r_i r_j}$.

Lights: Each robot is equipped with an externally visible light, which can assume a $\mathcal{O}(1)$ number of predefined colours. The robots communicate with each other using these colours. The colours of the light are not deleted at the end of a cycle, but otherwise, the robots are oblivious. The colours used in our algorithm are {`off`, `terminal1`, `symmetric`, `decider`, `call`, `leader1`, `leader`, `done`}.

4.3 Notations and Definitions

We have used some notations throughout the chapter. A list of these notations along with their definitions are mentioned in the following table.

$\mathcal{L}_1$	First vertical line on left that contains at least one robot.
$\mathcal{L}_V(r)$	The vertical line on which the robot r is located.
$\mathcal{L}_H(r)$	The horizontal line on which the robot r is located.
$\mathcal{L}_I(r)$	The left immediate vertical line of robot r which has at least one robot on it.
$\mathcal{R}_I(r)$	The right immediate vertical line of robot r which has at least one robot on it.
$H_L^O(r)$	Left open half for the robot r.
$H_L^C(r)$	Left closed half for the robot r (i.e $H_L^O(r) \cup \mathcal{L}_V(r)$).
$H_B^O(r)$	Bottom open half for the robot r.
$H_B^C(r)$	Bottom closed half for the robot r (i.e $H_B^O(r) \cup \mathcal{L}_H(r)$).
$H_U^O(r)$	Upper open half for the robot r.
$H_U^C(r)$	Upper closed half for the robot r (i.e $H_U^O(r) \cup \mathcal{L}_H(r)$).
L_{t_j-1}	The horizontal line below the target position t_j.
K	The horizontal line passing through the middle point of the line segment between two robots with light `decider` or `terminal1` on the same vertical line.
L_{H1}	The immediate horizontal line above the robot with light `leader`.

Some additional definitions are needed to be explained which will be useful later.

Configuration: Let us consider a team of robots placed on an simple undirected connected graph $G = (V, E)$. Let us define a function $f : V \to \{0\} \cup \mathcal{N}$, where

$f(v)$ is the number of robots placed on vertex v. The graph G together with the function f is called a configuration which is denoted by $\mathbb{C} = (G, f)$. For ant time T, $\mathbb{C}(T)$ will denote the configuration of the robots at time T.

For a graph $G = (V, E)$, $\phi : V \to V$ is an automorphism if ϕ is a bijection and $\phi(u)\phi(v)$ is adjacent iff u and v are adjacent $\forall u, v \in V$. All the automorphisms of G form a group denoted by $Aut(G)$. Similarly we can define an automorphism ϕ for a configuration (G, f) where $\phi \in Aut(G)$ and $f(u) = f(\phi(u)), \forall u \in V$. All automorphisms on (G, f) form a group denoted by $Aut(G, f)$.

Symmetric configuration: For any configuration $\mathbb{C} = (G, f)$, we can define the group $Aut(\mathbb{C})$. $\phi(v) = v, \forall v \in V$ is called a trivial symmetry. Every non trivial $\phi \in Aut(\mathbb{C})$ is called a symmetry of $\mathbb{C}$. Note that all symmetric configurations of a configuration $\mathbb{C}$ is basically generated by some translations, rotations and reflections. Translation shifts all the vertices by the same amount. Since the number of robots in the configuration $\mathbb{C}$ is finite it is easy to see that there is no translation in $Aut(\mathbb{C})$. Reflections are defined by some axis or line of reflection. It can be vertical, horizontal or diagonal. The angle of rotation can be $90°$ or $180°$. The center of rotation can be a vertex of the grid, center of an unit square or a center of an edge.

Stable Configuration: A configuration $\mathbb{C}$ is called a stable configuration if the following conditions are satisfied in $\mathbb{C}$.

1. There are two robots with light `decider` on same vertical line and all other robots in $\mathbb{C}$ have light `off`.

2. The vertical and horizontal line on which the robots with light `decider` are located don't have any other robots.

3. The robots with light `decider` have no robots on left open half and also their upper closed half or bottom closed half have no other robots.

Leader Configuration: A configuration $\mathbb{C}$ is called a leader configuration if the following conditions are satisfied in $\mathbb{C}$.

1. There are exactly one robot with light `leader` and all other robots have light `off`.

2. The vertical line and the horizontal line on which the robot with light `leader` is located do not have other robots.

3. The robots with light `leader` has no robots on left open half and also upper open half or bottom open half is empty .

Compact Line: A line is called compact if there is no unoccupied grid position between any two robots on that line.

Terminal Robot: A robot r is called a terminal robot if $\mathcal{L}_V(r) \cap H$ is empty, where $H \in \{H_B^O(r), H_U^O(r)\}$.

Symmetry of a vertical line L w.r.t K: Let L be a vertical line of the grid and λ be a binary sequence defined on L such that j-th term of λ is defined as follows:

$$\lambda(j) = \begin{cases} 1 & \text{if } \exists \text{ a robot on the } j\text{-th grid point from } K \cap L \text{ on the line } L. \\ 0 & \text{otherwise.} \end{cases}$$

By definition of λ, it follows that there are two such values of λ, say λ_1 and λ_2. We say that the line L is symmetric with respect to K if $\lambda_1 = \lambda_2$. For future, whenever symmetry of a line is mentioned, it is assumed that it means the symmetry of the line with respect to K.

Dominant half: A robot r is said to be on the dominant half if the following conditions are satisfied:

1. $\mathcal{R}_I(r)$ is not symmetric with respect to K.

2. If lexicographically $\lambda_1 > \lambda_2$ on $\mathcal{R}_I(r)$, then r and the portion of $\mathcal{R}_I(r)$ corresponding to λ_1 lie on same half plane delimited by K.

4.4 The Algorithm

The main result of the chapter is Theorem 1. The proof of the 'only if' part is the same as in case for point robots, proved in [15]. The 'if' part will follow from the algorithm presented in this section.

Theorem 4.1. *For a set of opaque luminous robots having one axis agreement, $\mathcal{APF}$ is deterministically solvable if and only if the initial configuration is not symmetric with respect to a line K such that 1) K is parallel to the agreed axis and 2) K is not passing through any robot.*

For the rest of the chapter, we shall assume that the initial configuration $\mathbb{C}(0)$ does not admit the unsolvable symmetry stated in Theorem 4.1. Our algorithm works in two stages namely leader election and the pattern formation. The stages are described in details in 4.4.1 and 4.4.2.

4.4.1 Leader Election

In the first stage, Leader Election, the robots will agree on a leader. Since there are no common agreement on a global coordinate system, the robots will not be able to agree on the embedding of the pattern on the grid. Thus leader election is necessary for robots to agree on a global coordinate. This stage is further divided in two phases namely *Phase 1* and *Phase 2*. This phases are described in details in 4.4.1.1 and 4.4.1.2.

4.4.1.1 Phase 1

The procedure *Phase 1* starts from the initial configuration with the aim of forming a stable configuration or achieving a configuration with a robot with light `leader1`. Initially all the robots are located on an infinite grid with light `off`. On waking, each robot r will check if the robot is terminal and if their open left half has no other robots and no robot `leader1` in $\mathcal{R}_I(r)$. Note that there will be at most two and at least one such robot. These robots will change their lights to `terminal1` and move left (Figure 4.1).

Algorithm 4: Phase 1

```
1  Procedure PHASE1()
2      r ← myself
3      if r.light = off then
4          if r is terminal and there is no robot in H_L^O(r) and no robot leader1
             in R_I(r) then
5              r.light ← terminal1
6              Move left
7          else if there are exactly two robots in L_I(r) having light terminal1
             and r is on K then
8              r.light ← leader1
9      else if r.light = terminal1 then
10         if there is a robot with light terminal1 on L_V(r) then
11             if no robots on K ∩ R_I(r) then
12                 if R_I(r) is symmetric with respect to K then
13                     r.light ← symmetric
14                 else
15                     if r is in dominant half then
16                         r.light ← leader1
17             else if there is a robot on K ∩ R_I(r) with light leader1 then
18                 r.light ← off
19         else if there is a robot with light symmetric on L_V(r) then
20             r.light ← symmetric
21         else if there is a robot with light leader1 or off on L_V(r) then
22             r.light ← off
23         else if r is singleton on L_V(r) and all robots in R_I(r) are off then
24             r.light ← leader1
25     else if r.light = symmetric then
26         if there is a robot r' with light symmetric or decider on L_V(r) then
27             if there is other robot both in H_U^C(r) and H_B^C(r) then
28                 move vertically opposite to r'
29             else
30                 r.light ← decider
```

If there is only one robot (say, r) with light `terminal1` on $\mathcal{L}_1$ and light of all robots on $\mathcal{R}_I(r)$ are `off`, then r will change its light to `leader1` (Figure 4.2, 4.3). This might also happen due to asynchrony of the system that there are two

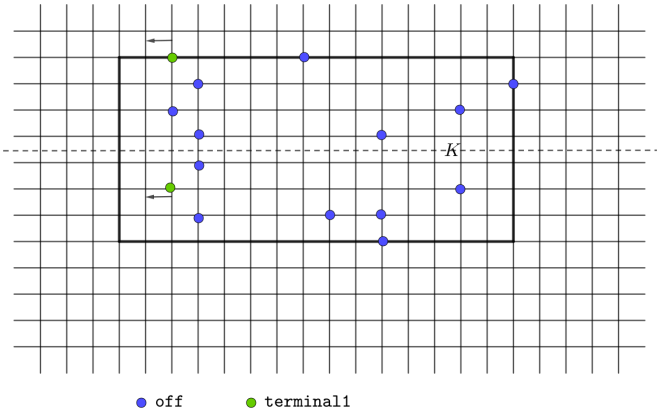


Figure 4.1: Terminal robots on the line $\mathcal{L}_1$ change lights to `terminal1` and move left.

robots with `terminal1` light but one (say, r_1) in $\mathcal{L}_1$ and the other one (say, r_2) is still in $\mathcal{R}_I(r_1)$. In this case, the robot on $\mathcal{L}_1$ will see that all robots on $\mathcal{R}_I(r_1)$ do not have lights `off` and will wait for r_2 to reach $\mathcal{L}_1$.

If there are two robots (say, r_1 and r_2) on $\mathcal{L}_1$ with `terminal1` light, then observe that both the robots r_1 and r_2 and all the robots on $\mathcal{R}_I(r_1)$ $(= \mathcal{R}_I(r_2))$ can recognise the line K. Now, if there exists a robot (say r_l) occupying the grid point $\mathcal{R}_I(r_1) \cap K$, then r_l changes its light to `leader1` and the robots with light `terminal1` changes their light to `off` after seeing the robot r_l with light `leader1` (Figure 4.4, 4.5). If the grid point $\mathcal{R}_I(r_1) \cap K$ is empty, then the robots r_1 and r_2 check the symmetry of the line $\mathcal{R}_I(r_1)$ $(= \mathcal{R}_I(r_2))$. If $\mathcal{R}_I(r_1)$ is not symmetric, then the robot $r_i (i = 1 \text{ or, } 2)$, which is on the dominant half changes its light to `leader1` (Figure 4.6, 4.7). On the other hand if $\mathcal{R}_I(r_1)$ is symmetric, then both the robots r_1 and r_2 change their lights from `terminal1` to `symmetric` (Figure 4.8, 4.9). Note that, due to asynchrony it might happen that one robot, let's say r_1 changes its light to `symmetric` before r_2. Then r_1 does not move until it sees another robot (r_2) on $\mathcal{L}_V(r_1)$ with light `symmetric`. This technique prevents the configuration from getting symmetric in this phase. Now after seeing another robot (r_2) on $\mathcal{L}_V(r_1)$ with `symmetric` light, r_1 moves vertically opposite of r_2 until the closed upper half or the closed bottom half of r_1 has no other robots (similar argument can be given for r_2) (Figure 4.10). After reaching their

designated positions both the robots r_1 and r_2 change their light from `symmetric` to `decider`, achieving a stable configuration (Figure 4.11). Note that, due to asynchrony it might happen that one robot, let's say r_1 reaches to its designated position and changes its light to `decider` before r_2.

The following Lemmas 4.1, 4.2 and 4.3 and Theorem 4 justify the correctness of the Algorithm 4.

Lemma 4.1. *If the initial Configuration* $\mathbb{C}(0)$ *has exactly one robot on* $\mathcal{L}_1$*, then* $\exists\ T_1 > 0$ *such that there will be exactly one robot with light* `leader1` *in* $\mathbb{C}(T_1)$*.*

Proof. Let us assume the initial configuration $\mathbb{C}(0)$ has exactly one robot (say, r) on $\mathcal{L}_1$. According to the Algorithm 4, when r wakes it will see that it is a terminal robot, the open left half is empty and no `leader1` in $\mathcal{R}_I(r)$. Then it changes the light to `terminal1` and moves left. Note that, after r moves left, $\mathcal{L}_1$ now denotes the new vertical line where r is located. Now on waking again, r sees that it is the only robot on $\mathcal{L}_1$ with light `terminal1` and all other robots on $\mathcal{R}_I(r)$ having light `off`. In this case, r changes the light to `leader1`. Note that all the other robots in this case have the light `off` throughout the Algorithm 4. So during the execution of Algorithm 4, if this case occurs, there will be exactly one robot with light `leader1` (Figure 4.2, 4.3).

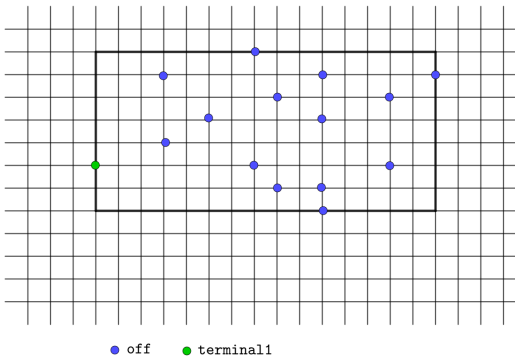


Figure 4.2: Single robot on $\mathcal{L}_1$ with light `terminal1`.

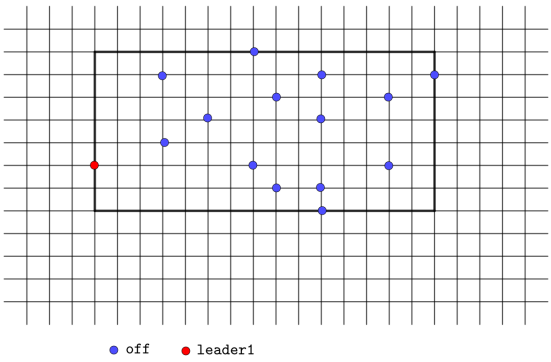


Figure 4.3: The single robot with light `terminal1` changes its light to `leader1`.

□

Lemma 4.2. *If the initial configuration* $\mathbb{C}(0)$ *has exactly two robots on* $\mathcal{L}_1$*, then* $\exists\ T' > 0$ *such that* $\mathbb{C}(T')$ *is either a stable configuration or there is exactly one robot with light* `leader1` *in* $\mathbb{C}(T')$*.*

Proof. Let r_1 and r_2 be two robots on $\mathcal{L}_1$ in the initial configuration $\mathbb{C}(0)$. In this case, both the robots change their lights to `terminal1` and move left. Note that after moving left, $\mathcal{L}_1$ now denotes the vertical line on which the robots are located now. Due to asynchrony of the system, two cases may occur:

Case 1: Both the robots r_1 and r_2 reaches $\mathcal{L}_1$ before waking again. In this case, r_1 and r_2 can see each other and can calculate the line K. Now if there is another robot (say r_l) on the intersection of K and $\mathcal{R}_I(r_1)$ ($\mathcal{R}_I(r_1) = \mathcal{R}_I(r_2)$), then r_l changes the light to `leader1`. Note that r_l can see both r_1 and r_2, so it can calculate the line K itself. After seeing r_l with light `leader1`, the robots with light `terminal1` change their lights to `off` (Figure 4.4, 4.5).

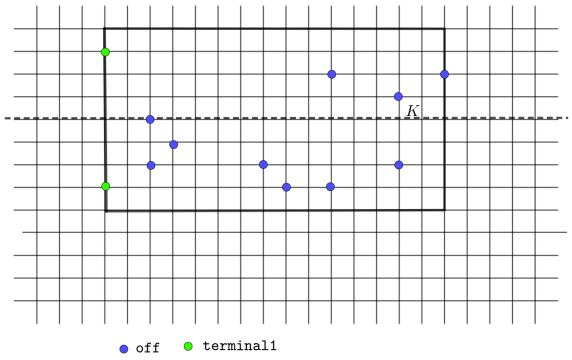


Figure 4.4: Both the robots with light `terminal1` see a robot on the right next occupied vertical line and on the line K.

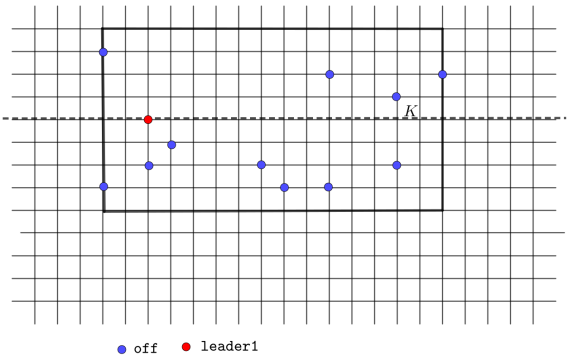


Figure 4.5: The robot on K changes its light to `leader1` and next the `terminal1` robots see robot `leader1` and change their lights to `off`.

On the other hand, if there is no robot on the intersection of K and $\mathcal{R}_I(r_1)$, then r_1 and r_2 both check the symmetry of $\mathcal{R}_I(r_1)$ with respect to the line K. If $\mathcal{R}_I(r_1)$ is asymmetric with respect to K, then the robot in the dominant half (r_1 or r_2) will change it's light to `leader1` (Figure 4.6, 4.7).

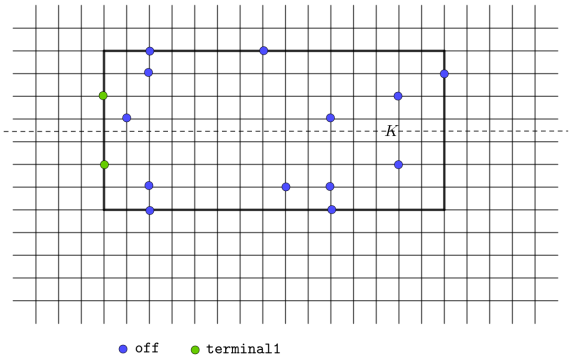


Figure 4.6: The robots with light `terminal1` see that the right next occupied vertical line is not symmetric.

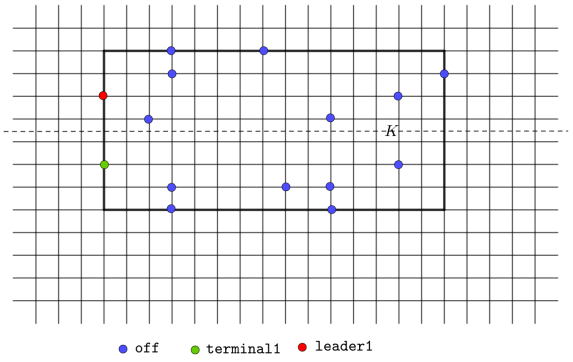


Figure 4.7: The robot with light `terminal1` on the dominant half changes the light to `leader1`.

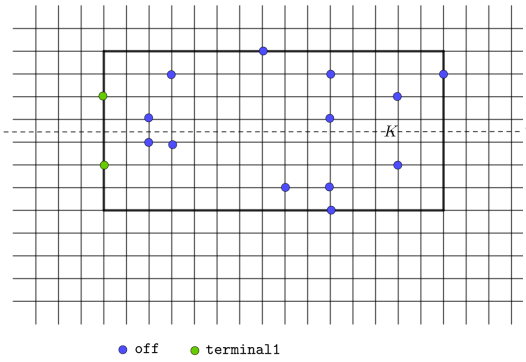


Figure 4.8: The `terminal1` robots see that the right next occupied vertical line is symmetric.

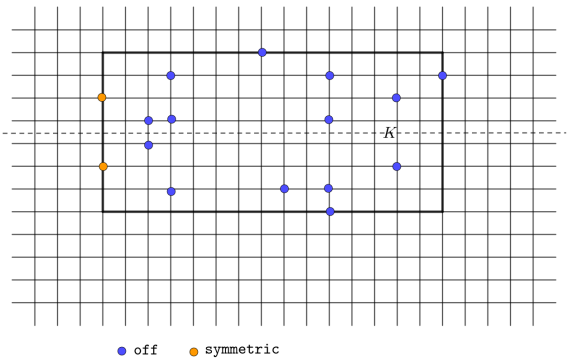


Figure 4.9: Robots with light `terminal1` change their lights to `symmetry`.

Otherwise if $\mathcal{R}_I(r_1)$ is symmetric with respect to K, then r_1 and r_2 change their light to `symmetric` (Figure 4.8, 4.9). Note that due to asynchrony, it may happen that one robot (say ,r_1) changes light to `symmetric` while r_2 still has the light `terminal1`. In this case, r_1 will not move until r_2 wakes and changes it's light to `symmetric`, after seeing the `symmetric` light of r_1. After both the robots r_1 and r_2 changed their lights to `symmetric`, they move vertically in the opposite direction of each other until one of the region H_C^U or, H_C^B has no other robots (Figure 4.10). After this step, both r_1 and r_2 change their lights from `symmetric`

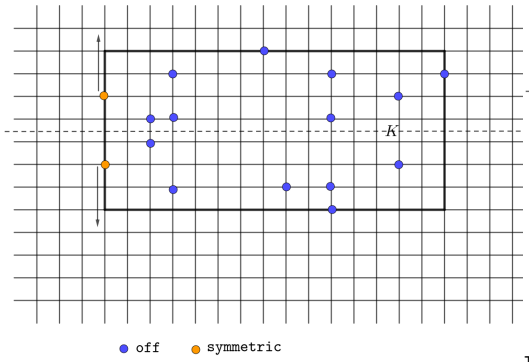


Figure 4.10: The robots with light `symmetry` move opposite of each other until they find either bottom or upper closed half has no other robots.

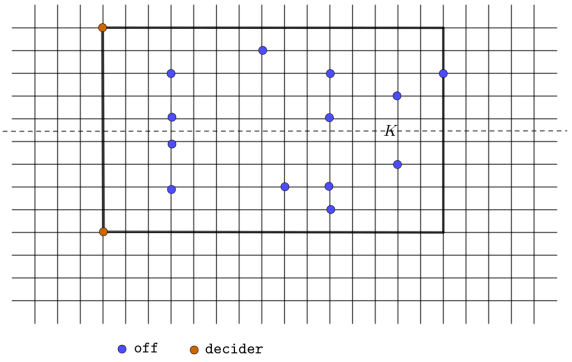


Figure 4.11: After reaching designated positions where upper closed or bottom closed half has no other robots, the robots with light `symmetry` change lights to `decider` to reach a stable configuration.

to `decider`, thus reaching a stable configuration (Figure 4.11). Note that the robots will not check the symmetry of $\mathcal{R}_I(r_1)(=\mathcal{R}_I(r_2))$ if the light of robots are `symmetric`.

Case 2: Lets assume r_1 is on $\mathcal{L}_1$ and awake before another robot r_2 with `terminal1` light is yet to reach on $\mathcal{L}_1$. In this case, r_1 checks $\mathcal{R}_I(r_1)$ and finds out all the robots except one (r_2 with light `terminal1`) have light `off`. In this case, r_1 waits until r_2 reaches $\mathcal{L}_1$ and then by similar argument like in case 1, either reaches a configuration with a robot having light `leader1` or reaches a stable configuration.

So after the execution of Algorithm 4, the configuration is either stable or has exactly one robot with light `leader1`. □

Lemma 4.3. *If the initial configuration* $\mathbb{C}(0)$ *has more than two robots on* $\mathcal{L}_1$*, then* $\exists\ T'' > 0$ *such that* $\mathbb{C}(T'')$ *is either a stable configuration or there is exactly one robot with light* `leader1` *in* $\mathbb{C}(T'')$*.*

Proof. Let us assume there are h robots on the line $\mathcal{L}_1$, denoted by $\{r_i : i \in [1, h] \cap \mathbb{N}$ and $h > 2\}$. Suppose r_1 and r_2 are the two terminal robots. Since r_1

and r_2 can identify themselves as terminal robots and can see there is no robot in $H_L^O(r)$ and no robot `leader1` in $\mathcal{R}_I(r)$, they will change their light to `terminal1` and move left. Note that after r_1 or r_2 move left, $\mathcal{L}_1$ now denotes the vertical line on which the robots are located now.

Now, with the same argument as in both the cases of Lemma 4.2, we can easily say that after the execution of Algorithm 4, the configuration is either stable or has exactly one robot with light `leader1`. □

Theorem 4.2. *For any initial configuration* $\mathbb{C}(0)$, $\exists\ T > 0$ *such that either* $\mathbb{C}(T)$ *is a stable configuration or has exactly one robot with light* `leader1`.

Proof. This follows directly from Lemma 4.1, Lemma 4.2 and Lemma 4.3. □

4.4.1.2 Phase 2

After completion of *Phase 1*, two configurations may occur. In the first possible configuration, there exists exactly one robot with light `leader1` and other robots with light `off`. The other possible configuration is a stable configuration. After completion of *Phase 2*, these configurations transforms into a leader configuration.

First, let us assume that after *Phase 1*, the configuration transforms into the stable configuration. In this configuration, there are two robots with light `decider` such that their $H_U^C \cap H_L^C$ or $H_B^C \cap H_L^C$ have no other robots. All the other robots have light `off`. In *Phase 2*, a robot with light `off` (say, r) checks whether it can see two robots with light `decider` on $\mathcal{L}_I(r)$. If r can see two such robots on $\mathcal{L}_I(r)$ and r is on $K \cap \mathcal{L}_V(r)$, then r changes its light to `leader1`. Note that r can calculate the line K as it can see both the robots having light `decider`. On the other hand, if r is not on $K \cap \mathcal{L}_V(r)$, then it checks the symmetry of $\mathcal{R}_I(r)$ with respect to K. If $\mathcal{R}_I(r)$ is not symmetric, then the terminal robot on $\mathcal{L}_V(r)$ on the dominant half changes its light to `leader1`. Otherwise, if $\mathcal{R}_I(r)$ is symmetric with respect to K, the robot closest to K and on $\mathcal{L}_V(r)$ changes its light to `call`. Note that a robot with light `off` also changes its light to `call` when it sees another robot with light `call` on the same vertical line. In this way,

if r is not on $K \cap \mathcal{L}_V(r)$ and $\mathcal{R}_I(r)$ is symmetric with respect to K, all the robots on $\mathcal{L}_V(r)$ will eventually change their lights to `call` (Figure 4.12, 4.13).

Now let r_{d1} be a robot with light `decider`. It checks whether there is a robot on $K \cap \mathcal{R}_I(r_{d1})$. Observe that in the beginning of *Phase 2*, r_{d1} can see another robot with light `decider` (say r_{d2}) on $\mathcal{L}_V(r_{d1})$, thus can calculate the line K. If there is no robot on $\mathcal{R}_I(r_{d1}) \cap K$ and $\mathcal{R}_I(r_{d1})$ is not symmetric with respect to K, then if r_{d1} was in dominant half, it changes its light to `leader1` (Figure 4.14). In this case, since there are two robots with light `decider` and both are terminal, one will always be in the dominant half. Now if there is no robot on $\mathcal{R}_I(r_{d1}) \cap K$ and the line $\mathcal{R}_I(r)$ (where, $r \in \mathcal{R}_I(r_{d1})$) is symmetric with respect to K, after a certain time all the robots on $\mathcal{R}_I(r_{d1})$ will have light `call`.

Algorithm 5: Phase 2

```
1  Procedure PHASE2()
2      r ← myself
3      if r.light = decider then
4          execute FUNCDECIDER()
5      else if r.light = off then
6          if there are two robots with light decider in L_I(r) then
7              if r is on K ∩ L_V(r) then
8                  r.light ← leader1
9              else
10                 if R_I(r) is symmetric with respect to K then
11                     if r is closest to K or there is a robot with light call on
                           L_V(r) then
12                         r.light ← call
13                 else
14                     if r is terminal on L_V(r) and in dominant half then
15                         r.light ← leader1
16     else if r.light = call then
17         if there is a robot with light leader1 in R_I(r) or L_I(r) then
18             r.light ← off
19     else if r.light = leader1 then
20         if (all robots in L_I(r) are off) or (H_L^O(r) is empty and all robots in
               R_I(r) are off) then
21             if there is no other robot in H_U^C(r) or H_B^C(r) then
22                 if there is other robot in H_L^C(r) then
23                     move left
24                 else
25                     r.light ← leader
26             else
27                 if r is not terminal on L_V(r) then
28                     move left
29                 else if r is terminal on L_V(r) and there is another robot r' on
                       L_V(r) then
30                     move vertically opposite to r'
31                 else if r is singleton on L_V(r) then
32                     move vertically according to its positive y − axis.
33         else if there is a robot with light leader1 in L_I(r) then
34             r.light ← off
```

Algorithm 6: FUNCDECIDER

```
1  Procedure FUNCDECIDER()
2      r ← myself
3      if there is a robot with light leader1 on R_I(r) or L_V(r) then
4          r.light ← off
5      else
6          if there is a robot with light decider on L_V(r) and no robots on
             K ∩ R_I(r) then
7              if R_I(r) is symmetric with respect to K then
8                  if all robots in R_I(r) are call then
9                      move right
10             else
11                 if r is in dominant half then
12                     r.light ← leader1
13         else if there is a robot with light decider in R_I(r) then
14             move right
15         else if there is a robot with light call in L_V(r) and no robot with light
             decider in L_I(r) then
16             if all robots in R_I(r) are call then
17                 move right
```

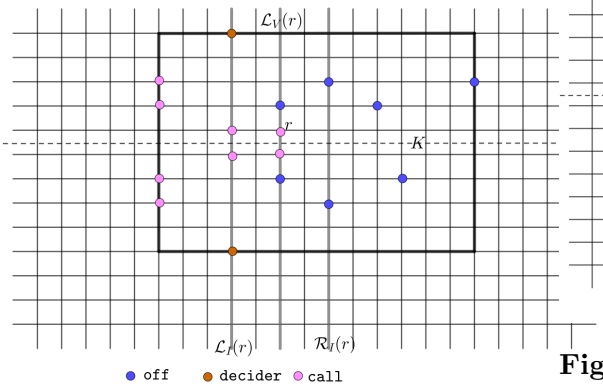


Figure 4.12: Robot r (closest of K) sees two `decider` robot on $\mathcal{L}_I(r)$ and $\mathcal{R}_I(r)$ is symmetric, so it changes light to `call`.

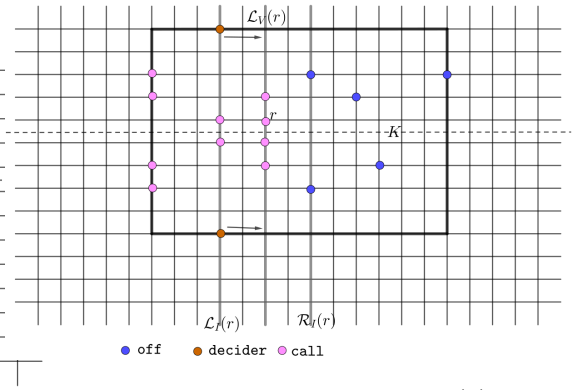


Figure 4.13: The robots on $\mathcal{L}_V(r)$, when see a robot with light `call` on the same vertical line, change their light to `call`. After all robots on $\mathcal{L}_V(r)$ have `call` light, the `decider` robots move right.

Then if $\mathcal{R}_I(r_{d1})$ $(= \mathcal{R}_I(r_{d2}))$ is symmetric, both the robots r_{d1} and r_{d2} move right. In case, due to asynchrony if one robot (say, r_{d1}) moves right and wakes before r_{d2} moves, then even if r_{d1} sees all robots on $\mathcal{R}_I(r_{d1})$ with light `call`, it will not move right until it sees there is no robot with light `decider` on $\mathcal{L}_I(r_{d1})$ and sees

a robot with light `call` or r_{d2} on $\mathcal{L}_V(r_{d1})$. Note that r_{d1} will see r_{d2} on $\mathcal{L}_I(r_{d1})$ until r_{d2} moves right. On the other hand, r_{d2} sees r_{d1} on $\mathcal{R}_I(r_{d2})$ and then it moves right.

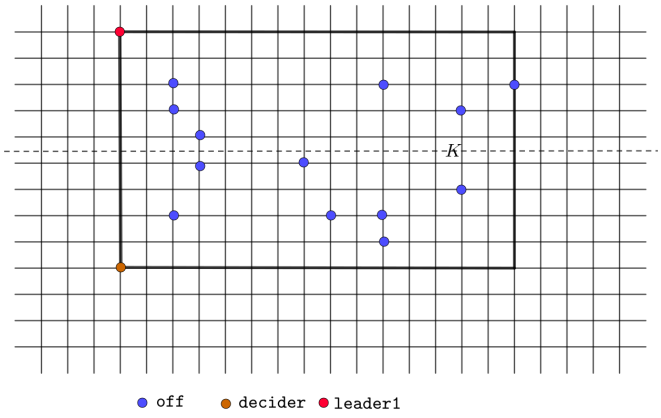


Figure 4.14: The robot r with light `decider` changes its light to `leader1` with $H_L^C(r)$ has no other robots.

On a lighter note, in this algorithm a robot r with light `off` basically calculates the line K by seeing the robots with light `decider` on $\mathcal{L}_I(r)$ and checks the symmetry of $\mathcal{R}_I(r)$ with respect to the line K and checks if it is on K. Note that, the robots with light `decider` also can calculate K when they see each other on the same vertical line and check the symmetry of the next vertical line having robots and also do search for robots on K. As the configuration is solvable and movement of robots with light `decider` do not actually make the configuration unsolvable, after a certain time there will be some robot r_0 with light `leader1`. Note that a robot r with light `call` changes the light to `off` if it sees any robot with light `leader1` on $\mathcal{R}_I(r)$ or $\mathcal{L}_I(r)$.

Now, let us consider the case where there is a robot with light `leader1` (say, r_0). It first checks whether all robots on the line $\mathcal{L}_I(r_0)$ have light `off` or if its left open half is empty. If one of the cases becomes true and all robots on $\mathcal{R}_I(r_0)$ have light `off`, then r_0 checks whether one of $H_U^C(r_0)$ or $H_B^C(r_0)$) has no other robots. If so and also there is other robots on the left closed half ($H_L^C(r_0)$) of

r_0, then it moves left until $H_L^C(r_0)$ has no other robots. Then it changes its light to `leader`. On the other hand, if both $H_U^C(r_0)$ and $H_B^C(r_0)$ have other robots, then r_0 checks whether it is a terminal robot on $\mathcal{L}_V(r_0)$. Let us assume r_0 is terminal on $\mathcal{L}_V(r_0)$. Now, if r_0 is singleton on $\mathcal{L}_V(r_0)$, then it moves according to its positive y-axis until either $H_U^C(r_0)$ or $H_B^C(r_0)$ has no other robots. If r_0 is not singleton but terminal on $\mathcal{L}_V(r_0)$, then there exists a robot (say, r') on $\mathcal{L}_V(r_0)$. In this case, r_0 moves vertically in the opposite direction of r' until $H_U^C(r_0)$ or $H_B^C(r_0)$ has no other robots. If r_0 was not terminal on $\mathcal{L}_V(r_0)$, then it moves left. Note that using such movements, r_0 always reaches a grid point such that either $H_U^C(r_0) \cap H_L^C(r_0)$ or $H_B^C(r_0) \cap H_L^C(r_0)$ has no other robots where it changes its light to `leader`. Also, it might happen that a robot with light `leader1` already sees another robot with light `leader1`. In this case, the robot with light `leader1`, who sees the other robot with light `leader1` on its left, changes light to `off`. Also if a robot with light `decider` (say, r) finds a robot with light `leader1` on $\mathcal{R}_I(r)$, then it changes its light to `off`. Thus after a certain time, there is exactly one robot with light `leader` and others have light `off`.

So, after completion of *Phase 2*, the configuration transforms into a leader configuration. The following Lemmas 4.4, 4.5, 4.6 and 4.7 justify the correctness of the Algorithm 5.

Lemma 4.4. *In Phase 2, a robot r with light `decider` changes its light to `leader1` only if $H_L^C(r)$ has no other robots.*

Proof. Let r and r' be two robots with light `decider` at some time $T > 0$ in *Phase 2* on the same vertical line. Without loss of generality, let us assume that r be the robot which changes its light at time $T_1 > T$ to `leader1` while it was still on the same vertical line at time T. Note that by Algorithm 5, this only happens if $\mathcal{R}_I(r)$ is not symmetric with respect to K and r is on the dominant half at T_1. We will denote the vertical line on which a robot r situated at any time t by $L_V^t(r)$.

Let us assume that $H_L^C(r)$ has other robots. Now note that a robot r with light `decider` moves right if it sees all the robots on $\mathcal{R}_I(r)$ have light `call` or sees

another robot with light `decider` on $\mathcal{R}_I(r)$. Also in the beginning of *Phase 2*, $H_L^C(r)$ $(= H_L^C(r'))$ has no other robots. So, at the time T, $H_L^C(r)$ has other robots and still has light `decider`, implies r and r' moved right at least once and did not see any robot with light `leader1` on $\mathcal{R}_I(r)$ till that time. Let us assume that at a time $(0 <)T_2 < T$, r and r' were on $L_V^{T_2}(r)$ which is actually $L_I(r)$ at time T. Now r and r' moves to $L_V^T(r)$ from $L_V^{T_2}(r)$ only when all robots on $L_V^T(r)$ have light `call`. That is only possible if at time T, the robots on $R_I(r)$ are symmetric with respect to K. Thus it is impossible for r to change its light to `leader1` while still on the same line $L_V^T(r)$ at time T_1. So our assumption that $H_L^C(r)$ has other robots at time T, is wrong. Hence the result follows (Figure 4.14).

□

Lemma 4.5. *If after a certain time $T > 0$ in Phase 2, the configuration $\mathbb{C}(T)$ has two robots r and r' both with light `leader1`, then $\exists\ 0 < T_1, T_2 < T$ such that r (or r') changed its light to `leader1` from `decider` at T_1 and r' (or, r) changed its light to `leader1` from `off` at T_2.*

Proof. First, we will show that both the robots r and r' can not have changed their lights to `leader1` from light `decider` at T_1 and T_2. From the Algorithm 5, it is clear that a robot r with light `decider` changes its light to `leader1` only if $\mathcal{R}_I(r)$ is not symmetric, $K \cap \mathcal{R}_I(r)$ is empty and r is in dominant half. Now both the robots with light `decider` can not be on the dominant half. So, both r and r' can not have change their lights to `leader1` from `decider` at T_1 and T_2.

Secondly, it has to be shown that both r and r' can not have changed their lights to `leader1` from light `off` at T_1 and T_2. From Algorithm 5, it is obvious that a robot r_0 with light `off` can only change its light to `leader1` if it sees two decider robots r_{d1} and r_{d2} on $\mathcal{L}_I(r_0)$. So if at a time $T' < T_1, T_2$ both r and r' had light `off`, then r and r' was on the same vertical line $\mathcal{L}_V(r)$ $(= \mathcal{L}_V(r'))$. Now we will show by contradiction that it is not possible for both r and r' to change their lights to `leader1`. Let us assume $\mathcal{R}_I(r)$ $(= R_I(r'))$ is symmetric with respect to K. Then r or r' will only change their lights to `leader1` if both are on the grid position $K \cap \mathcal{R}_I(r_{d1})$. Which is not possible. So, let us now assume $\mathcal{R}_I(r)$ is not

symmetric with respect to K and also let one of r or, r' is on $K \cap \mathcal{R}_I(r_{d1})$. Without loss of generality, let r is on $K \cap \mathcal{R}_I(r_{d1})$. Then according to the Algorithm 5, r changes its light to `leader1` but no other robot on $\mathcal{L}_V(r)$ changes their lights as they are not closest to K or sees no robot with light `call` on $\mathcal{L}_V(r)$. Hence in this case, there can be only one robot on $\mathcal{L}_V(r)$ who will change its light to `leader1`, arriving at a contradiction again. So let $\mathcal{R}_I(r)$ is not symmetric and there is no robot on the grid point $K \cap \mathcal{R}_I(r_{d1})$. Then only the terminal robot of $\mathcal{L}_V(r)$ who is in the dominant half changes its color to `leader1`. So again a contradiction that both r and r' on $\mathcal{L}_V(r)$ change their light to `leader1`. Hence our assumption was wrong.

Hence one of r or r' will change its light to `leader1` from light `decider` at a time $T_1 < T$ and the other robot will change its light to `leader1` from light `off` at time $T_2 < T$ (Figure 4.15, 4.16).

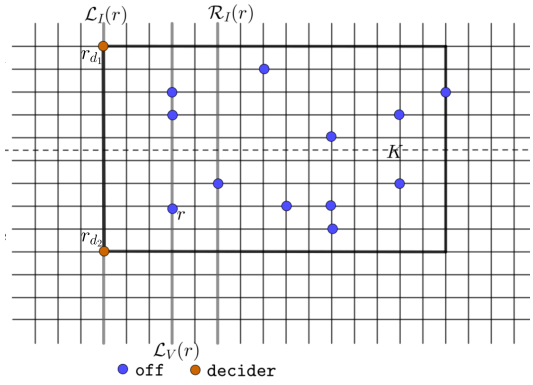


Figure 4.15: The robot r_{d1} with light `decider` and r with light `off` both are on the dominant half.

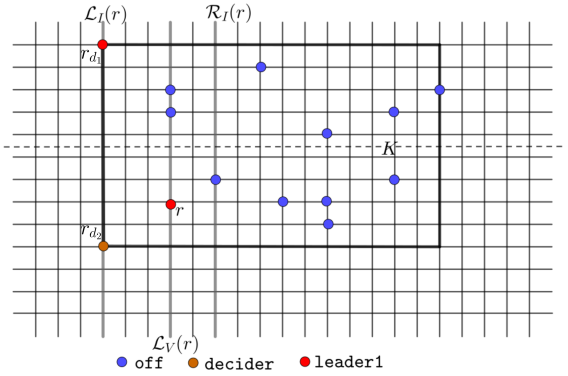


Figure 4.16: The robot r_{d1} with light `decider` and r with light `off` change their lights to `leader1`.

□

Lemma 4.6. *If in Phase 2, there exists a configuration $\mathbb{C}(T)$ at a time $T > 0$ such that $\exists$ a robot r' with light `leader1` which sees another robot r with light `leader1` on $L_I(r')$ in $\mathbb{C}$, then r' will not move and change its light to `off`.*

Proof. Since at time T, there are two robots with light `leader1` in the configuration during *Phase 2*, robot r must have changed its light to `leader1` from light

`decider` and the other robot r' must have changed the light to `leader1` from `off` (by Lemma 4.5). Now from Lemma 4.4, $H_L^C(r)$ has no other robots. Also r' is on $\mathcal{R}_I(r)$ (as r' can only change its light to `leader1` if it has seen two robots with light `decider` on $L_I(r')$). Thus it can be seen that if the configuration has two robots with light `leader1` at any time T, the two robots will be on two consecutive occupied vertical line. So by Algorithm 5, r' will see r on $L_I(r')$ with light `leader1` and change its light to `off` without moving (Figure 4.17, 4.18).

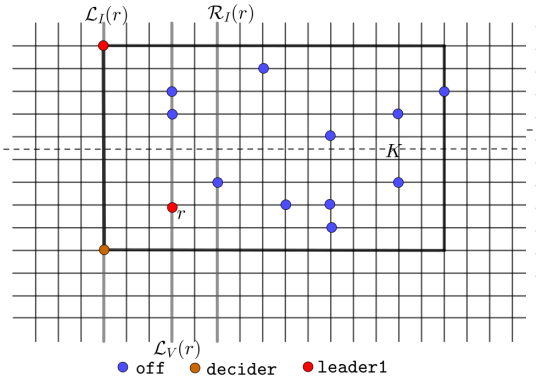


Figure 4.17: There are two robots with light `leader1` in the configuration.

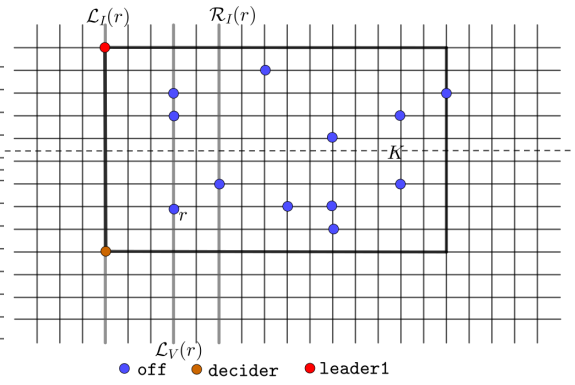


Figure 4.18: The robot r changes its light to `off` after seeing another `leader1` robot on $L_I(r)$.

□

Lemma 4.7. *If a robot r with light `leader1` is not terminal on $\mathcal{L}_V(r)$ and does not see any robot with light `leader1` on $\mathcal{L}_I(r)$, then $H_L^O(r) \cap \mathcal{L}_H(r)$ does not contain any robot.*

Proof. Let r is not terminal on $\mathcal{L}_V(r)$ with light `leader1`. Then by Algorithm 5, r is on $K \cap R_I(r_{d_1})$ ($\mathcal{R}_I(r_{d1}) = \mathcal{R}_I(r_{d2})$), where r_{d_1} and r_{d_2} be two robots with light `decider`. If possible let, there is a robot r' on $H_L^O(r) \cap \mathcal{L}_H(r)$. Then $\exists$ $T > 0$ such that r_{d_1} and r_{d_2} are on $L_I(r')$. Now observe that at time T, r' is on $K \cap R_I(r_{d_1})$, which implies r' will change its light to `leader1`. Then both r_{d_1} and r_{d_2} change their lights to `off` seeing r' on $R_I(r_{d_1})$ with light `leader1`. Hence r_{d_1} and r_{d_2} never reach $\mathcal{L}_I(r)$. Thus r can never change its light to `leader1`.

So we can conclude that our primary assumption was wrong. Hence the result follows (Figure 4.19).

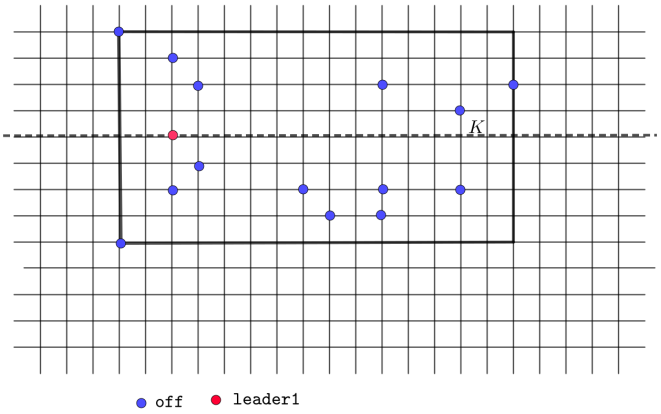


Figure 4.19: The robot with light `leader1` has its horizontal line empty on the left.

□

4.4.2 Pattern formation from leader configuration

In this phase, initially the configuration is a leader configuration. Note that the robots can agree on a global co-ordinate system based on the position of the robot r_0 with light `leader`. We denote the position of r_0 with the co-ordinate $(0, -1)$. Also all robots with light `off` lie on one of the open half planes delimited by the horizontal line $\mathcal{L}_H(r_0)$. This half plane will correspond to positive direction of Y-axis (Figure 4.20).

Therefore an agreement on a global co ordinate system can happen between the robots who see r_0 at $(0, -1)$. After completion of this phase, the robots achieve the target configuration (Figure 4.21). The robots first form a compact line and from that line the robots then move to their designated target positions. The difficulty of this phase is to differentiate between two configurations where a robot is going to form a compact line and where the robot is going to its target position which are described in subsections 4.4.2.1 and 4.4.2.2.

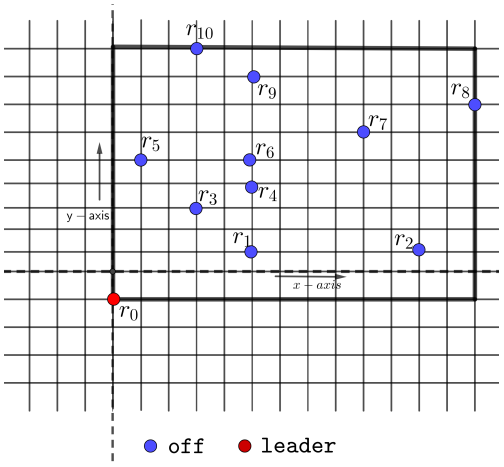


Figure 4.20: A leader configuration where robot r_0 is the robot with light `leader` at $(0, -1)$ in the agreed coordinate system.

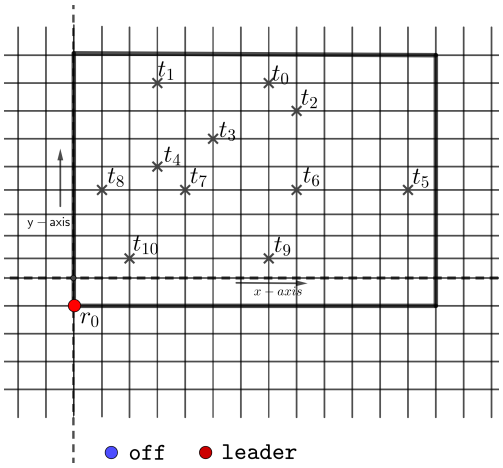


Figure 4.21: The pattern embedded in the coordinate system.

4.4.2.1 Compact line formation

Observe that according to the Algorithm 7, a robot with light `leader` will not move during the formation of line. Also at the beginning of *Phase 3*, there are no other robots on the line $\mathcal{L}_H(r_0)$. A robot r with light `off` will first check if it can see r_0 and if it is the leftmost robot on the line $\mathcal{L}_H(r)$ and also if there are any robots on $H_B^O(r) \cap H_O^U(r_0)$. If all the conditions are true, (i.e r is the leftmost robot in its horizontal line and there are no other robots in between horizontal lines of r and r_0) r counts the number of robots on $\mathcal{L}_H(r_0)$. Lets assume if there are no robots on $\mathcal{L}_H(r_0)$ except r_0. In this case, r checks for other robots with light `done` on the grid. Note that a robot changes its light to `done` only if it has reached its target position. So, clearly while forming the line, r finds that there are no robots with light `done` on the grid and moves to the position $(1, -1)$ following the procedure LINEMOVE(1). On the other hand if there are i robots except r_0 who are already in the line $\mathcal{L}_H(r_0)$ occupying the positions $(1, -1), (2, -1)...(i, -1)$, then r simply move to $(i + 1, -1)$ following the procedure LINEMOVE$(i + 1)$.

Algorithm 7: Pattern Formation from Leader Configuration

```
   Input The configuration of robots visible to me.
   :
1  Procedure PatternFormationFromLeaderConfiguration()
2      r ← myself
3      r_0 ← the robot with light leader
4      if r.light = off then
5          if (r_0 ∈ H_B^O(r)) and (r is leftmost on L_H(r)) and (there is no robot in
             H_B^O(r) ∩ H_U^O(r_0)) then
6              if there are no robots on L_H(r_0) other than r_0 then
7                  if there is a robot with light done then
8                      if r is at t_{n-2} then
9                          r.light ← done
10                     else
11                         TargetMove(n − 2)
12                 else
13                     LineMove(1)
14             3 else if there are i robots on L_H(r_0) other than r_0 at
                 (1, −1), ..., (i, −1) then
15                 LineMove(i + 1)
16             else if there are i robots on L_H(r_0) other than r_0 at
                 (n − i, −1), ..., (n − 1, −1) then
17                 if r is at t_{n-i-2} then
18                     r.light ← done
19                 else
20                     TargetMove(n − i − 2)
21         else if r_0 ∈ L_H(r) and H_U^O(r) has no robots with light off then
22             if r is at (i, −1) then
23                 Move to (i, 0)
24     else if r.light = leader then
25         if there are no robots with light off then
26             if r is at t_{n-1} then
27                 r.light ← done
28             else
29                 TargetMove(n − 1)
```

Algorithm 8: LineMove

```
1  Procedure LineMove(j)
2      r ← myself
3      if r is on L_{H1} then
4          if r is at (j, 0) then
5              Move to (j, −1)
6          else
7              Move horizontally towards (j, 0)
8      else
9          Move vertically towards L_{H1}
```

Algorithm 9: TARGETMOVE

```
1 Procedure TargetMove(j)
2     r ← myself
3     if r is on L_{t_j − 1} then
4         if r is at (t_j(x), t_j(y) − 1) then
5             Move to (t_j(x), t_j(y))
6         else
7             Move horizontally towards (t_j(x), t_j(y) − 1)
8     else
9         Move vertically towards L_{t_j − 1}
```

During the procedure LINEMOVE(j), a robot (say, r) can recognize if it is on the line L_{H1}, where L_{H1} is the immediate horizontal line above $\mathcal{L}_H(r_0)$. If it is not on L_{H1}, it moves vertically downwards until it reaches L_{H1}. Otherwise, if r is already on L_{H1}, it checks if it is on the co-ordinate $(j, 0)$. If it is not on $(j, 0)$, it moves horizontally to $(j, 0)$. Observe that during this horizontal movement, there will be no collision as r will be the only robot on L_{H1} due to the fact that the robots in Algorithm 7 move sequentially. Now, if r is on the co-ordinate $(j, 0)$, it moves vertically to the co-ordinate $(j, -1)$ (Figure 4.22, 4.23). Note that since the robots move sequentially, no robot will change there light to `done` before forming the compact line.

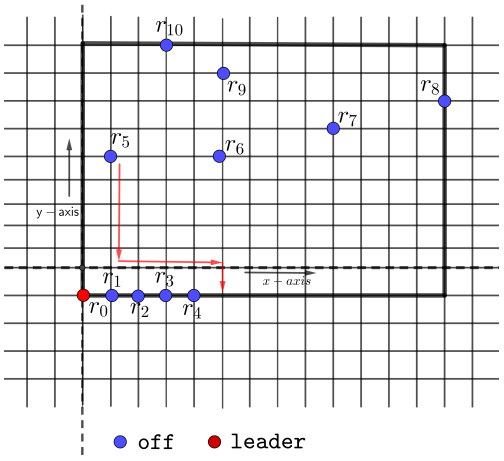


Figure 4.22: Movement of r_5 to $\mathcal{L}_H(r_0)$.

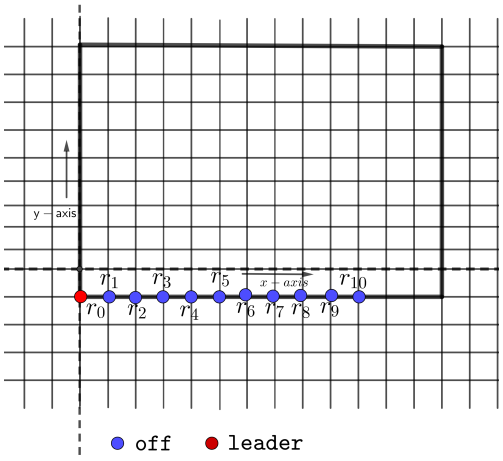


Figure 4.23: All robots forms an compact line on $\mathcal{L}_H(r_0)$.

4.4.2.2 Target Pattern Formation

After formation of the compact line, the robots now will move to the target co-ordinates. Note that the target co-ordinates are unique for each robot who can see the robot r_0 with light `leader` as there is a agreement on global co-ordinate. The target co-ordinates are denoted as t_i, where $i \in \{0, 1, 2, ...\overline{n-1}\}$. Also observe that if t_i and t_j are in the same horizontal line where $i < j$, then t_i is on the right of t_j. And if t_i and t_j are not on the same horizontal line, then t_i will be above of t_j.

If a robot sees r_0 on the same horizontal line, it vertically moves to L_{H1} with y-coordinate 0. Now observe that after moving to L_{H1} with y-coordinate 0, it can see all the robots on the line including the robot r_0.

From this information about the number of robots on the line except r_0, it can calculate the target position it need to reach and follow the procedure TARGET-MOVE() to reach that position. Note that the robot changes its light to `done` only after reaching its designated target position and in the mean time, no other robot moves from the line while they see robots with light `off` above them. This technique, of moving the robots sequentially and the ordering of the target co-ordinates, avoid collision in our algorithm. Thus all robots except r_0 and r_{n-1} reach their designated target co-ordinates. Now while r_{n-1} moves above from $(n-1, -1)$ to $(n-1, 0)$, it sees there is no other robot on the line $\mathcal{L}_H(r_0)$ except r_0 and moves to t_{n-2}. And finally the robot with light `leader` moves to the only remaining vacant target position t_{n-1}.

If a robot (say, r) executes the procedure TARGETMOVE(j), that means that the target position of r is t_j with co-ordinate $(t_j(x), t_j(y))$. Observe that, r executes this procedure when it is on L_{H1}. Now during this procedure, if r is not on L_{t_j-1}, then it moves vertically upwards until it reaches L_{t_j-1}. Now, when r is at L_{t_j-1}, it checks if the co-ordinate of its current position is $(t_j(x), t_j(y)-1)$. If not, then it moves horizontally to reach the point with the co-ordinate $(t_j(x), t_j(y)-1)$. After that it moves vertically upwards to the point with co-ordinate $(t_j(x), t_j(y))$, which is the target position of r (Figure 4.24, 4.25, 4.26). Note that during the

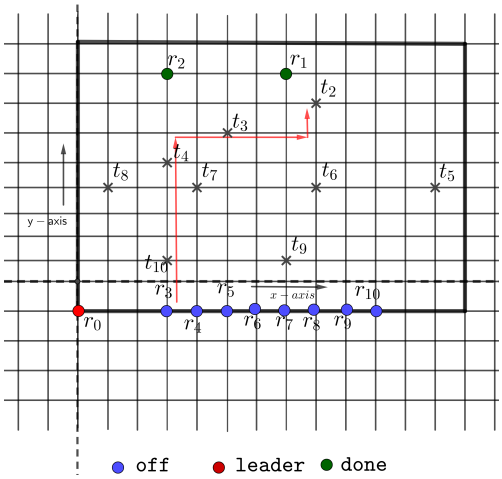


Figure 4.24: Movement of r_3 to t_2.

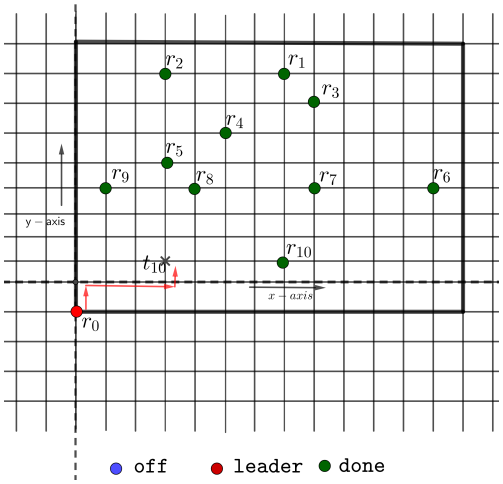


Figure 4.25: Movement of r_0 to t_{10}.

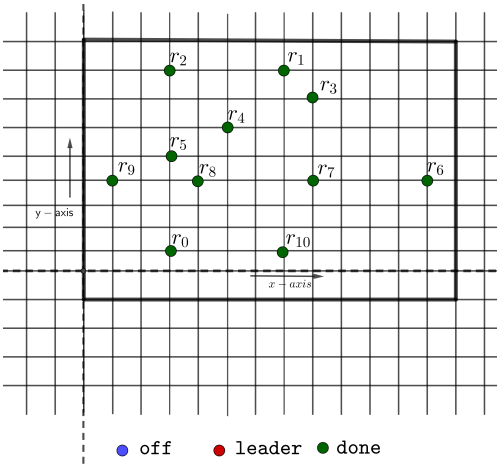


Figure 4.26: Target formation achieved.

movement of r, there will be no collision as the closed half delimited by L_{t_j-1} and $\mathcal{L}_H(r)$ does not contain any other robot.

Hence from the above discussions, we can conclude the following theorem.

Theorem 4.3. *If* $\mathbb{C}(T_1)$ *is a leader configuration, then* $\exists\ T_2 > T_1$ *such that* $\mathbb{C}(T_2)$ *is the target configuration.*

4.5 Conclusion

Arbitrary pattern formation ($\mathcal{APF}$) has been a very active topic in the field of swarm robotics. It has been thoroughly researched in many different settings. For example, it has been studied when the robots are on a plane or on an infinite grid. Considering obstructed visibility model for robots on a plane, it has been shown that for certain initial configurations, $\mathcal{APF}$ is solvable with opaque robots having one axis agreement and 6 lights under asynchronous scheduler([15]). In [11], $\mathcal{APF}$ has been solved even with opaque fat robots with light on plane. Comparing to how thoroughly $\mathcal{APF}$ has been studied with robots on plane with obstructed visibility model, it remains quite far behind when it comes to robots on infinite grid with obstructed visibility model. This chapter is a stepping stone towards the goal of removing this gap of research regarding $\mathcal{APF}$ between opaque robots on the plane and robots on infinite grid.

In this chapter, we have provided a deterministic algorithm for $\mathcal{APF}$ for all solvable initial configurations with one axis agreement and 8 lights under the asynchronous scheduler. For the immediate course of future research, one can think of solving this problem with less numbers of lights. Another interesting way of extending this problem would be to allow multiplicities in the pattern.

Chapter 5

Arbitrary Pattern Formation on Infinite Grid by Opaque Fat Robots

This chapter [1] deals with the problem of arbitrary pattern formation on an infinite grid using luminous opaque fat robots with 9 colours. The robots are considered to be a disk having a fixed radius '*rad*', which is less or equal to $\frac{1}{2}$. The robots manoeuvre in a Look-Compute-Move (LCM) cycle under an adversarial asynchronous scheduler. The robots are autonomous, anonymous, identical and homogeneous. The robots only move to one of its four adjacent grid points and their movement is considered to be instantaneous (i.e a robot can only be seen on a grid point). The robots have one-axis agreement. Here, it is assumed that the robots do not agree upon any global coordinate though all robots agree on the direction and orientation of the x-axis . Initially, the centre of each robot is on a grid point of the infinite grid and a target pattern is provided to each of them. The robots are needed to agree on a global coordinate system and embed the target pattern according to the global coordinate and then move to the target locations to form the target pattern.

[1] Based on this chapter, the following paper has been published:
Manash Kumar Kundu, Pritam Goswami, Satakshi Ghosh and Buddhadeb Sau. **Arbitrary pattern formation by opaque fat robots on infinite grid**. International Journal of Parallel, Emergent and Distributed Systems, Vol. 37, pages 542-570, 2022. https://doi.org/10.1080/17445760.2022.2088750

The main difficulty of $\mathcal{APF}$ lies in the problem of Leader Election problem. For that, the initial configuration is assumed to be asymmetric or there is at least one robot on some line of symmetry. Even with this assumption, it is quite hard to elect a leader as the vision of the robots becomes obstructed since the robots are opaque and fat. So, the main challenge of this problem is to elect a leader depending on the local view of each robot. The algorithm described in this chapter does so. Another massive challenge of this problem is to avoid collision during the movement of robots on the grid. Our algorithm handles this by providing sequential movement of the robots and for this purpose $\mathcal{LUMI}$ model has been used.

The problem, we have considered in this chapter, is very practical in nature. Restricted movement, robots with dimension and obstructed visibility all these assumptions are very much practical in terms of designing robots. The algorithm presented in this chapter solves the $\mathcal{APF}$ on infinite grid with a swarm of luminous, opaque and fat robots with finite time. A comparison table is provided below which will help readers to compare our work to the previous such works.

Paper	**Environment**	**Visibilty**	**Robot Type**	**#Colours**
[13]	Grid	Unobstructed	Point	0
[14]	Plane	Opaque	Point	6
[72]	Grid	Opaque	Point	8
[12]	Plane	Opaque	Fat	10
This chapter	Grid	Opaque	Fat	9

5.1 Model and Definitions

In this section, we present the model, some definitions and notations.

5.1.1 Model

Grid: The infinite two-dimensional grid $\mathcal{G}$ is a weighted graph $\mathcal{G} = (V, E)$ such that each node $v \in V$ has four adjacent nodes v_0, v_1, v_2 and $v_3 \in V$ and the edges $vv_{i \pmod 4} \in E$ is perpendicular to the edge $vv_{i+1 \pmod 4} \in E$. Also, the weight of each edge $e \in E$ is basically the length of the edge e which is considered to be 1 unit in this work.

Robots: In this work, a set of n robots $R = \{r_0, r_1, \ldots, r_{n-1}\}$ are considered to be autonomous, anonymous, homogeneous and identical. This means that the robots do not have any central control, they do not have any unique identifiers such as IDs and they are indistinguishable by their physical appearance. The robots are also considered to have some dimension i.e. the robots are considered to be a disk of radius 'rad'($rad \leq \frac{1}{2}$) rather than points. The robots are deployed on a two-dimensional infinite grid $\mathcal{G}$, where each of them is initially positioned in such a way that their centre is on distinct grid points of $\mathcal{G}$. The robots are considered to have an agreement over the direction and orientation of x-axis i.e, all the robots have an agreement over left and right but the robots do not have any agreement over the y-axis. Also, they do not have knowledge of any global coordinate system other than their agreement over the direction of x-axis. Here in this chapter, we have considered the robots to have light. A light of any robot can have $\mathcal{O}(1)$ distinct colours. A robot $r \in R$ can see the colour of its own light and the colour of the lights of other robots that are visible to r. In this work, we have assumed that the light of each robot has nine distinct colours namely `off`, `terminal1`, `candidate`, `call`, `moving1`, `reached`, `leader1`, `leader` and `done`.

Look-Compute-Move cycles: A robot $r \in R$, when active, operates according to the Look-Compute-Move (LCM) cycle. In the Look phase, a robot takes the snapshot of the configuration to get the positions represented in its own local coordinate system and the colours of the light of all other robots visible to it. Then, r performs the computation phase where it decides the position of the adjacent grid point where it will move next and changes the colour of its light if necessary depending on the input it got from the Look phase. In the Move phase, r moves to the decided grid point or makes a null move. The movements

of robots are restricted only along grid lines from one grid point to one of its four adjacent grid points. The movements of robots are assumed to be instantaneous in discrete domain. Here, we assume that the movements are instantaneous i.e., they are always seen on grid points, not on edges.

Scheduler: We assume that the robots are controlled by an asynchronous adversarial scheduler. That implies the duration of the three phases LOOK, COMPUTE and MOVE are finite but unbounded. So, there is no common notion of round for this asynchronous scheduler.

Visibility: The visibility of robots is unlimited but by the presence of other robots it can be obstructed. A robot r_i can see another robot r_j if and only there is a point p_{r_j} on the boundary of r_j and p_{r_i} on the boundary of r_i such that the line segment $\overline{p_{r_i}p_{r_j}}$ does not intersect with any point occupied by other robots in the configuration. Now, it follows from the definition that r_i can see r_j implies r_j can see r_i.

5.1.2 Notations and Definitions

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

$\mathcal{L}_1$	First vertical line on left that contains at least one robot.
$\mathcal{L}_V(r)$	The vertical line on which the robot r is located.
$\mathcal{L}_H(r)$	The horizontal line on which the robot r is located.
$\mathcal{L}_I(r)$	The left immediate vertical line of robot r which has at least one robot on it.
$\mathcal{R}_I(r)$	The right immediate vertical line of robot r which has at least one robot on it.
$H_L^O(r)$	Left open half for the robot r.
$H_L^C(r)$	Left closed half for the robot r (i.e $H_L^O(r) \cup \mathcal{L}_V(r)$).
$H_B^O(r)$	Bottom open half for the robot r.
$H_B^C(r)$	Bottom closed half for the robot r (i.e $H_B^O(r) \cup \mathcal{L}_H(r)$).
$H_U^O(r)$	Upper open half for the robot r.
$H_U^C(r)$	Upper closed half for the robot r (i.e $H_U^O(r) \cup \mathcal{L}_H(r)$).
K	The horizontal line passing through the middle point of the line segment between two robots with light `candidate` or `call` or `reached` on the same vertical line.
$l_{next}(r)$	The next vertical line on the right of $\mathcal{L}_V(r)$.
$\mathcal{H}_{last}$	The lowest horizontal line having a robot with colour `done`.

Configuration: We assume that the robots are placed on the infinite two-dimensional grid $\mathcal{G}$. Next we define a function $f : V \to \{0, 1\}$, where $f(v)$ is the number of robots placed on a grid point v. Then $\mathcal{G}$ together with the function f is called a configuration which is denoted by $\mathbb{C} = (\mathcal{G}, f)$. For any time T, $\mathbb{C}(T)$ will denote the configuration of the robots at T.

Terminal Robot: A robot r is called a terminal robot if there is no robot below or above r on $\mathcal{L}_V(r)$.

Symmetry of a vertical line L w.r.t K: Let λ be a binary sequence defined on a vertical line L such that i-th term of λ is defined as follows:

$$\lambda(i) = \begin{cases} 1 & \text{if } \exists \text{ a robot on the } i\text{-th grid point from } K \cap L \text{ on the line } L. \\ 0 & \text{otherwise.} \end{cases}$$

Since there are two i-th grid points from $K \cap L$ on the line L (above K and below K), there are two such values of λ, say λ_1 and λ_2. If $\lambda_1 = \lambda_2$, then L is said to be symmetric with respect to K. Otherwise, it is said to be asymmetric with respect to K. Henceforth, whenever the symmetry of a line is mentioned, it means the symmetry of the line with respect to K.

Dominant half: A robot r is said to be in the dominant half if for $\lambda_1 > \lambda_2$ (lexicographically) on $\mathcal{R}_I(r)$, r and the portion of $\mathcal{R}_I(r)$ corresponding to λ_1 lie on same half-plane delimited by K.

5.2 The Algorithm

The main result of the chapter is Theorem 5.1. The proof of the 'only if' part is the same as in the case for point robots, proved in [14]. The 'if' part will follow from the algorithm presented in this section.

Theorem 5.1. *For a set of opaque fat robots having one-axis agreement, $\mathcal{APF}$ is deterministically solvable if and only if the initial configuration is not symmetric with respect to a line K such that 1) K is parallel to the agreed axis and 2) K is not passing through any robot.*

For the rest of the chapter, we shall assume that the initial configuration $\mathbb{C}(0)$ does not admit the unsolvable symmetry stated in Theorem 5.1. Our Algorithm executes in two phases. In the first phase, a leader is elected and in the second phase, the robots form the target pattern embedded on the grid using the location of the leader as an agreement to the origin of a global coordinate system. The phases are described in detail in the following subsections 5.2.1 and 5.2.2.

5.2.1 Phase 1

In the Phase 1, the robots will agree on a leader. Since there are no common agreement on a global coordinate system, the robots will not be able to agree on

Algorithm 10: AppFatGrid: Phase 1

```
1  Procedure Phase1()
2      r ← myself
3      if r.light = off then
4          execute FuncOff()
5      else if r.light = terminal1 then
6          if there is no robot in H_L^O(r) then
7              r.light ← candidate
8              move left
9          else if there is a robot with light candidate in L_I(r) then
10             r.light = off
11     else if r.light = candidate then
12         execute FuncCandidate()
13     else if r.light = moving1 then
14         execute FuncMoving1()
15     else if r.light = call then
16         if there is a robot with light moving1 or, reached on L_V(r) and all robots in
             R_I(r) are off then
17             r.light = reached
18         else if there is a robot with light leader1 in R_I(r) then
19             r.light = off
20     else if r.light = reached then
21         if there is a robot with light reached or candidate on L_V(r), r is terminal
             on L_V(r) and all robots in R_I(r) are off then
22             r.light = candidate
23     else if r.light = leader1 then
24         if there is other robot in H_L^C(r) or l_next(r), no robot with light call in L_I(r)
             and no robot with light candidate on L_V(r) then
25             move left
26         else
27             if there is other robot both in H_U^C(r) and H_B^C(r) then
28                 move vertically according to its positive y−axis
29             else
30                 r.light = leader
```

Algorithm 11: FuncOff

```
1  Procedure FuncOff()
2      r ← myself
3      if there is no robot in H_L^O(r), no robot with light leader1 in R_I(r) ∪ L_V(r) and r
         is terminal on L_V(r) then
4          r.light ← terminal1
5      else if there are exactly two robots in L_I(r) and their lights are call and r is
         closest to K then
6          if r is on K then
7              r.light = leader1
8          else
9              r.light = moving1
10     else if there is a robot with light moving1 in L_V(r) then
11         r.light = moving1
```

Algorithm 12: FuncCandidate

```
1  Procedure FuncCandidate()
2      r ← myself
3      if r is singleton in H_L^C(r) and all robots in R_I(r) are off then
4          r.light ← leader1
5      else if there is a robot with light candidate or call on L_V(r), r is terminal on
         L_V(r) and all robots in R_I(r) are off then
6          if R_I(r) is symmetric with respect to K then
7              r.light = call
8          else
9              if r is in the dominant half then
10                 r.light = leader1
11     else if there is a robot with light leader1 on L_V(r) then
12         r.light = off
```

Algorithm 13: FUNCMOVING1

```
1  Procedure FUNCMOVING1()
2      r ← myself
3      if there is at least one robot with light call and no robot with light reached in
         L_I(r) and r is terminal on L_V(r) then
4          if there is other robot both in H_U^C(r) ∩ L_I(r) and H_B^C(r) ∩ L_I(r) then
5              if there is a robot r' on L_V(r) then
6                  move opposite to r'
7              else
8                  move according to its positive y−axis
9          else
10             move left
11     else if there is a robot with light reached on L_V(r) and all robots in R_I(r) are
         off then
12         move left
13     else if there is at least one robot with light reached or candidate in L_I(r) then
14         r.light = off
```

the embedding of the pattern on the grid. Thus leader election is necessary for robots to agree on a global coordinate.

Initially, at $\mathbb{C}(0)$ all the robots are on the grid $\mathcal{G}$ with colour `off`. Note that in $\mathbb{C}(0)$, there are at least one and at most two terminal robots on $\mathcal{L}_1$. These robots change their colours to `terminal1`. A robot with colour `terminal1` changes its colour to `candidate` and moves if it sees it has its left open half empty. Also, if a robot r with colour `candidate` is a singleton in $H_L^C(r)$ and all robots in $\mathcal{R}_I(r)$ are `off`, it changes its colour to `leader1`. Note that due to the asynchronous scheduler, it might happen that r is a singleton in $H_L^C(r)$ with colour `candidate` and there is another robot r' on $\mathcal{R}_I(r)$ with colour `terminal1`. In this case, if r awakes, it does not change its colour to `leader1` as it does not see all robots on $\mathcal{R}_I(r)$ have colour `off`. Also if r' awakes, it sees r with colour `candidate` in $\mathcal{L}_I(r')$ and turns its colour to `off`. In this scenario, r becomes singleton in $H_L^C(r)$ and sees all robots on $\mathcal{R}_I(r)$ have colour `off`. So, r changes its colour to `leader1`. Now consider that both r and r' are on the same vertical line $\mathcal{L}_V(r)$ with colour `candidate` such that there is no robot between $\mathcal{L}_H(r)$ and $\mathcal{L}_H(r')$. Note that in this configuration, all robots on $\mathcal{R}_I(r)$ (i.e. $\mathcal{R}_I(r')$) have colour `off`. In this case, both r and r' check the symmetry of $\mathcal{R}_I(r)$ with respect to the line

K (i.e the horizontal line which is equidistant from both r and r'). If $\mathcal{R}_I(r)$ is not symmetric with respect to K, then one of r or r' whichever is on the dominant half changes the colour to `leader1`. On the other hand, if $\mathcal{R}_I(r)$ is symmetric, both the robots r and r' change their colours to `call` from `candidate`. Now all the robots on $\mathcal{R}_I(r)$ have colour `off` and all of them can see exactly two robots with colour `call` on their left immediate line. Note that since all the robots on $\mathcal{R}_I(r)$ can see both r and r', all of them also know the line K. Now if there is any robot on K, it changes its colour to `leader1`. Otherwise, the robots on $\mathcal{R}_I(r)$ (at least one and at most two robots) which are closest to K change the colours to `moving1`. A robot with colour `off` on $\mathcal{R}_I(r)$ which is not closest to K, changes its colour to `moving1` when it sees another robot with colour `moving1` on the same vertical line. Note that after a finite time, at least all robots either above or below K which are on $\mathcal{R}_I(r)$ change their colours to `moving1` if $\mathcal{R}_I(r)$ is symmetric with respect to K. Now suppose a robot with colour `moving1` say r_1, is terminal on $\mathcal{R}_I(r)$. Also, note that r_1 can see at least one of r and r' on $\mathcal{L}_I(r_1)$. Now, if r_1 sees another robot r_2 on $\mathcal{L}_V(r_1)$ and no robot with colour `reached` on $\mathcal{L}_I(r_1)$, it moves vertically opposite to r_2. Otherwise, if it is singleton on $\mathcal{L}_V(r_1)$ and sees no robot with colour `reached` on $\mathcal{L}_I(r_1)$, it moves vertically according to its positive $y-$axis until there is no robot either in $H_U^C(r_1) \cap \mathcal{L}_I(r_1)$ or in $H_B^C(r_1) \cap \mathcal{L}_I(r_1)$ and then towards left until it reaches $\mathcal{L}_V(r)$. Now when r or r' with colour `call` sees all robots on $\mathcal{R}_I(r)$ have colour `off` and sees a robot with colour `moving1` or `reached` on the same vertical line, then it changes its colour to `reached`. Due to the asynchronous environment, it might happen that one of r or r' does not see a robot with colour `moving1` on the same vertical line, but it is guaranteed that it will see a robot with colour `reached` on the same vertical line after a finite time. So, r or r' can change their colours to `reached` if all robots on $\mathcal{R}_I(r)$ have colour `off` and there is a robot on $\mathcal{L}_V(r)$ with colour `reached`. So in finite time, both the robots with colour `call` change their colours to `reached` (when there was no robot on $K \cap \mathcal{R}_I(r)$). Now a robot say r_3 with colour `moving1` on $\mathcal{L}_V(r)$ moves to the left if all robots on $\mathcal{R}_I(r_3)$ are with colour `off` and it can see a robot with colour `reached` on $\mathcal{L}_V(r_3)$. Due to asynchrony, it may happen that r and r' changed their colours to `reached` and after that a

robot say r_4, on $\mathcal{R}_I(r)$ changes its colour to `moving1`. Observe that in this case, the robot r_4 changes its colour to `off` whenever it sees at least one robot with colour `reached` on $\mathcal{L}_I(r_4)$, otherwise the robots with colour `moving1` on $\mathcal{L}_V(r)$ will not move left. So, after a finite time, all robots with colour `moving1` on $\mathcal{L}_V(r)$ move to $\mathcal{L}_1$ and at this moment r and r' will be the only two robots with colour `reached` on $\mathcal{L}_V(r)$ that are terminal also. In this situation, r and r' change their colours to `candidate`. Now for asynchrony, it may happen that r and r' changed their colours to `candidate` and after that a robot say r_5, on $\mathcal{R}_I(r)$ changes its colour to `moving1`. In this case, the robot r_5 changes its colour to `off` whenever it sees at least one robot with colour `candidate` on $\mathcal{L}_I(r_5)$. Therefore, then r and r' are with colours `candidate` and all robots on $\mathcal{R}_I(r)$ have colour `off`. So, they again check the symmetry of the new $\mathcal{R}_I(r)$ repeating the whole process. Thus after a finite time, a robot with colour `off` or a robot r or r' with colour `candidate` whoever is on dominant half changes its colour to `leader1`. A robot say r_l with colour `leader1` always moves to the left when it sees other robot in $H_L^C(r_l)$ or $l_{next}(r_l)$, no robot with colour `call` on $\mathcal{L}_I(r_l)$ and no robot with colour `candidate` on $\mathcal{L}_V(r_l)$.

Note that, a robot with colour `call` changes its colour to `off` if it sees a robot with colour `leader1` on its right immediate line. Also, a robot with colour `candidate` changes the colour to `off` when it sees a robot with colour `leader1` on the same vertical line. r_l moves to the left until it becomes the singleton robot on the leftmost line of the configuration and there is no robot on $l_{next}(r_l)$ and then moves according to its positive $y-$axis until either one of $H_U^C(r_l)$ and $H_B^C(r_l)$ has no other robot. Note that it may happen due to the asynchronous environment that another robot with colour `candidate` moves to $l_{next}(r_l)$ while r_l is on $\mathcal{L}_1$. In this case, when r_l activates again it finds out it has non-empty $l_{next}(r_l)$ and moves left again even it was moving vertically in the previous activation. In this situation, when r_l reaches a point where either one of $H_U^C(r_l)$ and $H_B^C(r_l)$ has no other robot, it changes its colour to `leader` and *Phase 1* ends.

The following Theorem 5.2 and Lemmas 5.1−5.13 justify the correctness of the Algorithm 1.

Theorem 5.2. *For any initial configuration* $\mathbb{C}(0)$, $\exists\ T > 0$ *such that* $\mathbb{C}(T)$ *have exactly two robots with light* `candidate` *or exactly one robot with light* `leader1` *in* L_1.

Proof. Observe that there can be at least one and at most two robots in $\mathbb{C}(0)$ such that they have their left open half empty and are terminal on $\mathcal{L}_1$. Let there is only one robot r_1, who has $H_L^O(r_1)$ empty and is terminal on $\mathcal{L}_1$ (Figure 5.1). This implies r_1 is singleton on $\mathcal{L}_1$. In this case, r_1 changes its colour to `terminal1` at some time $T' > 0$ and eventually changes to `leader1` at a time $T > T'$.

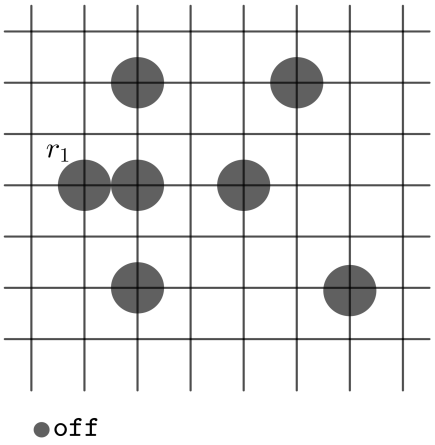


Figure 5.1: r_1 is singleton robot on $\mathcal{L}_1$.

Now let us consider the case where there are two robots r_1 and r_2 such that both r_1 and r_2 are terminal on $\mathcal{L}_1$ in $\mathbb{C}(0)$. Now if any one of r_1 or r_2 awakes, it changes its colour to `terminal1`. A robot with colour `terminal1` moves left after changing its colour to `candidate` if it has its left open half empty. Due to asynchronous environment, the following cases may occur.

Case-I: Let us consider the case where r_1 already changed its colour to `candidate` from `terminal1` and moved to $\mathcal{L}_1$ at a time $T_1 > 0$ and r_2 wakes after T_1 (Figure 5.2). Then r_2 remains with colour `off` as it sees it is not on $\mathcal{L}_1$ anymore. Then r_1 during the next activation sees it is singleton on $H_L^C(r_1)$ and all robots on $\mathcal{R}_I(r_1)$ have colour `off`. So, it changes its colour to `leader1`.

Case-II: Let us consider the case where r_1 already changed its colour to `candidate` from `terminal1` and moved to $\mathcal{L}_1$ at a time $T_1 > 0$ and r_2 wakes before T_1 and changes its colour to `terminal1` at a time $T_2 \geq T_1$ (Figure 5.3). Now if again r_1 wakes between the times T_1 and T_2 (in this scenario $T_1 > T_2$), then it sees all robots on $\mathcal{R}_I(r_1)$ have colour `off` and r_1 is singleton on $H_L^C(r_1)$. So, r_1 changes its colour to `leader1` and r_2 does not change its colour as $H_L^C(r_2)$ has other robots. Now if r_1 wakes at a time T_3 where $T_3 > T_2$ and r_2 has not woke again, then it does not change its colour to `leader1` as it sees r_2 with colour `terminal1` on $\mathcal{R}_I(r_1)$. Now when r_2 wakes again at a time $T_4 > T_2$, it changes its colour to `off` as it sees r_1 with colour `candidate` on $\mathcal{L}_I(r_2)$. Now when r_1 wakes after T_4 again, it sees it is singleton on $H_L^C(r_1)$ and have all robots with colour `off` on $\mathcal{R}_I(r_1)$ and so changes its colour to `leader1`.

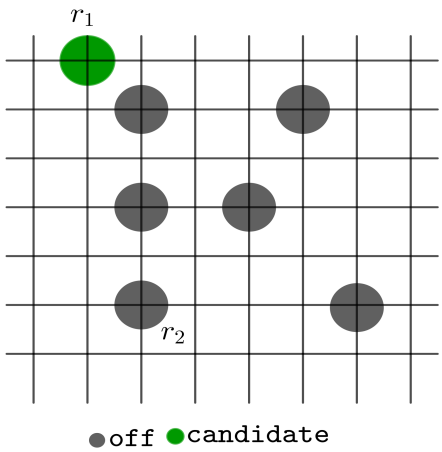


Figure 5.2: r_1 changes its colour to `candidate` and moves to $\mathcal{L}_1$ at time T_1 and r_2 wakes after time T_1.

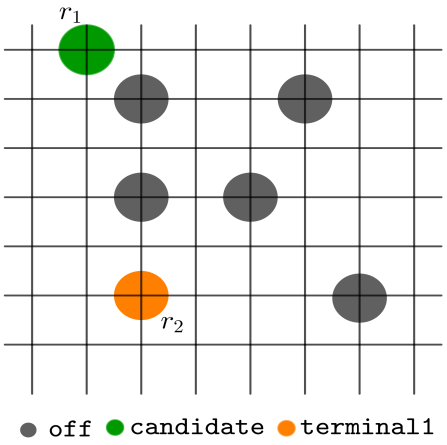


Figure 5.3: r_1 changed its colour to `candidate` and moves to $\mathcal{L}_1$ at time T_1 and r_2 changes its colour to `terminal1` at time $T_2 \geq T_1$.

Case-III: Let us consider the case where r_1 already changed its colour to `candidate` from `terminal1` and moved to $\mathcal{L}_1$ at a time $T_1 > 0$ and r_2 wakes before T_1 and changes its colour to `terminal1` at a time $T_2 < T_1$. Now, let r_2 wakes again at a time T_3.

Case-III(a): Now if $T_3 > T_1$, then even if r_1 wakes again between T_3 and T_1, it sees r_2 with colour `terminal1` on $\mathcal{R}_I(r_1)$. So, it does not change its colour to

`leader1`. Now r_2 at time T_3 wakes and sees r_1 with colour `candidate` on $\mathcal{L}_I(r_2)$ and so changes its colour to `off`. Now $\exists\ T_4$ such that r_1 wakes at $T_4 > T_3$ and sees it is singleton on $H_L^C(r_1)$ and have all robots on $\mathcal{R}_I(r_1)$ with colour `off`. So, r_1 changes its colour to `leader1`.

Case-III(b):

Now if $T_3 = T_1$, then both r_1 and r_2 changes its colour to `candidate` and moves to $\mathcal{L}_1$. In this case, there will be two robots with colour `candidate` on $\mathcal{L}_1$ (Figure 5.4).

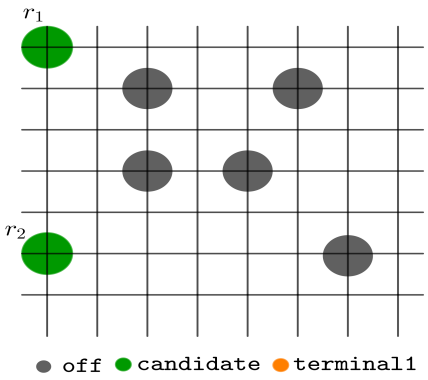


Figure 5.4: r_2 is with colour `terminal1` and r_1 changes its colour to `terminal1` and both r_1 and r_2 move to $\mathcal{L}_1$ changing their colour to `candidate at the same time.`

Case-III(c): Now when $T_3 < T_1$, it is similar as case-III(a).

Note that above all cases are exhaustive and in each case there is a time $T > 0$ such that in $\mathbb{C}(T)$, there is either one robot with colour `leader1` or two robots with colour `candidate` on $\mathcal{L}_1$. □

Lemma 5.1. *Any robot r with colour* `candidate` *or,* `call` *or,* `reached` *always can see all robots with colour* `off` *in $\mathcal{R}_I(r)$ (if exist) and vice versa.*

Proof. Let us consider that there is exactly one robot r and no other robot is on $\mathcal{L}_V(r)$. In this case, it is obvious that r can see all robots including the robots with colour `off` on $\mathcal{R}_I(r)$ and vice versa.

Now let us consider that there are at least two robots r and r' on $\mathcal{L}_V(r)$ where

colour of r and r' can be any one of `candidate`, `call` or, `reached` and r is above r'. Now there are two cases (Figure 5.5 and Figure 5.6).

Case-I: There are other empty vertical lines between $\mathcal{L}_V(r)$ and $\mathcal{R}_I(r)$. In this case, let us take the common tangent $line_1$ of all robots on $\mathcal{R}_I(r)$ which is parallel to the line $\mathcal{R}_I(r)$ and nearest to $\mathcal{L}_V(r)$ and similarly take the common tangent $line_2$ of the robots on line $\mathcal{L}_V(r)$ parallel to $\mathcal{L}_V(r)$ and nearest to $\mathcal{R}_I(r)$. Let us denote the points where $line_1$ touches the terminal robots on $\mathcal{R}_I(r)$ as p_1 and p_2 respectively (p_1 is above p_2) and the points where $line_2$ touches r' and r as p_3 and p_4 respectively. Now let us draw a line segment say $line_3 = \overline{p_4p_1}$ and $line_4 = \overline{p_3p_2}$. Observe that the area bounded by the lines $line_1, line_2, line_3$ and $line_4$ is a trapezoid which is a convex set containing no other robot (Figure 5.5). Let r_1 be any robot with colour `off` on $\mathcal{R}_I(r)$. Let $line_1$ touches the robot r_1 at a point say P. Then the line segments $\overline{Pp_3}$ and $\overline{Pp_4}$ contains no robot on them. So, each of r and r' can see r_1. Thus r and r' can see all robots with colour `off` on $\mathcal{R}_I(r)$ and all robots with colour `off` can see both of r and r'.

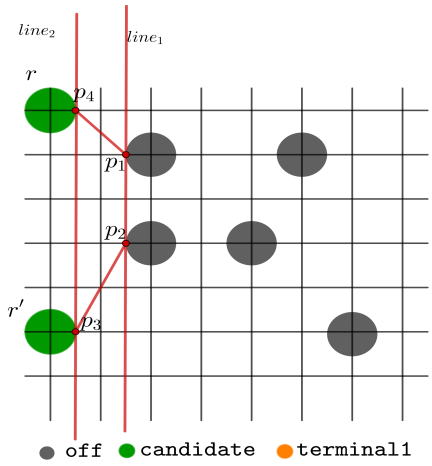


Figure 5.5: Area bounded by the quadrilateral $\overline{p_1p_2p_3p_4}$ is convex.

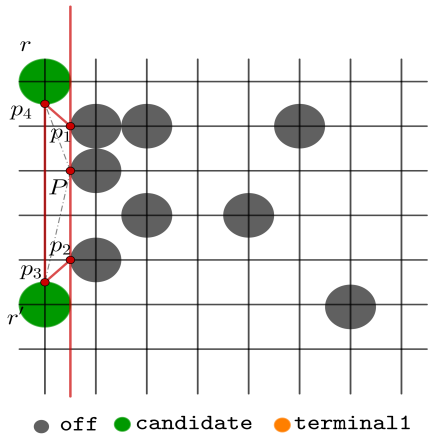


Figure 5.6: Area bounded by the quadrilateral $\overline{p_1p_2p_3p_4}$ is convex.

Case-II: Next let there is no other vertical line between $\mathcal{L}_V(r)$ and $\mathcal{R}_I(r)$. Note that all robots with colour `off` must be between the lines $\mathcal{L}_H(r)$ and $\mathcal{L}_H(r')$. In this scenario, let us draw the common tangent $line_1$ of the robots on $\mathcal{R}_I(r)$ which is parallel to $\mathcal{R}_I(r)$ and nearest to $\mathcal{L}_V(r)$. Let us denote the points where $line_1$

touches the terminal robots on $\mathcal{R}_I(r)$ as p_1 and p_2 (p_1 is above p_2). Let us now denote the points where boundary of r and r' intersect the line $\mathcal{L}_V(r)$ which is nearest to r' and r respectively as p_4 and p_3. Let us call the line segment $\overline{p_3p_4}$ as $line_2$, $\overline{p_4p_1}$ as $line_3$ and $\overline{p_3p_2}$ as $line_4$. Then the area bounded by these four lines is convex and there is no robot inside this area. 0For any robot r_1 on $\mathcal{R}_I(r)$ with colour `off`, let us denote the point where $line_1$ touches r_1 as P. Then both the line segment $\overline{Pp_3}$ and $\overline{Pp_4}$ do not contain any other robot (Figure 5.6). So, both r and r' can see r_1 and similarly r_1 sees both r and r'. Thus r and r' can see all robots with colour `off` on $\mathcal{R}_I(r)$ and all robots with colour `off` can see both of r and r'.

So, we can conclude the lemma. □

Lemma 5.2. *If r_1 and r_2 be two robots with colour `call` or `reached` or `candidate` on the same vertical line, then any terminal robot r with colour `moving1` on $\mathcal{R}_I(r_1)$ $(= \mathcal{R}_I(r_2))$ always can see at least one of r_1 and r_2.*

Proof. Without loss of generality, let us assume that r_1 is above r_2 and r is above K (i.e the horizontal line which is equidistant from both $\mathcal{L}_H(r_1)$ and $\mathcal{L}_H(r_2)$). Also, let there is no other vertical line between $\mathcal{L}_I(r)$ and $\mathcal{L}_V(r)$, otherwise with the same argument as Lemma 5.1 we can say that r can see both r_1 and r_2. Now there are three cases.

Case-I: r is below $\mathcal{L}_H(r_1)$. In this case, by similar argument in Case-II of Lemma 5.1, we can conclude that r can see both r_1 and r_2.

Case-II: r is on $\mathcal{L}_H(r_1)$. Let us draw the tangents of r, $line_1$ parallel to $\mathcal{L}_V(r)$ and nearest to $\mathcal{L}_I(r)$ and tangent of r_1, $line_2$ parallel to $\mathcal{L}_V(r_1)$ and nearest to $\mathcal{R}_I(r_1)$. Now let $line_1$ touches r at point p_1 and $line_2$ touches r_1 at point p_2 (Figure 5.7). Since $\overline{p_1p_2}$ does not contain any other robot, r can see r_1. Note that $line_1$ and $line_2$ can be same if the robots are of radius $\frac{1}{2}$. Now let $line_1$ touches both r and r_1 at a point p. Hence r can see r_1.

Case-III: r is above $\mathcal{L}_H(r_1)$. In this case, we claim that if there is any other robot with colour `moving1` on $\mathcal{L}_V(r)$, it must be below $\mathcal{L}_H(r_1)$. If possible let, there is another robot r' on $\mathcal{L}_V(r)$ which is above $\mathcal{L}_H(r_1)$ but below $\mathcal{L}_H(r)$ with

colour `moving1`. Since a robot turns its colour to `moving1` from `off`, there exists a time T when r' had colour `off`. So, in $\mathbb{C}(T)$, r' must be located below $\mathcal{L}_H(r_1)$. Now, r is already located on $\mathcal{L}_V(r')$ and above r'. So, if r' is terminal, it moves opposite to r and never reaches above $\mathcal{L}_H(r_1)$. And if r' is not terminal, then it never moves until r moves left. So, if r is above $\mathcal{L}_H(r_1)$ with colour `moving1` and is terminal, then there is no other robot on the grid points on $\mathcal{L}_V(r)$ between $\mathcal{L}_H(r)$ and $\mathcal{L}_H(r_1)$. Let the tangent of r which is parallel to $\mathcal{L}_V(r)$ and nearest to $\mathcal{L}_I(r)$ touches r at point p_1 and intersects $\mathcal{L}_H(r_1)$ at p_3. Also, boundary of r_1 touches the line $\mathcal{L}_H(r_1)$ at a point nearest to $\mathcal{L}_V(r)$ (say p_2) (Figure 5.8). Since p_1, p_2 and p_3 form a triangle and the area bounded by the triangle is a convex set containing no other robot, r can see r_1.

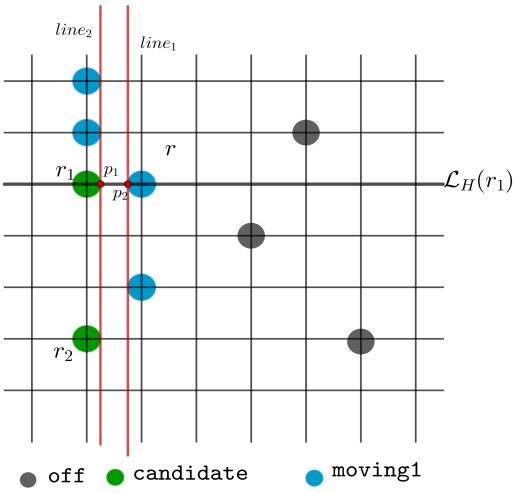


Figure 5.7: r is on $\mathcal{L}_H(r_1)$.

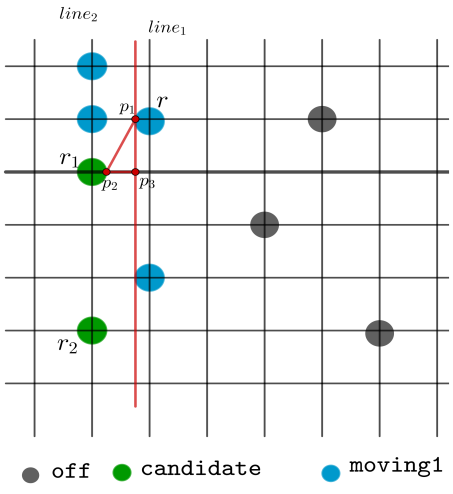


Figure 5.8: r is above $\mathcal{L}_H(r_1)$.

□

Lemma 5.3. *A robot changes its colour to* `leader1` *only from light* `candidate` *or* `off`.

Proof. From Algorithm 10, it follows directly that a robot can change its colour to `leader1` only if it was either with colour `candidate` or with colour `off`. □

Lemma 5.4. *A robot with light* `leader1` *always has empty grid point in its left.*

Proof. If at some tome $T > 0$, there is only one robot say r, on $\mathcal{L}_1$ with colour `candidate` and no robot with colour other than `off` on $\mathcal{R}_I(r)$, then r changes its colour to `leader1`. Observe that since r is on $\mathcal{L}_1$, it will have its left grid point empty.

Now, consider there are two robots r and r' with colour `candidate` on $\mathcal{L}_V(r)$ (i.e. $\mathcal{L}_V(r')$). Now by Lemma 5.3, it is evident that a robot with colour `candidate` or `off` can only change its colour to `leader1`. So, let us consider these cases (Figure 5.9 and Figure 5.10).

Case-I: Let a robot r_1 with colour `off` changes its colour to `leader1`. That implies only r and r' is on $\mathcal{L}_I(r_1)$ having colour `call` and r_1 is on $K \cap \mathcal{R}_I(r)$. Note that if r and r' are adjacent on $\mathcal{L}_V(r)$, then K can not be a horizontal line of the grid $\mathcal{G}$. So, r and r' are not adjacent on $\mathcal{L}_V(r)$ (Figure 5.9). Now note that even if there are robots other than r and r' on $\mathcal{L}_V(r)$ or, $\mathcal{L}_I(r)$, they are not on or between the line $\mathcal{L}_H(r)$ and $\mathcal{L}_H(r')$. So, $\mathcal{L}_H(r_1)$ lies between $\mathcal{L}_H(r)$ and $\mathcal{L}_H(r')$ and r_1 is on $\mathcal{R}_I(r)$. So, we can say that $\mathcal{L}_H(r_1) \cap H_L^O(r_1)$ is always empty. As r_1 moves only on left until it becomes singleton on $\mathcal{L}_1$ and has $l_{next}(r_1)$ empty, r_1, the robot with light `leader1` always has its left grid point empty.

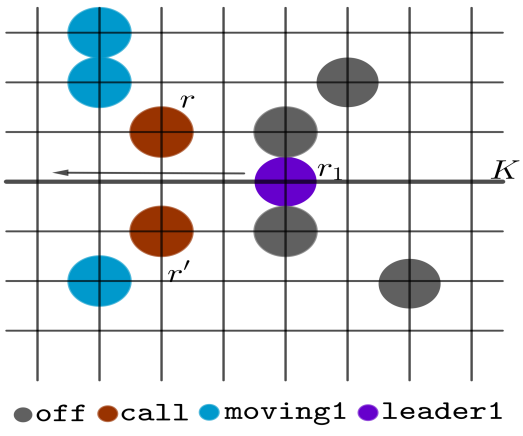


Figure 5.9: r_1 has colour `leader1` and has $\mathcal{L}_H(r_1) \cap H_L^O(r_1)$ empty.

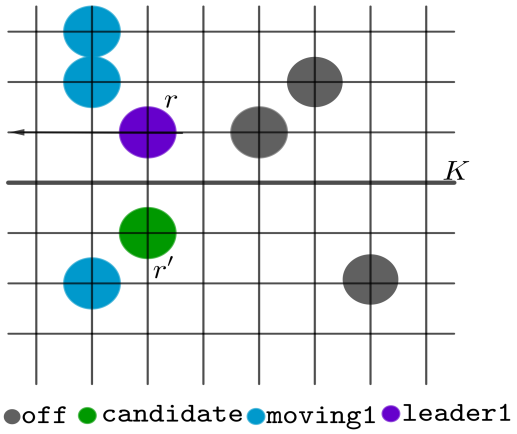


Figure 5.10: r is on $\mathcal{L}_2$ with colour `leader1` and has $\mathcal{L}_H(r) \cap H_L^O(r)$ empty.

Case-II: Without loss of generality, let r be the robot with colour `candidate` that changes the colour to `leader1`. Note that, either r is on $\mathcal{L}_1$ or it is on $\mathcal{L}_2$. If

r is on $\mathcal{L}_1$ then it finds its left grid point is empty and moves to left and become singleton on $\mathcal{L}_1$. So, left grid point of r is empty . Now if r was on $\mathcal{L}_2$ (Figure 5.10). Then $\mathcal{L}_1 \cap \mathcal{L}_H(r)$ is empty as there are no robots between the line $\mathcal{L}_H(r)$ and $\mathcal{L}_H(r')$ on $\mathcal{L}_I(r)$. So, r can move left and become singleton on $\mathcal{L}_1$. Hence r always has its left grid point empty.

□

Lemma 5.5. *If a robot r with colour* `call` *does not see another robot with colour* `leader1` *on $\mathcal{R}_I(r)$, then there is a time T when $\mathcal{L}_V(r)$ will always have a robot with colour* `moving1` *and two robots with colour* `call` *in $\mathbb{C}(T)$.*

Proof. r is a robot with colour `call` on $\mathcal{L}_V(r)$. This implies $\mathcal{R}_I(r)$ is symmetric with respect to K, where K is known because there is another robot say r' on $\mathcal{L}_V(r)$ with colour `call` or `candidate`. Note that if r' has colour `candidate`, it changes the colour to `call` after a finite time. In this situation, if there is a robot say r_1 on $K \cap \mathcal{R}_I(r)$, then r_1 changes its colour to `leader1` from `off`. And also, r sees r_1 on $\mathcal{R}_I(r)$. Since it is assumed that r is not seeing any robot with colour `leader1` on $\mathcal{R}_I(r)$, it is evident that there is no robot on $K \cap \mathcal{R}_I(r)$. In this scenario, the robots on $\mathcal{R}_I(r)$ see that there are exactly two robots r and r' with colour `call` on left immediate vertical line. So, the robots on $\mathcal{R}_I(r)$, which are closest to K change their colours to `moving1` upon activation and all the robots who can see a robot with colour `moving1` on their vertical line eventually change their colours to `moving1`. Observe that in this way, after a finite time there will be at least one robot on $\mathcal{R}_I(r)$ which has colour `moving1` and also will be terminal on $\mathcal{R}_I(r)$. Let r_2 be that robot. Now r_2 will move vertically in one fixed direction until at least one of $H^C_U(r) \cap \mathcal{L}_I(r)$ and $H^C_B(r) \cap \mathcal{L}_I(r)$ has no other robot and then it moves left to $\mathcal{L}_V(r)$ (Figure 5.11). Also, note that r and r' do not change their colours until r_2 reaches $\mathcal{L}_V(r)$. So, after a finite time say T, there will be a robot r_2 with colour `moving1` and two robots r and r' with colour `call` on $\mathcal{L}_V(r)$ in $\mathbb{C}(T)$ (Figure 5.12).

□

Lemma 5.6. *During movement of robots with colour* `moving1` *in Phase 1, no*

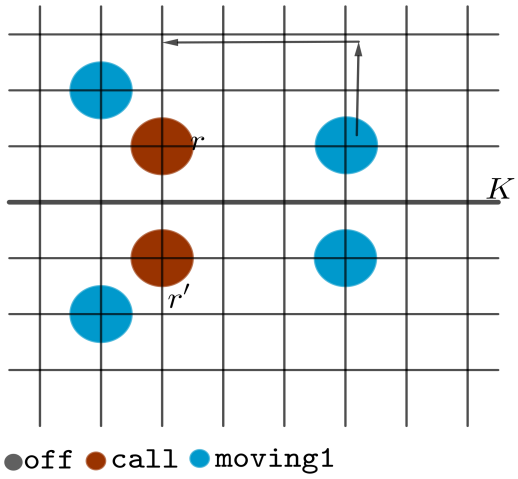


Figure 5.11: Terminal robot on $\mathcal{R}_I(r)$ see r with colour `call` and move according to the path shown by the arrow.

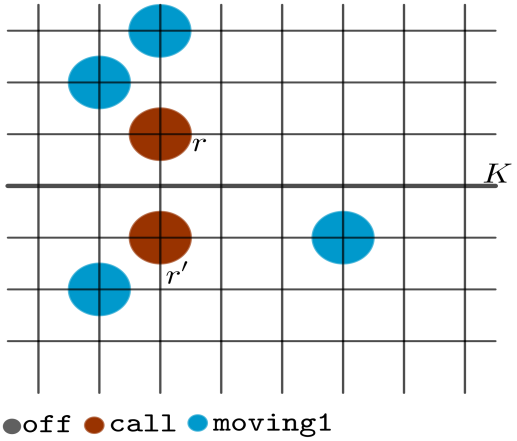


Figure 5.12: The robot with colour `moving1` reaches above r. In this moment, $\mathcal{L}_V(r)$ has two robots r and r' with colour `call` and one robot with colour `moving1`.

collision occurs.

Proof. A robot r with colour `moving1` can have two type of moves, horizontal to the left and vertical. r moves vertically on $\mathcal{L}_V(r)$ only when it sees at least one robot with colour `call` on $\mathcal{L}_I(r)$. Note that during vertical movement of r, no other robot on $\mathcal{L}_V(r)$ moves vertically in the same direction as r. This is because if another robot say r' moves on $\mathcal{L}_V(r)$, it must be terminal on $\mathcal{L}_V(r)$ and has colour `moving1`. But since r is already on $\mathcal{L}_V(r)$, r' moves opposite of r. So, as long as r moves vertically on $\mathcal{L}_V(r)$ no collision occurs. Note that r moves vertically in such a way such that at least one of $H^C_U(r) \cap \mathcal{L}_I(r)$ or $H^C_B(r) \cap \mathcal{L}_I(r)$ has no other robot and then it moves left towards $\mathcal{L}_I(r)$ (i.e the same vertical line were the robot with colour `call` is located). Now let there is a non-terminal robot r_1 which is nearest to r and below r on $\mathcal{L}_V(r)$ with colour `moving1`. Now observe that r_1 only moves when r reaches the vertical line of the robot with colour `call`. In this scenario, r_1 moves vertically in such a way such that it has either $H^C_U(r_1) \cap \mathcal{L}_I(r_1)$ or $H^C_B(r_1) \cap \mathcal{L}_I(r_1)$ has no other robot and then moves left to the empty grid point. So, during horizontal or vertical movement of robots with colour `moving1`, no collision occurs. Hence the result. □

Lemma 5.7. *If at time T, two robots r and r' have colour* `call` *on the same vertical line and there is no robot on $K \cap \mathcal{R}_I(r)$, then there exist $T' > T$ such that both r and r' are with colour* `reached` *at $\mathbb{C}(T')$.*

Proof. Let r and r' be two robots with colour `call` at time T on same vertical line $\mathcal{L}_V(r)$ (i.e. $\mathcal{L}_V(r')$). Then $\mathcal{R}_I(r)$ must be symmetric with respect to K. Also, there is no robot on $K \cap \mathcal{R}_I(r)$. So, no robot with colour `off` on $\mathcal{R}_I(r)$ changes its colour to `leader1`. Now in this scenario, the robots which are closest to K on $\mathcal{R}_I(r)$ change their colours to `moving1`. Note that a robot with colour `off` also can change its colour to `moving1` if it sees another robot with colour `moving1` on the same vertical line. Also, no robot with colour `moving1` moves unless it is terminal on the same vertical line. Hence we can say that at least all robots of above or below K on $\mathcal{R}_I(r)$ change their colours to `moving1`. Now by Algorithm 10, the terminal robots with colour `moving1` move to $\mathcal{L}_V(r)$. Then next robot becomes terminal and do the same. So, after a finite time say $T_1 > T$, all robots with colour `moving1` on $\mathcal{R}_I(r)$ move to $\mathcal{L}_V(r)$. In this moment, all robots of $\mathcal{R}_I(r)$ have colour `off`. Note that in this scenario, at least one of r or r' must see a robot with colour `moving1` on the same vertical line upon activation. Without loss of generality, let r sees a robot with light `moving1` on $\mathcal{L}_V(r)$ and all robots on $\mathcal{R}_I(r)$ have colour `off` (Figure 5.13). Then r changes its colour to `reached` at time say $T_2 \geq T_1 \geq T$ (Figure 5.14). Now when r' activates, it sees r with colour `reached` on $\mathcal{L}_V(r')$ and changes its colour to `reached` at a time $T_3 \geq T_2$ (here $T' = T_3$) (Figure 5.15). Now it may be possible due to asynchronous environment that after r changes its colour to `reached` at time T_2, a robot say r_1, on $\mathcal{R}_I(r)$ changes its colour to `moving1` . Then r' will not change its colour to `reached` now, even after seeing r with colour `reached` as all robots on $\mathcal{R}_I(r')$ now do not have colour `off`. Now when r_1 wakes again at a time say $T_4(\geq T_2)$, it sees r with colour `reached` on $\mathcal{L}_I(r)$ and changes its colour to `off`. Now when r' wakes again at some time $T' \geq T_4 \geq T_2 \geq T_1 > T$, it changes its colour to `reached`. Note that r does not change its colour from `reached` to `candidate` before r' wakes and changes its colour to `reached` as it will not see any other robot with colour `reached` or `candidate` on $\mathcal{L}_V(r)$ before r' wakes. So, we can conclude that $\exists$

$T' > T$ when both r and r' have colour `reached`.

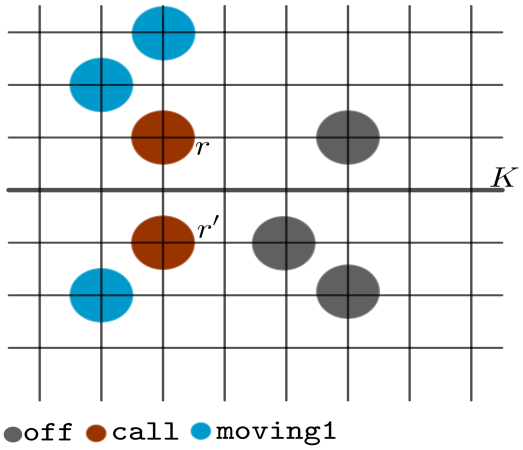


Figure 5.13: r sees a robot with colour `moving1` on $\mathcal{L}_V(r)$ and sees all robots on $\mathcal{R}_I(r)$ with colour `off`. r' does not see any robot with colour `moving1` or `reached` on $\mathcal{L}_V(r')$.

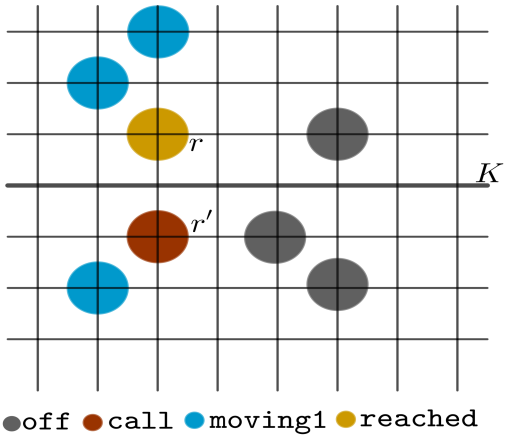


Figure 5.14: r changes its colour to `reached`.

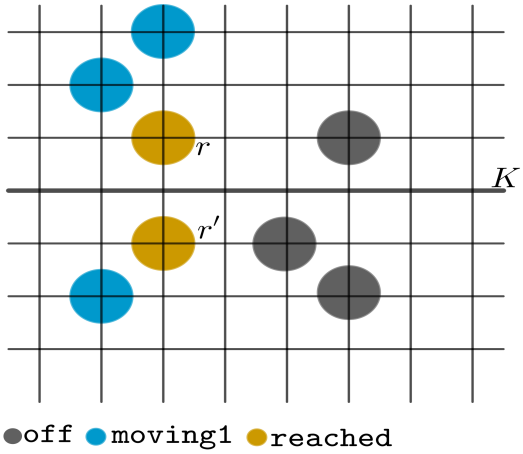


Figure 5.15: Now r' sees r with colour `reached` on $\mathcal{L}_V(r')$ and all robots on $\mathcal{R}_I(r')$ with colour `off`. So, r' changes its colour to `reached`.

□

Lemma 5.8. *If a robot r changed its colour to* ***`reached`*** *at some time $T > 0$, then $\exists\ T' \geq T$ such that all robot in $\mathcal{R}_I(r)$ in $\mathbb{C}(T')$ have colour* ***`off`****.*

Proof. During the look phase, r must have seen robots on $\mathcal{R}_I(r)$ have colour `off`. Now if no robot on $\mathcal{R}_I(r)$ change its colour to `moving1` in between the completion of look phase of r and time T, then $T' = T$. Now if a robot, say r_1 changes its colour to `moving1` in between completion of look phase of r and time T, then there exists $T' > T$ when r_1 sees a robot with colour `reached` on $\mathcal{L}_I(r_1)$ and so changes its colour to `off`. Note that before r_1 changes its colour to `off`, r does not change its colour as it sees r_1 with colour `moving1` on $\mathcal{R}_I(r)$. So, we can conclude $\exists\ T' \geq T$ such that all robots on $\mathcal{R}_I(r)$ have colour `off` in $\mathbb{C}(T')$. □

Lemma 5.9. *If at time T, a robot changes its colour to `leader1` from `off`, then $\mathbb{C}(T')$ has no robot with colour `candidate` or `terminal1`, where $T' \geq T$.*

Proof. If r changes its colour to `leader1` from `off` at some time T, then it must have seen exactly two robots say r_1 and r_2 with colour `call` on $\mathcal{L}_I(r)$ at a time T_1 where $T_1 < T$ (Figure 5.16). Note that a robot can only have colour `call` at some time T_2 if it had colour `candidate` at some time $T_3 < T_2$. Also, a robot can change its colour to `candidate` from `terminal1` only if it sees there is no other robot on its left open half. Also, a robot with colour `off` changes to colour `terminal1` only if its left open half empty, there is no robot with colour `leader1` on $\mathcal{R}_I(r_1)$ or on $\mathcal{L}_V(r_1)$ and it is terminal on $\mathcal{L}_V(r)$. Since during the whole execution of *Phase 1*, no other robot having colour `off` except r_1 and r_2 can see its left open half empty and find themselves to be terminal, no other robot except r_1 and r_2 can change their colours to `terminal1`. Now upon activation again at any time $T_4 > T$, both r_1 and r_2 sees r on $\mathcal{R}_I(r_1)$ with colour `leader1` and change their colours to `off` (Figure 5.17). Observe that after time T_4, r_1 and r_2 can never change their colour to `terminal1` and hence to `candidate` as they will see r with colour `leader1` on $\mathcal{R}_I(r_1)$ or on $\mathcal{L}_V(r_1)$ or r_1 and r_2 would have its left open half non-empty. So, we can conclude the lemma.

□

Lemma 5.10. *If at a time T, a robot r changed its colour to `leader1`, then there will be no robot with colour `reached` in $\mathbb{C}(T')$, where $T' \geq T$.*

Proof. Note that a robot can only change its colour to `reached` at a time T if $\exists$

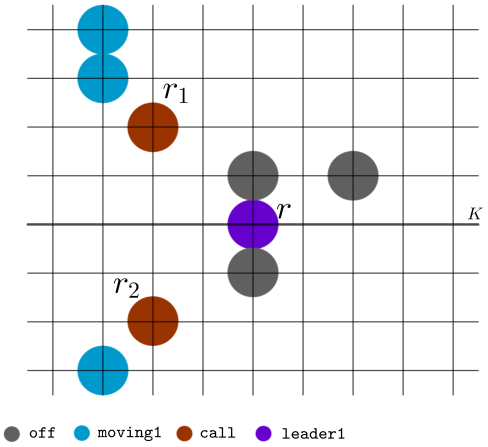


Figure 5.16: r_1 and r_2 with colour `call` both see r with colour `leader1` on $\mathcal{R}_I(r_1) \cap K$.

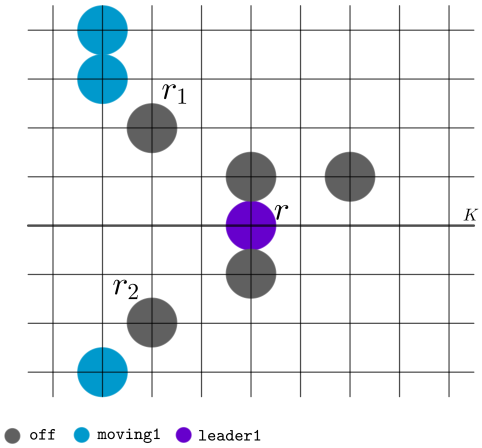


Figure 5.17: Both r_1 and r_2 change their colours to `off` after seeing r.

$T_1 < T$ such that the robot had colour `candidate` in $\mathbb{C}(T_1)$. Now if r changed its colour to `leader1` from `candidate`, then even if there is another robot say r' with colour `candidate` on $\mathcal{L}_V(r)$ (Figure 5.18), r' will change its colour to `off` upon first activation at a time $T_2 > T$. So, the configuration now has no robot with colour `candidate` or `reached` (as both r and r' with colour `candidate` who could have changed their colour to `reached` changed it to `leader1` and `off`) (Figure 5.19). Also, note that during the period between T and T_2, the configuration does not have colour `reached` as in this time r has colour `leader1` and r' has colour `candidate`. Also, no other robot with colour `off` will ever change its colour to `candidate` after time T_2 as a robot say r_1 with colour `off` or `terminal1` either sees r with colour `leader1` on $\mathcal{L}_V(r_1)$ or on $\mathcal{R}_I(r_1)$ or it has its left open half non-empty. And since a robot can only change its colour to `reached` when it had colour `candidate` before, there will be no robot with colour `reached` in $\mathbb{C}(T')$, where $(T' \geq T)$.

Now, if r has changed its colour to `leader1` from `off` at time T, then r must have seen two robots say, r_1 and r_2 on $\mathcal{L}_I(r)$ with colour `call` at some time $T_1 < T$. Now upon activation after time T, both r_1 and r_2 see r on $\mathcal{R}_I(r_1)$ and turn their colours to `off`. Now for r_1 and r_2 to ever have the colour `reached` again must have

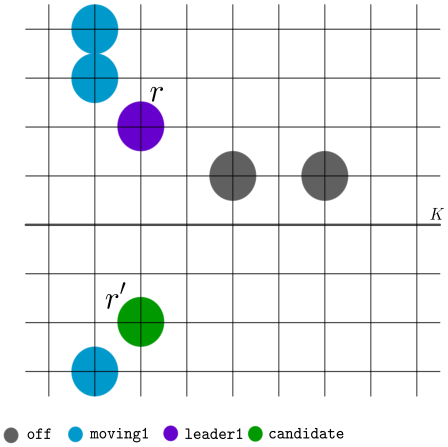


Figure 5.18: r' with colour `candidate` sees r with colour `leader1` on $\mathcal{L}_V(r')$.

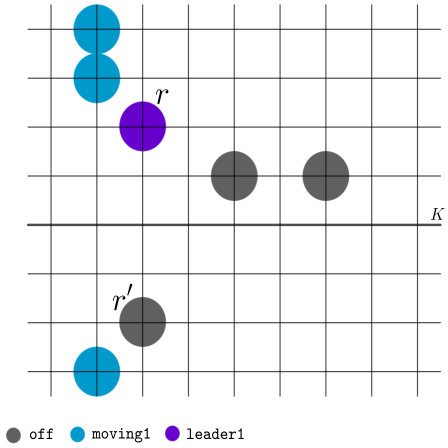


Figure 5.19: r' changes its colour to `off`. Now, no other robot changes colour to `terminal1` and hence to `candidate` and hence to `reached`.

colour `candidate` first. But by Lemma 5.9, after r changes its colour to `leader1`, the configuration can never have a robot with colour `candidate`. Hence, $\mathbb{C}(T')$ $(T' \geq T)$ has no robot with colour `reached` if r changed its colour to `leader1` at time T. □

Lemma 5.11. *At any time T, there can be at most one robot with colour `leader1` and at most one robot with colour `leader` in the configuration.*

Proof. Note that by Lemma 5.3, a robot can change its colour to `leader1` only from the colour `off` or `candidate`. Let us consider the following cases:

Case-I: Consider the case where a robot changes its colour to `leader1` from the colour `candidate`. Now from Theorem 5.2, for any initial configuration $\mathbb{C}(0)$, there exist a time T such that $\mathbb{C}(T)$ has either one robot with colour `leader1` who has changed its colour to `leader1` from `candidate` or two robots with colour `candidate` on the same vertical line.

Case-I(a): Let r is a robot with colour `leader1` in $\mathbb{C}(T)$ who has changed its colour from `candidate`. We claim that in $\mathbb{C}(T')$ where $T' \geq T$, there is no other robot who changes its colour to `leader1`. For this, we first show that no other robot with colour `terminal1` ever change their colour to `candidate`.

This is because no other robot with colour `terminal1` will find its left open half empty (as the robot with colour `leader1` is there) (Figure 5.20). So $\mathbb{C}(T')$, where $T' \geq T$, will not have any robot with colour `candidate` who can change further to `leader1`. Also observe that after at $T' \geq T$, no other robot with colour `off` changes its colour to `leader1` as they will not see any robot with colour `call` on their left immediate occupied vertical line. This is also for the reason that $\mathbb{C}(T')$ where $T' \geq T$ will not have any other robot with colour `candidate` who can change its colour to `call` further. So, there will be exactly one robot with colour `leader1` which eventually changes its colour to `leader`.

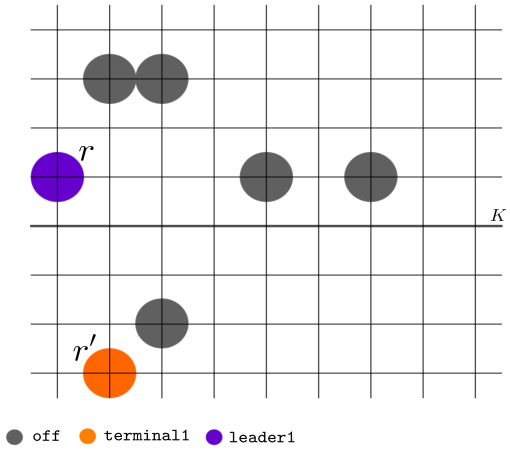


Figure 5.20: r' with colour `terminal1` sees r with colour `leader1` on $\mathcal{L}_I(r')$. So, r' does not change its colour as it does not have its left open half empty.

Case-I(b): Let us now assume the case where there are two robots r_1 and r_2 with colour `candidate` both on the same vertical line $\mathcal{L}_V(r_1)$(i.e. $\mathcal{L}_V(r_2)$). In this case, we will first show that both r_1 and r_2 can not change their colour to `leader1`. Then we will show if one of r_1 or r_2 changes its colours to `leader1`, then no other robot with colour `off` changes its colour to `leader1`.

In this case, r_1 and r_2 check the symmetry of the line $\mathcal{R}_I(r_1)$(i.e $\mathcal{R}_I(r_2)$). If $\mathcal{R}_I(r)$ is asymmetric, then the robot (r_1 or, r_2) whichever is on the dominant half changes its colour to `leader1`. Without loss of generality. let at some time T_1, r_1 changes its colour to `leader1` from `candidate`. Then r_2 must have colour `candidate` in $\mathbb{C}(T_1)$ (Figure 5.21). Now when r_2 wakes again at a time say $T_2 > T_1$, it sees r_1

with colour `leader1` on $\mathcal{L}_V(r_2)$ and changes its colour to `off` (Figure 5.22). Note that between time T_1 and T_2 even if r_1 awakes again, it does not move as it sees r_2 with colour `candidate` on $\mathcal{L}_V(r_1)$. Now even if r_2 is terminal with colour `off` and has left open half empty, it would not change its colour to `terminal1` and then to `candidate` again as it sees r_1 with colour `leader1` on the same vertical line. So, between two robots with colour `candidate` only one can change its colour to `leader1`.

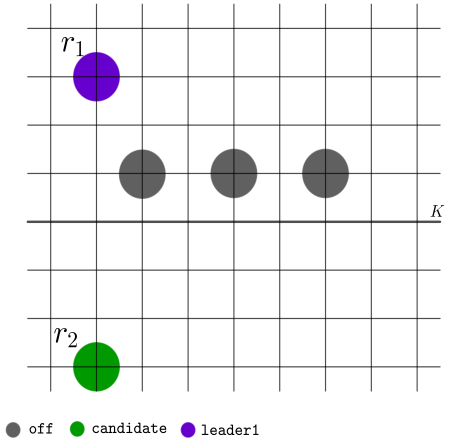


Figure 5.21: r_2 with colour `candidate` sees r with colour `leader1` on $\mathcal{L}_V(r_2)$. No robot with colour `call` in the configuration.

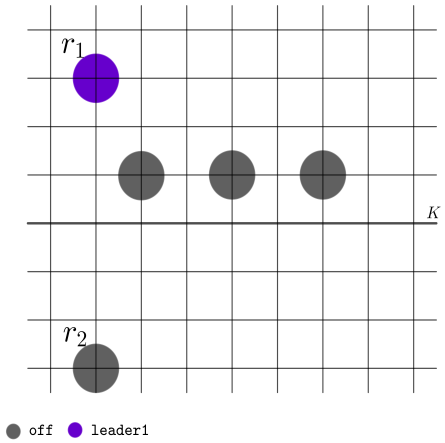


Figure 5.22: r_2 changes its colour to `off`. No robot with colour `call` in the configuration and no other robot with colour `off` changes colour to `terminal1` and hence to `candidate` and hence to `leader1` or `call` .

Now a robot say r with colour `off` can never change its colour to `leader1` as it would not see exactly two robots with colour `call` on $\mathcal{L}_I(r)$. This is because a robot can only change its colour to `call` from colour `candidate` and no other robot with colour `off` will ever change its colour to `terminal1` and then to `candidate` as in this case even if a robot with colour `off` has its left open half empty and is terminal on its vertical line, it will see r_1 with colour `leader1` on the same vertical line (Figure 5.22). So, it would not change its colour to `terminal1`. So, if a robot changes its colour to `leader1` from `candidate`, then no other robot will change its colour to `leader1`.

Case-II: Next we show that if a robot say r has changed its colour to `leader1` from `off`, then no other robot with colour `candidate` or `off` ever changes its colour to `leader1`.

Let r changed its colour to `leader1` from colour `off` at some time T_2. This implies r must have seen exactly two robots say r_1 and r_2 with colour `call` on $\mathcal{L}_I(r)$ and r is on $K \cap \mathcal{L}_V(r)$. Also $\mathbb{C}(T_2)$ has no robot with colour `candidate` (Figure 5.23). We will now show that no robot will ever change its colour to `candidate` again. Now when r_1 and r_2 wake again (lets say at time $T_2' > T_2$), it sees r with colour `leader1` on $\mathcal{R}_I(r_1)$ and so change their colours to `off` (Figure 5.24).

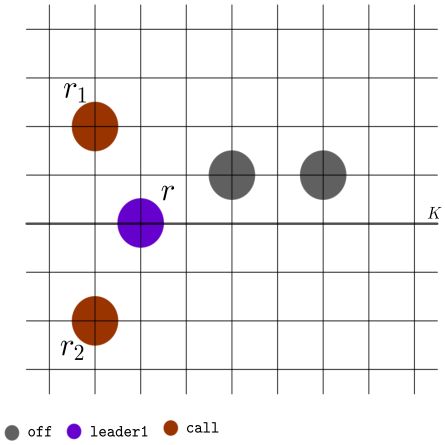


Figure 5.23: r_1 and r_2 with colour `call` see r with colour `leader1` on $\mathcal{R}_I(r_1)$. No robot with colour `candidate` in the configuration.

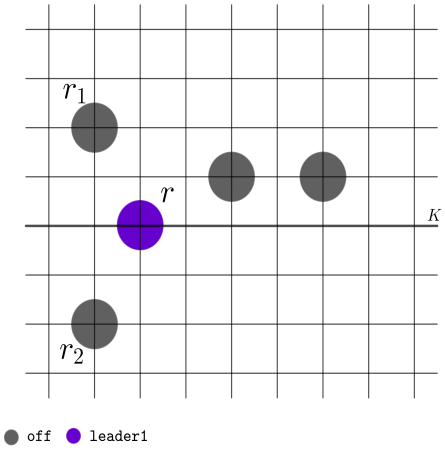


Figure 5.24: r_1 and r_2 change their colour to `off`. No robot with colour `reached` or `terminal1` or `candidate` or `call` in the configuration and no other robot with colour `off` changes colour to `terminal1` and hence to `candidate` and hence to `leader1` or `call` or directly to `leader1`.

Observe that, in this scenario there is no robot with colour `reached` and `terminal1` in the configuration $\mathbb{C}(T_2')$ and no robot with colour `off` will ever change its colour to `terminal1` and then to `candidate` eventually. This is because even if a robot with colour `off` finds its left open half empty and it is terminal on its vertical line, it sees r with colour `leader1` on its right immediate vertical line or on its

own vertical line. So after time T_2', the configuration has no robot with colour `candidate`, so no robot with colour `call` or `reached`. Thus if r changed its colour to `leader1`, no other robot can change its colour to `leader1` from `candidate`.

Again this scenario, all robots with colour `off` are either on $H_R^C(r)$ or on $\mathcal{L}_I(r)$. Note that all robots with colour `off` that are on $\mathcal{L}_I(r)$, never change their colours as they either see r on their right immediate vertical line or on the same vertical line or they find their left open half is non-empty. Similarly, the robots with colour `off` on $H_R^C(r)$ never change their colours again as they find their left open half non-empty, never see exactly two robots with colour `call` on their left immediate vertical line and never sees a robot with colour `moving1` on their same vertical line. So, we have proved if a robot has changed its colour to `leader1` from colour `off`, no other robot ever changes its colour to `leader1` again.

So, from all the cases, it is evident that in *Phase 1* any configuration can have at most one robot with colour `leader1` and since a robot changes its colour to `leader` from `leader1` only, there can be at most one robot with colour `leader` in any configuration during *Phase 1*.

□

Lemma 5.12. *If a robot r changes its colour to `leader1` at a time T and no robot with colour `terminal1` changes its colour to `candidate` at T' where $T' \geq T$, then robots on $\mathcal{R}_I(r)$ in $\mathbb{C}(T)$ never move in Phase 1.*

Proof. Let r changes its colour to `leader1` from colour `off` at a time T. Observe that in this case, r must have seen two robots say r_1 and r_2 with colour `call` at some time $T_1 < T$ and it is on $K \cap \mathcal{R}_I(r_1)$. Note that in this case, all robots on $\mathcal{R}_I(r)$ have colour `off`, so they do not move. Now upon activation after time T, both r_1 and r_2 change their colours to `off` and never change their colours again as they see r with colour `leader1` on $\mathcal{L}_V(r_1)$ or on $\mathcal{R}_I(r_1)$ or other robots on their left open half throughout completion of *Phase 1*. So, robots on $\mathcal{R}_I(r)$(at T) never see any robot with colour `call` and also, they do not see their left open half empty after time T . Thus robots on $\mathcal{R}_I(r)$ at time T never change their colours and never move until completion of *Phase 1*.

Now if r changes its colour from `candidate` to `leader1` at time T and no robot with colour `terminal1` changes its colour to `candidate` at some time T' where $T' \geq T$, then all robots which are on $\mathcal{R}_I(r)$ in $\mathbb{C}(T)$ can have colour either `moving1` or `off` or, `terminal1`. Note that the robots with colour `moving1` will not move as it does not see any robot with colour `call` on its left immediate vertical line or, robot with colour `reached` on its same vertical line and on its left immediate vertical line. Similarly robots with colour `off` on $\mathcal{R}_I(r)$ does not change its colour if it wakes after T as it finds out that it has its left open half non-empty and there is no robot with colour `call` on its left immediate vertical line. Note that a robot with colour `terminal1` may change its colour to `off` but after that this robot with colour `off` will not move by similar argument above. So, no robot on $\mathcal{R}_I(r)$ ever moves after time T until *Phase 1* is complete. □

Lemma 5.13. *If a robot r changes its colour to `leader1` from `candidate` at some time T and another robot r' changes its colour to `candidate` at a time T' where $T' \geq T$, then no collision occurs even if both r and r' move.*

Proof. Let r changes its colour to `leader1` from `candidate` at a time T and r' changes its colour to `candidate` from colour `terminal1` at a time $T' \geq T$. Note that at time T, r must be singleton on $\mathcal{L}_V(r)$. This implies there is a time $T_1 < T$ when r had colour `off` and was terminal on $\mathcal{L}_1$ in $\mathbb{C}(T_1)$. Now when r wakes at a time say T_2, where $T_1 \leq T_2 < T$, it changes its colour to `terminal1` and there exist a time $T_3 > T_2 \geq T_1$ and $T_3 < T$ such that r changes its colour to `candidate` and moves left and become singleton on $\mathcal{L}_V(r)$. We claim that r' changes its colour to `candidate` only if r' was on $\mathcal{L}_V(r)$ in $\mathbb{C}(T_2)$ and it was also terminal on $\mathcal{L}_V(r)$ (i.e $\mathcal{L}_V(r')$) as otherwise r' can not see its left open half empty. Now, let r' wakes before time T_2 and decide to changes its colour to `candidate` but it changes its colour at a time $T' \geq T$ and has a pending move. Then observe that now r and r' are on two different vertical lines and r' has a pending move to the left. So, if there are other vertical lines in between $\mathcal{L}_V(r)$ and $\mathcal{L}_V(r')$, then even if both of them move, no collision occurs as r can only move either vertically or on left and they are on different horizontal line. So, let us consider r' is on $l_{next}(r)$. Then r can not move vertically as $l_{next}(r)$ is non-empty. Hence, both

r and r' move left and no collision occurs as r and r' are on different horizontal lines. □

Now, from the above lemmas and the discussions, we can conclude the following theorem.

Theorem 5.3. *For any initial configuration* $\mathbb{C}(0)$*, there exists a* $T > 0$ *such that* $\mathbb{C}(T)$ *has exactly one robot with colour* `leader` *and it's left closed half and one of upper and bottom closed half have no other robots.*

5.2.2 Phase 2

In this phase, initially the configuration is a leader configuration and the robots form the target pattern embedded on the grid using the location of the leader as an agreement to the origin of a global coordinate system.

After completion of *Phase 1*, the configuration has exactly one robot r_0 with colour `leader` such that r_0 is singleton on $H_L^C(r_0)$ and also singleton on $\mathcal{L}_H(r_0)$ and there is no other robot on either below or above $\mathcal{L}_H(r_0)$. Note that in this configuration, all the robots who can see r_0 can agree on a global coordinate. Let r_1 be a robot which can see r_0. Then it assumes the position of r_0 as the coordinate $(0, -1)$. Now since all robots agree on the direction and orientation of the x-axis (i.e the horizontal lines), r_1 can think of the horizontal line let's say $\mathcal{H}$ which is just above r_0 as the x-axis where right half of the line of $\mathcal{L}_V(r_0)$ correspond to the positive direction of $x-$axis. Now r_1 agrees on the the vertical line $\mathcal{L}_V(r_0)$ as y-axis. Note that r_1 can also know the orientation of y-axis by assuming its own $y - coordinate$ to be greater or equals to the $y - coordinate$ of r_0 (i.e if $H_U^C(r_0)$ has robots, then the direction of $\mathcal{L}_V(r_0)$ from r_0 towards $H_U^C(r_0) \cap \mathcal{L}_V(r_0)$ is the direction of positive $y - axis$ and similar for the case if $H_B^C(r_0)$ has robots) (Figure 5.25). In this phase, robots first form a line and then from that line move to their corresponding target positions which are embedded in the grid assuming the global coordinate which has been agreed upon by robots after seeing r_0. Thus the pattern formation is done. We provided a detailed description of the algorithm for *Phase 2*.

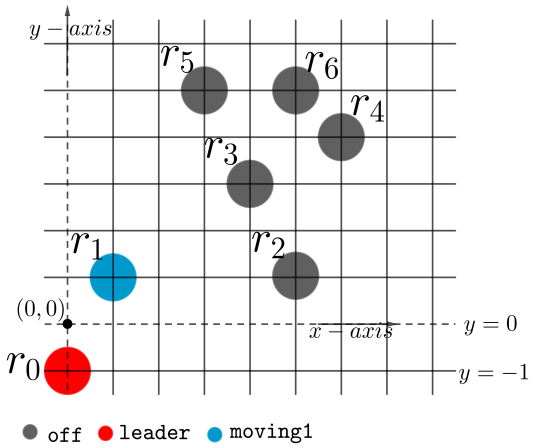


Figure 5.25: r_0 with colour `leader` is at $(0,-1)$. Any robot that sees r_0 can agree on a global coordinate system as shown in this diagram.

5.2.2.1 Line Formation

In the beginning of *Phase 2*, if a robot r sees r_0 with colour `leader1`, it agrees on a global coordinate as mentioned above and find that r_0 is on $H_B^O(r)$. Now, if r sees there is no robot in $H_B^O(r) \cap H_U^O(r_0)$ (i.e there is no horizontal line containing any robot between $\mathcal{L}_H(r_0)$ and $\mathcal{L}_H(r)$) and r is leftmost on $\mathcal{L}_H(r)$ and also if it finds there are i other robots on $(1,-1),(2,-1),\ldots,(i,-1)$ and no robot with colour `done`, then it changes its colour to `off` (if r has colour `off`, it would not change the colour) and moves to $(i+1,-1)$ by a method GOTOLINE(). The method GOTOLINE() is described as follows. In this method, if r is above the horizontal line where $y=0$, then it moves vertically downwards until it reaches the line where $y=0$. Note that no collision occurs during this vertical movement as there are no robots between $\mathcal{L}_H(r)$ and $\mathcal{L}_H(r_0)$ and no other robot will move until it reaches at $(i+1,-1)$. This is because other robots even if gets activated before r reaches $(i+1,-1)$, sees r between its horizontal line and $\mathcal{L}_H(r_0)$. Now, when r is at the horizontal line where $y=0$, it moves horizontally to the position $(i+1,0)$. Note that the horizontal line $y=0$ may not be initially empty at the beginning of *Phase 2*. Also, initially all robots on line $y=0$ have $x-coordinate>0$. Now let at some time T, the robot r which is leftmost on the line $y=0$ sees robots on $(1,-1),(2,-1),\ldots(i,-1)$. Now if there is no other robot except r on line $y=0$

and r is not on $(i+1,0)$, then r can simply move horizontally without collision to reach $(i+1,0)$. Note that during this movement, no other robot from above moves as they see r in $H_B^O(r) \cap H_U^O(r_0)$. Now if there are other robots on the line $y = 0$ other than r, then this implies r has $x - coordinate > i$. This is because the i robots say $r_1, r_2, \ldots, r_i$ must have reached their current position at $(1,-1),(2,-1),\ldots(i,-1)$ from the line $y = 0$ and all robots on line $y = 0$ have $x - coordinate > 0$. Now since r has $x - coordinate > i$, r will move to its left until it reaches $(i+1,0)$. Note that during this movement, no collision occurs as r is leftmost robot on $\mathcal{L}_H(r)$ and no other robot on $\mathcal{L}_H(r)$ moves as they are not leftmost on $\mathcal{L}_H(r)$. Next after r reaches the position $(i+1,0)$, it moves vertically downward to $(i+1,-1)$ (Figure 5.26). So after a finite time, all robots will reach on the line $y = -1$ in such a way that there is no empty grid point between two robots on the line $y = -1$ (Figure 5.27). From the above discussion, we have the following Lemmas 5.14 and 5.15.

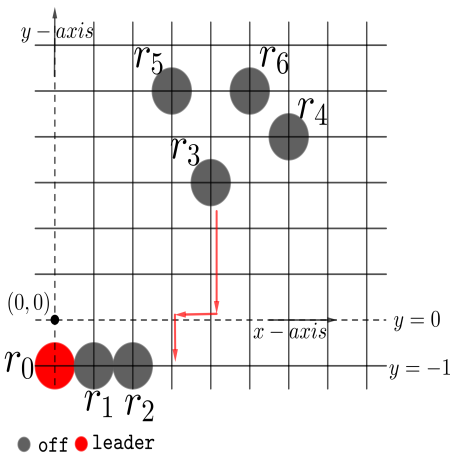


Figure 5.26: Movement of r_3 to $\mathcal{L}_H(r_0)$ by executing the method GoToLine().

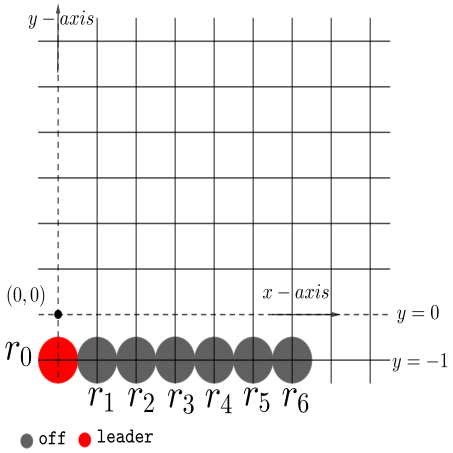


Figure 5.27: All robots formed a line on $\mathcal{L}_V(r_0)$. There is no empty grid point between any two robots on the line.

Algorithm 14: ApfFatGrid: Phase 2

```
1  r ← myself
2  r_0 ← the robot with light leader
3  if r.light = moving1 or candidate or terminal1 then
4  |  execute LineWithLeader()
5  else if r.light = off then
6  |  if (r_0 ∈ H_B^O(r)) and (r is leftmost on L_H(r)) and (there is no robot in
   |    H_B^O(r) ∩ H_U^O(r_0)) then
7  |  |  if there are no robots on L_H(r_0) other than r_0 then
8  |  |  |  if there is a robot with light done then
9  |  |  |  |  if r is at t_{n-2} then
10 |  |  |  |  |  r.light ← done
11 |  |  |  |  else
12 |  |  |  |  |  move to an empty grid point towards t_{n-2} by GotoTarget
13 |  |  |  else
14 |  |  |  |  move to (1, −1) by GotoLine
15 |  |  else if there are i robots on L_H(r_0) other than r_0 at (1, −1), ..., (i, −1) then
16 |  |  |  move to a empty grid point towards (i + 1, −1) by GotoLine
17 |  |  else if there are i robots on L_H(r_0) other than r_0 at
   |  |  |  (n − i, −1), ..., (n − 1, −1) then
18 |  |  |  if r is at t_{n-i-2} then
19 |  |  |  |  r.light ← done
20 |  |  |  else
21 |  |  |  |  move to an empty grid point towards t_{n-i-2} by GotoTarget
22 |  else if r_0 ∈ L_H(r) and H_U^O(r) has no robots with light off then
23 |  |  if r is at (i, −1) then
24 |  |  |  move to (i, 0)
25 else if r.light = leader then
26 |  if all robots, which are visible to r, have light done then
27 |  |  if r is at t_{n-1} then
28 |  |  |  r.light ← done
29 |  |  else
30 |  |  |  move to an empty grid point towards t_{n-1} by LeaderMove
```

Lemma 5.14. *During movement of a robot r, that is executing the method* GotoLine() *in Phase 2, r does not collide with any other robot in the configuration.*

Lemma 5.15. *There exists a $T > 0$ such that $\mathbb{C}(T)$ has one robot with light* `leader` *and all other robots with light* `off` *in same horizontal line.*

Lemma 5.16. *The leftmost robot r of a horizontal line can always see all the robots on the horizontal line $y = -1$ if there is no robot in $H_B^O(r) \cap H_U^O(r_0)$, where r_0 is the robot with colour* `leader` *on the line $y = -1$.*

Algorithm 15: LineWithLeader

```
1 Procedure LineWithLeader()
2     r ← myself
3     r_0 ← the robot with light leader
4     if (r_0 ∈ H_B^O(r)) and (r is leftmost on L_H(r)) and (there is no robot in
        H_B^O(r) ∩ H_U^O(r_0)) then
5         if there are i robots on L_H(r_0) other than r_0 at (1,−1),...,(i,−1) then
6             r.light ← off
7             move to an empty grid point towards (i+1,−1) by GotoLine
```

Proof. Let r be the leftmost robot on a horizontal line $y = s$ where $s > 0$. Then it is obvious that r will see all robots on $y = -1$. Now if r is on $y = 0$ and is singleton, then also it is obvious that r can see all robots on $y = -1$. Now if r is not singleton on $y = 0$. Then from the discussions above, it is clear that $x - coordinate$ of $r > x - coordinate$ of rightmost robot on $y = -1$. So, r can see all robots on $y = -1$. □

5.2.2.2 Target Pattern Formation

After the line is formed, all the robots except r_0 with colour `leader` on $(0, -1)$ have colour `off` and they are all placed on the horizontal line $y = -1$ in such a way that r_0 is the leftmost robot on $y = -1$ and any two robots do not have any unoccupied grid points between them. Now, the target pattern is embedded on the grid (based on the global coordinate system described above) such that $x - coordinate, y - coordinate$ of any target position say, t_i is greater than 0. Also, the target position of robot with colour `leader`, t_{n-1} is on line $y = 1$. And for any two other robots, let there be two target positions t_i and t_j. If both t_i and t_j are on the same horizontal line and $i < j$, then t_i is at the right of t_j. Also, if t_i and t_j are on two different horizontal lines and $i < j$, then t_i is on the above horizontal line (Figure 5.28).

Note that a robot say r who can see the robot r_0 with colour `leader` can agree on the global coordinate system as described earlier. So, r can agree on its target position which is embedded target positions on the grid. Now a robot r, who can

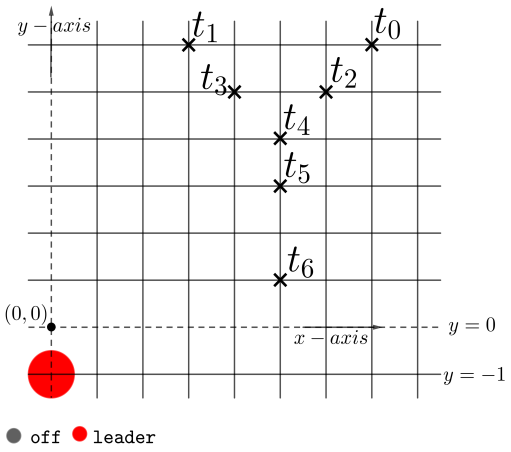


Figure 5.28: The target pattern embedded in the coordinate system.

see r_0 and has coordinate $(s, -1)$ on $\mathcal{L}_H(r)$ and sees $H_U^O(r)$ has no robot with colour `off`, moves to position $(s, 0)$ (Figure 5.29). Now, r can see all robots on $\mathcal{L}_H(r_0)$ as r is singleton on $y = 0$. Now, if r finds out that there are i robots on $\mathcal{L}_H(r_0)$ other than r_0 at the positions $(n-i, -1), (n-i-1, -1), \ldots (n-1, -1)$, then r moves to t_{n-i-2} by a method GotoTarget() (Figure 5.30) and changes its colour to `done`. The method GotoTarget() is described as follows. When executing the method GoToTarget(), a robot r first moves vertically to the horizontal line that is just below the horizontal line of its target location say t_r. Note that during this movement, no other robot on $\mathcal{L}_H(r_0)$ moves even if they see r_0 as they see r with colour `off` on their upper open half. Now, let coordinate of t_r be (x_{t_r}, y_{t_r}). Note that after the vertical movement, r is now singleton on the line $y = y_{t_r} - 1$. Now, if r is not already on the position $(x_{t_r}, y_{t_r} - 1)$, it moves horizontally to the position $(x_{t_r}, y_{t_r} - 1)$. Note that since r is singleton on $y = y_{t_r} - 1$ and no robot from $\mathcal{L}_H(r_0)$ has started its vertical movement (this is because they still see r with colour `off` on their upper open half), no collision occurs during this movement by r. Now when r reaches the position $(x_{t_r}, y_{t_r} - 1)$, it moves above once and reaches the designated target position t_r of r. Observe that the last robot say r_{n-1} on $(n-1, -1)$ (i.e on $\mathcal{L}_H(r_0)$) sees no other robot except r_0 on $\mathcal{L}_H(r_0)$ after it starts moving vertically above. Now, the problem is if it can distinguish whether r_{n-1} is meant to execute GoToLine()

or GoToTarget() when it is above the line $\mathcal{L}_H(r_0)$. Note that r_{n-1} will see at least one robot having colour `done` above it or on the same line while it is meant to execute GoToTarget() as $n > 2 \implies n-1 > 1$ which implies $\mathcal{L}_H(r_0)$ had at least one other robot r_{n-2} between r_0 and r_{n-1} which already executed GoToTarget() and changed its colour to `done` before r_{n-1} started executing GoToTarget(). So in this case, r_{n-1} sees at least one robot with colour `done` and moves to its designated target location $t_{r_{n-1}} = t_{n-2}$. So, we can conclude that after a finite time, all robots except the robot r_0 with colour `leader` move to their designated target locations embedded on the grid as described earlier.

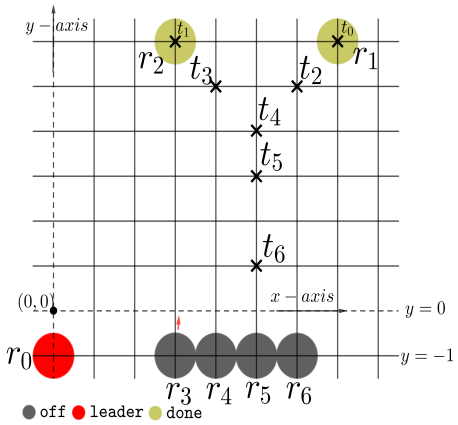


Figure 5.29: r_1 and r_2 already moved to their target position and changed their colour to `done`. r_3 sees r_0 with colour `leader` and moves to line $y = 0$ by moving vertically.

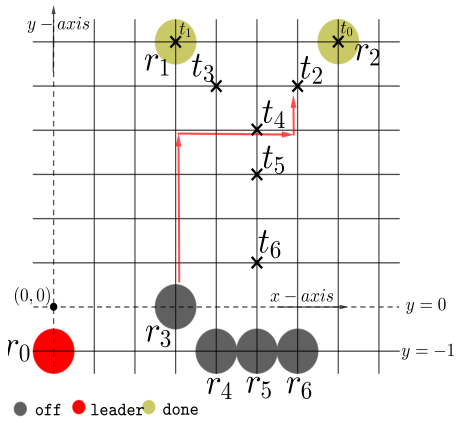


Figure 5.30: From $(3, 0)$, r_3 can see 3 robots on $(4, -1), (5, -1), (6, -1)$ and moves to t_2 by executing the method GoToTarget().

From the above discussion, we can conclude the following lemma.

Lemma 5.17. *During the execution of the method* GoToTarget(), *a robot r never collides with another robot in the configuration.*

Now the robot r_0 with colour `leader` sees that all the visible robots have colour `done`. So, it now moves to its designated target location t_{n-1} by a method LeaderMove(). The method LeaderMove() is described as follows. In this method, r_0 first moves to $(0, 0)$. Now, let the lowest horizontal line be $\mathcal{H}_{last}$ having a robot with colour `done`. Now, note that r_0 can always see the leftmost

robot r_{n-1} on the horizontal line $\mathcal{H}_{last}$. So, r_0 can always know its own position on the global coordinate as it knows the target position of r_{n-1} from the input even if it is not at $(0, -1)$. Now the target was embedded in such a way that the target position t_{n-1} of r_0 is on line $y = 1$. Let $(x_{t_0}, 1)$ be the target position of r_0. Now from $(0, 0)$, r_0 moves horizontally to the location $(x_{t_0}, 0)$ and then moves vertically once to $t_{n-1} = (x_{t_0}, 1)$ and changes the colour to `done`. Note that below $y = 1$, there is no other robot while r_0 starts moving (Figure 5.31).

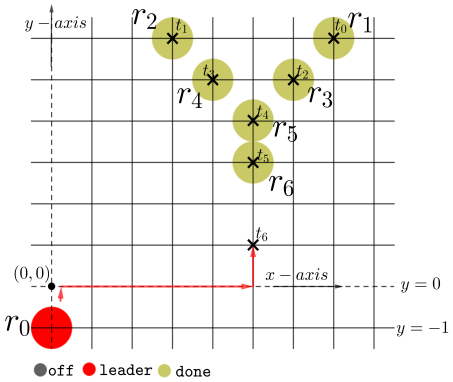


Figure 5.31: When all other robots except r_0 with colour `leader` reach their corresponding target positions, r_0 moves to t_6 executing the method LeaderMove().

So, we can conclude the following lemma.

Lemma 5.18. *While executing the method* LeaderMove(), *a robot r never collides with other robots in the configuration.*

So, from Lemmas 5.14, 5.17 and 5.18, we can directly conclude the following result.

Lemma 5.19. *During movement of robots in Phase 2, no collision occurs.*

Now from the above lemmas and the discussions, we can now finally conclude the following theorem.

Theorem 5.4. *There exists a $T > 0$ such that $\mathbb{C}(T)$ is a final configuration similar to the given pattern and has all robots with light* `done` *(Figure 5.32).*

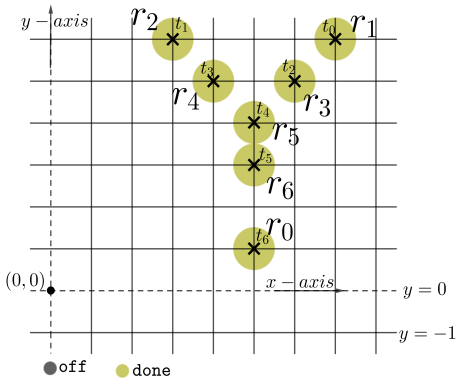


Figure 5.32: After r_0 reaches t_6, it changes its colour `done` and the target pattern has been formed.

5.3 Concluding remarks

The problem of arbitrary pattern formation ($\mathcal{APF}$) is a widely studied area of research in the field of swarm robotics. It has been studied under various assumptions on plane and discrete domain (eg. infinite regular tessellation grid). With obstructed visibility model, this problem has been considered on plane and infinite grid using luminous opaque robots. But using fat robots (i.e. robots with certain dimensions), it is only done in plane. In this chapter, we have taken care of this. We have shown that with a swarm of luminous opaque fat robots having one-axis agreement on an infinite grid, any arbitrary pattern can be formed from an initial configuration which is either asymmetric or has at least one robot on the line of symmetry using one light having 9 distinct colours which are less than the number of colours used to form an arbitrary pattern on plane using opaque and fat robots with one-axis agreement. For future courses of research, it would be interesting to see if the same problem can be solved using less number of colours under the same assumptions.

Chapter 6

Conclusion

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

In Chapter 2, we study the *network localization problem*, i.e., the problem of determining node positions of a wireless sensor network modeled as a unit disk graph. In this chapter, we propose a distributed localization scheme with a theoretical characterization of nodes that are guaranteed to be localized. However we considered the wireless sensor network modeled as a unit disk graph where two nodes are adjacent if and only if their distance is $\leq r$ (communication range$= r$). But more general network model is the quasi unit disk graph (QUDG) model.

Open Problem 1. *A distributed localization scheme with a theoretical characterization of nodes of a wireless sensor network modeled as a quasi unit disk graph.*

In Chapter 3, we consider the problem of gathering a set of autonomous, identical, oblivious, asynchronous, mobile robots at a vertex of an anonymous hypercube. We have shown that the problem is unsolvable if the robots do not have multiplicity detection capability. With weak multiplicity detection capability, we have provided a complete characterization of all gatherable configurations in ASYNC for $2k+1$ and $4k$ ($k > 0$) number of robots. However our strategy does not work for $4k + 2$ robots.

Open Problem 2. *Gathering in Hypercubes for* $4k+2$ $(k > 0)$ *number of robots.*

Also we considered that the visibility range of each robot is unlimited. The immediate open problem is to consider this problem with limited visibility.

Open Problem 3. *Gathering in Hypercubes with limited visibility.*

In Chapter 4, we studied $\mathcal{APF}$ on infinite grid with asynchronous opaque robots with lights. The robots do not share any global co-ordinate system, however they have an agreement over left and right. It would be interesting to study this problem without the agreement on the co-ordinate system.

Open Problem 4. $\mathcal{APF}$ *on infinite grid with asynchronous opaque robots without axis agreement.*

Again $\mathcal{LUMI}$ model has been considered in which the visible lights can be used by robots as a means of communication and persistent memory. In the $\mathcal{OBLOT}$ model, the robots are considered to be oblivious (i.e the robots do not have any persistent memory to remember any previous state). So the immediate open problem is to consider this problem in $\mathcal{OBLOT}$ model.

Open Problem 5. *Can* Arbitrary Pattern Formation *on infinite grid with asynchronous opaque robots be solved in* $\mathcal{OBLOT} + \mathcal{ASYNC}$ *model?*

In Chapter 5, we considered the $\mathcal{APF}$ problem assuming opaque fat robots on infinite grid. As in the Chapter 4, it has been assumed that the robots have an agreement over left and right and used model is $\mathcal{LUMI}$. So we have the following open problems:

Open Problem 6. $\mathcal{APF}$ *on infinite grid with asynchronous opaque fat robots without axis agreement.*

Open Problem 7. *Can* Arbitrary Pattern Formation *on infinite grid by opaque fat robots be solved in* $\mathcal{OBLOT} + \mathcal{ASYNC}$ *model?*

Also in Chapter 4 and Chapter 5, we considered that even if the robots have obstructed visibility, the visibility range of each robot is unlimited. The immediate open problem is to consider the $\mathcal{APF}$ with limited visibility.

Open Problem 8. Arbitrary Pattern Formation *on infinite grid by opaque point and fat robots with limited visibility.*

Bibliography

[1] Ranendu Adhikary, Manash Kumar Kundu, and Buddhadeb Sau. Circle formation by asynchronous opaque robots on infinite grid. *Comput. Sci.*, 22(1), 2021. doi:10.7494/csci.2021.22.1.3840.

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

[3] J. Albowicz, Alvin Chen, and Lixia Zhang. Recursive position estimation in sensor networks. In *9th International Conference on Network Protocols (ICNP 2001), 11-14 November 2001, Riverside, CA, USA*, pages 35–43. IEEE Computer Society, 2001. doi:10.1109/ICNP.2001.992758.

[4] James Aspnes, David Kiyoshi Goldenberg, and Yang Richard Yang. On the computational complexity of sensor network localization. In *Algorithmic Aspects of Wireless Sensor Networks: First International Workshop, ALGOSENSORS 2004, Turku, Finland, July 16, 2004. Proceedings*, volume 3121 of *Lecture Notes in Computer Science*, pages 32–44. Springer, 2004. doi:10.1007/978-3-540-27820-7_5.

[5] Lali Barrière, Paola Flocchini, Pierre Fraigniaud, and Nicola Santoro. Rendezvous and election of mobile agents: Impact of sense of direction. *Theory Comput. Syst.*, 40(2):143–162, 2007. doi: 10.1007/s00224-005-1223-5.

[6] Subhash Bhagat, Sruti Gan Chaudhuri, and Krishnendu Mukhopadhyaya. Gathering of opaque robots in 3d space. In Paolo Bellavista and Vijay K. Garg, editors, *Proceedings of the 19th International Conference on Distributed Computing and Networking, ICDCN 2018, Varanasi, India, January 4-7, 2018*, pages 2:1–2:10. ACM, 2018. doi:10.1145/3154273.3154322.

[7] Pratik Biswas, Tzu-Chen Liang, Ta-Chung Wang, and Yinyu Ye. Semidefinite programming based algorithms for sensor network localization. *ACM Trans. Sens. Networks*, 2(2):188–220, 2006. doi:10.1145/1149283.1149286.

[8] Pratik Biswas, Kim-Chuan Toh, and Yinyu Ye. A distributed SDP approach for large-scale noisy anchor-free graph realization with applications to molecular conformation. *SIAM J. Sci. Comput.*, 30(3):1251–1277, 2008. doi:10.1137/05062754X.

[9] Pratik Biswas and Yinyu Ye. Semidefinite programming for ad hoc wireless sensor network localization. In Kannan Ramchandran, Janos Sztipanovits, Jennifer C. Hou, and Thrasyvoulos N. Pappas, editors, *Proceedings of the Third International Symposium on Information Processing in Sensor Networks, IPSN 2004, Berkeley, California, USA, April 26-27, 2004*, pages 46–54. ACM, 2004. doi:10.1145/984622.984630.

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

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

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

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

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

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

[16] Quentin Bramas and Sébastien Tixeuil. Probabilistic asynchronous arbitrary pattern formation (short paper). In *Stabilization, Safety, and Security of Distributed Systems - 18th International Symposium, SSS 2016, Lyon, France, November 7-10, 2016, Proceedings*, pages 88–93, 2016. doi:10.1007/978-3-319-49259-9_7.

[17] Quentin Bramas and Sébastien Tixeuil. Probabilistic asynchronous arbitrary pattern formation (short paper). In Borzoo Bonakdarpour and Franck Petit, editors, *Stabilization, Safety, and Security of Distributed Systems - 18th International Symposium, SSS 2016, Lyon, France, November 7-10, 2016, Proceedings*, volume 10083 of *Lecture Notes in Computer Science*, pages 88–93, 2016. doi:10.1007/978-3-319-49259-9_7.

[18] Quentin Bramas and Sébastien Tixeuil. Arbitrary pattern formation with four robots. In Taisuke Izumi and Petr Kuznetsov, editors, *Stabilization,*

Safety, and Security of Distributed Systems - 20th International Symposium, SSS 2018, Tokyo, Japan, November 4-7, 2018, Proceedings, volume 11201 of *Lecture Notes in Computer Science*, pages 333–348. Springer, 2018. doi:10.1007/978-3-030-03232-6_22.

[19] Nirupama Bulusu, John S. Heidemann, and Deborah Estrin. Gps-less low-cost outdoor localization for very small devices. *IEEE Wirel. Commun.*, 7(5):28–34, 2000. doi:10.1109/98.878533.

[20] Tashnim J. S. Chowdhury, Colin Elkin, Vijay Devabhaktuni, Danda B. Rawat, and Jared Oluoch. Advances on localization techniques for wireless sensor networks: A survey. *Comput. Networks*, 110:284–305, 2016. doi:10.1016/j.comnet.2016.10.006.

[21] Po-Jen Chuang and Cheng-Pei Wu. An effective pso-based node localization scheme for wireless sensor networks. In *Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies, PDCAT 2008, Dunedin, Otago, New Zealand, 1-4 December 2008*, pages 187–194. IEEE Computer Society, 2008. doi:10.1109/PDCAT.2008.73.

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

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

[24] Mark Cieliebak, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Solving the robots gathering problem. In *Automata, Languages and Programming, 30th International Colloquium, ICALP 2003, Eindhoven, The Netherlands, June 30 - July 4, 2003. Proceedings*, pages 1181–1196, 2003. doi: 10.1007/3-540-45061-0_90.

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

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

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

[28] Jose A. Costa, Neal Patwari, and Alfred O. Hero III. Distributed weighted-multidimensional scaling for node localization in sensor networks. *ACM Trans. Sens. Networks*, 2(1):39–64, 2006. doi:10.1145/1138127.1138129.

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

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

[31] Gianlorenzo D'Angelo, Gabriele Di Stefano, and Alfredo Navarra. Gathering of six robots on anonymous symmetric rings. In *Structural Information and Communication Complexity - 18th International Colloquium, SIROCCO 2011, Gdansk, Poland, June 26-29, 2011. Proceedings*, pages 174–185, 2011. doi: 10.1007/978-3-642-22212-2_16.

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

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

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

[35] Shantanu Das, Matús Mihalák, Rastislav Srámek, Elias Vicari, and Peter Widmayer. Rendezvous of mobile agents when tokens fail anytime. In *Principles of Distributed Systems, 12th International Conference, OPODIS 2008, Luxor, Egypt, December 15-18, 2008. Proceedings*, pages 463–480, 2008. doi: 10.1007/978-3-540-92221-6_29.

[36] Anders Dessmark, Pierre Fraigniaud, Dariusz R. Kowalski, and Andrzej Pelc. Deterministic rendezvous in graphs. *Algorithmica*, 46(1):69–96, 2006. doi: 10.1007/s00453-006-0074-2.

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

[38] Stefan Dobrev, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Multiple agents rendezvous in a ring in spite of a black hole. In *Principles of Distributed Systems, 7th International Conference, OPODIS 2003 La Martinique, French West Indies, December 10-13, 2003 Revised Selected Papers*, pages 34–46, 2003. doi: 10.1007/978-3-540-27860-3_6.

[39] Tolga Eren, David Kiyoshi Goldenberg, Walter Whiteley, Yang Richard Yang, A. Stephen Morse, Brian D. O. Anderson, and Peter N. Belhumeur.

Rigidity, computation, and randomization in network localization. In *Proceedings IEEE INFOCOM 2004, The 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, Hong Kong, China, March 7-11, 2004*, pages 2673–2684. IEEE, 2004. doi:10.1109/INFCOM.2004.1354686.

[40] Jia Fang, Ming Cao, A. Stephen Morse, and Brian D. O. Anderson. Sequential localization of sensor networks. *SIAM J. Control. Optim.*, 48(1):321–350, 2009. doi:10.1137/070679144.

[41] Jia Fang, D. Duncan, and A. Stephen Morse. Sequential localization with inaccurate measurements. In *American Control Conference, ACC 2009. St. Louis, Missouri, USA, June 10-12, 2009*, pages 1970–1975. IEEE, 2009. doi:10.1109/ACC.2009.5160383.

[42] 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 - 20th International Symposium, SSS 2018, Tokyo, Japan, November 4-7, 2018, Proceedings*, pages 317–332, 2018. doi:10.1007/978-3-030-03232-6_21.

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

[44] Paola Flocchini, Evangelos Kranakis, Danny Krizanc, Flaminia L. Luccio, Nicola Santoro, and Cindy Sawchuk. Mobile agents rendezvous when tokens fail. In *Structural Information and Communication Complexity, 11th International Colloquium , SIROCCO 2004, Smolenice Castle, Slowakia, June 21-23, 2004, Proceedings*, pages 161–172, 2004. doi:10.1007/978-3-540-27796-5_15.

[45] Paola Flocchini, Evangelos Kranakis, Danny Krizanc, Nicola Santoro, and Cindy Sawchuk. Multiple mobile agent rendezvous in a ring. In *LATIN 2004: Theoretical Informatics, 6th Latin American Symposium, Buenos Aires, Argentina, April 5-8, 2004, Proceedings*, pages 599–608, 2004. doi: 10.1007/978-3-540-24698-5_62.

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

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

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

[49] Wade H Foy. Position-location solutions by taylor-series estimation. *IEEE transactions on aerospace and electronic systems*, (2):187–194, 1976.

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

[51] Chris Godsil and Gordon F Royle. *Algebraic graph theory*, volume 207. Springer Science & Business Media, 2013.

[52] David Kiyoshi Goldenberg, Arvind Krishnamurthy, Wesley C. Maness, Yang Richard Yang, Anthony Young, A. Stephen Morse, Andreas Savvides, and Brian D. O. Anderson. Network localization in partially localizable networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies, 13-17 March 2005, Miami, FL, USA*, pages 313–326. IEEE, 2005. doi:10.1109/INFCOM.2005.1497902.

[53] Pritam Goswami, Satakshi Ghosh, Avisek Sharma, and Buddhadeb Sau. Gathering on an infinite triangular grid with limited visibility under asynchronous scheduler. *CoRR*, abs/2204.14042, 2022. arXiv:2204.14042, doi:10.48550/arXiv.2204.14042.

[54] K Haba, T Izumi, Y Katayama, N Inuzuka, and K Wada. On gathering problem in a ring for 2n autonomous mobile robots. In *Proceedings of the 10th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS), poster*, 2008.

[55] Bruce Hendrickson. Conditions for unique graph realizations. *SIAM J. Comput.*, 21(1):65–84, 1992. doi:10.1137/0221008.

[56] Rui Huang and Gergely V. Záruba. Static path planning for mobile beacons to localize sensor networks. In *Fifth Annual IEEE International Conference on Pervasive Computing and Communications - Workshops (PerCom Workshops 2007), 19-23 March 2007, White Plains, New York, USA*, pages 323–330. IEEE Computer Society, 2007. doi:10.1109/PERCOMW.2007.109.

[57] Tomoko Izumi, Taisuke Izumi, Sayaka Kamei, and Fukuhito Ooshita. Mobile robots gathering algorithm with local weak multiplicity in rings. In *Structural Information and Communication Complexity, 17th International Colloquium, SIROCCO 2010, Sirince, Turkey, June 7-11, 2010. Proceedings*, pages 101–113, 2010. doi: 10.1007/978-3-642-13284-1_9.

[58] Bill Jackson and Tibor Jordán. Connected rigidity matroids and unique realizations of graphs. *J. Comb. Theory, Ser. B*, 94(1):1–29, 2005. doi: 10.1016/j.jctb.2004.11.002.

[59] Jinfang Jiang, Guangjie Han, Huihui Xu, Lei Shu, and Mohsen Guizani. LMAT: localization with a mobile anchor node based on trilateration in wireless sensor networks. In *Proceedings of the Global Communications Conference, GLOBECOM 2011, 5-9 December 2011, Houston, Texas, USA*, pages 1–6. IEEE, 2011. doi:10.1109/GLOCOM.2011.6133668.

[60] Sayaka Kamei, Anissa Lamani, Fukuhito Ooshita, and Sébastien Tixeuil. Asynchronous mobile robot gathering from symmetric configurations without global multiplicity detection. In *Structural Information and Communication Complexity - 18th International Colloquium, SIROCCO 2011, Gdansk, Poland, June 26-29, 2011. Proceedings*, pages 150–161, 2011. doi: 10.1007/978-3-642-22212-2_14.

[61] Sayaka Kamei, Anissa Lamani, Fukuhito Ooshita, and Sébastien Tixeuil. Gathering an even number of robots in an odd ring without global multiplicity detection. In *Mathematical Foundations of Computer Science 2012 - 37th International Symposium, MFCS 2012, Bratislava, Slovakia, August 27-31, 2012. Proceedings*, pages 542–553, 2012. doi: 10.1007/978-3-642-32589-2_48.

[62] Anushiya A. Kannan, Baris Fidan, and Guoqiang Mao. Analysis of flip ambiguities for robust sensor network localization. *IEEE Trans. Veh. Technol.*, 59(4):2057–2070, 2010. doi:10.1109/TVT.2010.2040850.

[63] Anushiya A. Kannan, Guoqiang Mao, and Branka Vucetic. Simulated annealing based localization in wireless sensor network. In *30th Annual IEEE Conference on Local Computer Networks (LCN 2005), 15-17 November 2005, Sydney, Australia, Proceedings*, pages 513–514. IEEE Computer Society, 2005. doi:10.1109/LCN.2005.125.

[64] Ralf Klasing, Adrian Kosowski, and Alfredo Navarra. Taking advantage of symmetries: Gathering of many asynchronous oblivious robots on a ring. *Theor. Comput. Sci.*, 411(34-36):3235–3246, 2010. doi: 10.1016/j.tcs.2010.05.020.

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

[66] Martyna Koreń. Gathering small number of mobile asynchronous robots on ring. *Zeszyty Naukowe Wydziału ETI Politechniki Gdańskiej. Technologie Informacyjne*, 18:325–331, 2010.

[67] Dimitrios Koutsonikolas, Saumitra M. Das, and Y. Charlie Hu. Path planning of mobile landmarks for localization in wireless sensor networks. *Comput. Commun.*, 30(13):2577–2592, 2007. doi:10.1016/j.comcom.2007.05.048.

[68] Evangelos Kranakis, Danny Krizanc, and Euripides Markou. Deterministic symmetric rendezvous with tokens in a synchronous torus. *Discrete Applied Mathematics*, 159(9):896–923, 2011. doi: 10.1016/j.dam.2011.01.020.

[69] Evangelos Kranakis, Nicola Santoro, Cindy Sawchuk, and Danny Krizanc. Mobile agent rendezvous in a ring. In *23rd International Conference on Distributed Computing Systems (ICDCS 2003), 19-22 May 2003, Providence, RI, USA*, pages 592–599, 2003. doi: 10.1109/ICDCS.2003.1203510.

[70] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. Ad hoc networks beyond unit disk graphs. *Wirel. Networks*, 14(5):715–729, 2008. doi:10.1007/s11276-007-0045-6.

[71] Pawel Kulakowski, Javier Vales-Alonso, Esteban Egea-López, Wieslaw Ludwin, and Joan García-Haro. Angle-of-arrival localization based on antenna arrays for wireless sensor networks. *Comput. Electr. Eng.*, 36(6):1181–1186, 2010. doi:10.1016/j.compeleceng.2010.03.007.

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

[73] Sangho Lee, Eunchan Kim, Chungsan Kim, and Kiseon Kim. Localization with a mobile beacon based on geometric constraints in wireless sensor networks. *IEEE Trans. Wirel. Commun.*, 8(12):5801–5805, 2009. doi:10.1109/TWC.2009.12.090319.

[74] Yunhao Liu, Zheng Yang, Xiaoping Wang, and Lirong Jian. Location, localization, and localizability. *J. Comput. Sci. Technol.*, 25(2):274–297, 2010. doi:10.1007/s11390-010-9324-2.

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

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

[77] Giuseppe Antonio Di Luna, Paola Flocchini, Linda Pagli, Giuseppe Prencipe, Nicola Santoro, and Giovanni Viglietta. Gathering in dynamic rings. In *Structural Information and Communication Complexity - 24th International Colloquium, SIROCCO 2017, Porquerolles, France, June 19-22, 2017, Revised Selected Papers*, pages 339–355, 2017. doi: 10.1007/978-3-319-72050-0_20.

[78] Guoqiang Mao, Baris Fidan, and Brian D. O. Anderson. Wireless sensor network localization techniques. *Comput. Networks*, 51(10):2529–2553, 2007. doi:10.1016/j.comnet.2006.11.018.

[79] Gianluca De Marco, Luisa Gargano, Evangelos Kranakis, Danny Krizanc, Andrzej Pelc, and Ugo Vaccaro. Asynchronous deterministic rendezvous in graphs. *Theor. Comput. Sci.*, 355(3):315–326, 2006. doi: 10.1016/j.tcs.2005.12.016.

[80] David C. Moore, John J. Leonard, Daniela Rus, and Seth J. Teller. Robust distributed network localization with noisy range measurements. In John A. Stankovic, Anish Arora, and Ramesh Govindan, editors, *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, SenSys 2004, Baltimore, MD, USA, November 3-5, 2004*, pages 50–61. ACM, 2004. doi:10.1145/1031495.1031502.

[81] David C. Moore, John J. Leonard, Daniela Rus, and Seth J. Teller. Robust distributed network localization with noisy range measurements. In John A. Stankovic, Anish Arora, and Ramesh Govindan, editors, *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, SenSys 2004, Baltimore, MD, USA, November 3-5, 2004*, pages 50–61. ACM, 2004. doi:10.1145/1031495.1031502.

[82] Asis Nasipuri and Kai Li. A directionality based location discovery scheme for wireless sensor networks. In Cauligi S. Raghavendra and Krishna M. Sivalingam, editors, *Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications, WSNA 2002, Atlanta, Georgia, USA, September 28, 2002*, pages 105–111. ACM, 2002. doi:10.1145/570738.570754.

[83] Dragos Niculescu and B. R. Badrinath. Ad hoc positioning system (APS) using AOA. In *Proceedings IEEE INFOCOM 2003, The 22nd Annual Joint Conference of the IEEE Computer and Communications Societies, San Franciso, CA, USA, March 30 - April 3, 2003*, pages 1734–1743. IEEE Computer Society, 2003. doi:10.1109/INFCOM.2003.1209196.

[84] Ewa Niewiadomska-Szynkiewicz and Michal Marks. Optimization schemes for wireless sensor network localization. *Int. J. Appl. Math. Comput. Sci.*, 19(2):291–302, 2009. doi:10.2478/v10006-009-0025-3.

[85] Chia-Ho Ou and Kuo-Feng Ssu. Sensor position determination with flying anchors in three-dimensional wireless sensor networks. *IEEE Trans. Mob. Comput.*, 7(9):1084–1097, 2008. doi:10.1109/TMC.2008.39.

[86] Linda Pagli, Giuseppe Prencipe, and Giovanni Viglietta. Getting close without touching: near-gathering for autonomous mobile robots. *Distributed Computing*, 28(5):333–349, 2015. doi: 10.1007/s00446-015-0248-5.

[87] David Peleg. *Distributed computing: a locality-sensitive approach*. SIAM, 2000.

[88] David Peleg. Distributed coordination algorithms for mobile robot swarms: New directions and challenges. In Ajit Pal, Ajay D. Kshemkalyani, Rajeev Kumar, and Arobinda Gupta, editors, *Distributed Computing - IWDC 2005, 7th International Workshop, Kharagpur, India, December 27-30, 2005, Proceedings*, volume 3741 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2005. doi:10.1007/11603771_1.

[89] Rong Peng and Mihail L. Sichitiu. Angle of arrival localization for wireless sensor networks. In *Proceedings of the Third Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks, SECON 2006, September 25-28, 2006, Reston, VA, USA*, pages 374–382. IEEE, 2006. doi:10.1109/SAHCN.2006.288442.

[90] Giuseppe Prencipe. Impossibility of gathering by a set of autonomous mobile robots. *Theor. Comput. Sci.*, 384(2-3):222–231, 2007. doi: 10.1016/j.tcs.2007.04.023.

[91] Nasir Saeed, Haewoon Nam, Tareq Y. Al-Naffouri, and Mohamed-Slim Alouini. A state-of-the-art survey on multidimensional scaling-based localization techniques. *IEEE Commun. Surv. Tutorials*, 21(4):3565–3583, 2019. doi:10.1109/COMST.2019.2921972.

[92] B. Sau and K. Mukhopadhyaya. Length-based anchor-free localization in a fully covered sensor network. In *2009 First International Communication Systems and Networks and Workshops*, pages 1–10, Jan 2009. doi:10.1109/COMSNETS.2009.4808851.

[93] Andreas Savvides, Chih-Chieh Han, and Mani B. Srivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In Christopher Rose,

editor, *MOBICOM 2001, Proceedings of the seventh annual international conference on Mobile computing and networking, Rome, Italy, July 16-21, 2001*, pages 166–179. ACM, 2001. doi:10.1145/381677.381693.

[94] James B Saxe. Embeddability of weighted graphs in k-space is strongly np-hard. In *Proc. of 17th Allerton Conference in Communications, Control and Computing, Monticello, IL*, pages 480–489, 1979.

[95] Yi Shang and Wheeler Ruml. Improved mds-based localization. In *Proceedings IEEE INFOCOM 2004, The 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, Hong Kong, China, March 7-11, 2004*, pages 2640–2651. IEEE, 2004. doi:10.1109/INFCOM.2004.1354683.

[96] Yi Shang, Wheeler Ruml, Ying Zhang, and Markus P. J. Fromherz. Localization from mere connectivity. In *Proceedings of the 4th ACM Interational Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2003, Annapolis, Maryland, USA, June 1-3, 2003*, pages 201–212. ACM, 2003. doi:10.1145/778415.778439.

[97] Francesco Betti Sorbelli, Cristina M. Pinotti, and Vlady Ravelomanana. Range-free localization algorithm using a customary drone: Towards a realistic scenario. *Pervasive Mob. Comput.*, 54:1–15, 2019. doi:10.1016/j.pmcj.2019.01.005.

[98] Francesco Sottile and Maurizio A. Spirito. Robust localization for wireless sensor networks. In *Proceedings of the Fifth Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON 2008, June 16-20, 2008, Crowne Plaza, San Francisco International Airport, California, USA*, pages 46–54. IEEE, 2008. doi:10.1109/SAHCN.2008.16.

[99] Seshan Srirangarajan, Ahmed H. Tewfik, and Zhi-Quan Luo. Distributed sensor network localization using SOCP relaxation. *IEEE Trans. Wirel. Commun.*, 7(12-1):4886–4895, 2008. doi:10.1109/T-WC.2008.070241.

[100] Kuo-Feng Ssu, Chia-Ho Ou, and Hewijin Christine Jiau. Localization with mobile anchor points in wireless sensor networks. *IEEE Trans. Veh. Technol.*, 54(3):1187–1197, 2005. doi:10.1109/TVT.2005.844642.

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

[102] Gabriele Di Stefano and Alfredo Navarra. Optimal gathering of oblivious robots in anonymous graphs and its application on trees and rings. *Distributed Computing*, 30(2):75–86, 2017. doi: 10.1007/s00446-016-0278-7.

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

[104] Gerard Tel. Network orientation. *Int. J. Found. Comput. Sci.*, 5(1):23–57, 1994. doi: 10.1142/S0129054194000037.

[105] Paul Tseng. Second-order cone programming relaxation of sensor network localization. *SIAM J. Optim.*, 18(1):156–185, 2007. doi:10.1137/050640308.

[106] George L Turin, William S Jewell, and Tom L Johnston. Simulation of urban vehicle-monitoring systems. *IEEE Transactions on Vehicular Technology*, 21(1):9–16, 1972.

[107] W.T. Tutte. *Graph Theory*. Cambridge Mathematical Library. Cambridge University Press, 2001.

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

[109] Massimo Vecchio, Roberto López-Valcarce, and Francesco Marcelloni. A two-objective evolutionary approach based on topological constraints for node localization in wireless sensor networks. *Appl. Soft Comput.*, 12(7):1891–1901, 2012. doi:10.1016/j.asoc.2011.03.012.

[110] Jing Wang, Ratan K Ghosh, and Sajal K Das. A survey on sensor localization. *Journal of Control Theory and Applications*, 8(1):2–11, 2010. doi: 10.1007/s11768-010-9187-7.

[111] Walter Whiteley. Some matroids from discrete applied geometry. *Contemporary Mathematics*, 197:171–312, 1996.

[112] Hongyuan Zha Xiang Ji. Sensor positioning in wireless ad-hoc sensor networks with multidimensional scaling. In *Proceedings IEEE INFOCOM 2004, The 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, Hong Kong, China, March 7-11, 2004*, pages 2652–2661. IEEE, 2004. doi:10.1109/INFCOM.2004.1354684.

[113] Bin Xiao, Hekang Chen, and Shuigeng Zhou. Distributed localization using a moving beacon in wireless sensor networks. *IEEE Trans. Parallel Distributed Syst.*, 19(5):587–600, 2008. doi:10.1109/TPDS.2007.70773.

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

[115] Yukiko Yamauchi and Masafumi Yamashita. Pattern formation by mobile robots with limited visibility. 07 2013. doi:10.1007/978-3-319-03578-9_17.

[116] Zheng Yang, Yunhao Liu, and Xiang-Yang Li. Beyond trilateration: On the localizability of wireless ad hoc networks. *IEEE/ACM Trans. Netw.*, 18(6):1806–1814, 2010. doi:10.1109/TNET.2010.2049578.

List of Publications

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

- Manash Kumar Kundu, Pritam Goswami, Satakshi Ghosh and Buddhadeb Sau. **Arbitrary pattern formation by opaque fat robots on infinite grid**. International Journal of Parallel, Emergent and Distributed Systems, Vol. 37, pages 542-570, 2022. https://doi.org/10.1080/17445760.2022.2088750

- Kaustav Bose, Manash Kumar Kundu, Ranendu Adhikary and Buddhadeb Sau. **Optimal Gathering by Asynchronous Oblivious Robots in Hypercubes**. Algorithms for Sensor Systems. ALGOSENSORS 2018. Lecture Notes in Computer Science(), vol 11410. Springer, Cham. https://doi.org/10.1007/978-3-030-14094-6_7

- Kaustav Bose, Manash Kumar Kundu, Ranendu Adhikary and Buddhadeb Sau. **Distributed Localization of Wireless Sensor Network Using Communication Wheel**.Information and Computation, 2022,104962,ISSN 0890-5401, https://doi.org/10.1016/j.ic.2022.104962.

  An earlier version of the paper appeared in Algorithms for Sensor Systems. ALGOSENSORS 2020. Lecture Notes in Computer Science(), vol 12503. Springer, Cham. https://doi.org/10.1007/978-3-030-62401-9_2

- Manash Kumar Kundu, Pritam Goswami, Satakshi Ghosh and Buddhadeb Sau. **Arbitrary pattern formation by asynchronous opaque robots on infinite grid**. Arxiv e-prints, May. 2022. arXiv:2205.03053. https://doi.org/10.48550/arXiv.2205.03053 (Communicated)

Index