Circle Formation by Autonomous, Anonymous and Oblivious Mobile Robots

A Project Report Submitted

in Partial Fulfilment of the Requirements

for the Degree of

BACHELOR OF TECHNOLOGY

in

Mathematics and Computing

by

Chirag Maheshwari

(Roll No. 10012315)

to the

DEPARTMENT OF MATHEMATICS

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

GUWAHATI - 781039, INDIA

April 2014

CERTIFICATE

This is to certify that the work contained in this project report entitled "**Circle Formation by Autonomous, Anonymous and Oblivious Mobile Robots**" submitted by **Chirag Maheshwari** (**Roll No.: 10012315**) to Indian Institute of Technology Guwahati towards partial requirement of **Bachelor of Technology** in Mathematics and Computing has been carried out by him/her under my supervision and that it has not been submitted elsewhere for the award of any degree.

Guwahati - 781 039
April 2014

(Partha Sarathi Mandal)
Project Supervisor

ABSTRACT

The distributed coordination and control of a set of autonomous mobile robots is a problem widely studied in a variety of fields, such as engineerig, artificial intelligence, artificial life, robotics. The problem of pattern formation by autonomous robots is practically important, because, if the robots can form a given pattern, they can agree on their respective roles in any coordinated action.

In the first phase, two distributed algorithms were proposed and simulated which when executed by a set of autonomous, anonymous mobile robots lying on a circle (non-uniform) will move to form a uniform circle. The robots were assumed to be dimensionless (point-based robots), anonymous, oblivious, haveing no common coordination system but a common sense of orientation of those axes.

In the second phase, we extended our previous work by taking a more practical approach. The assumption of modelling robots as dimensionless points does not reflect reality as real robots are not points but have a physical extent. Thus, the robots were modelled as *fat* robots: closed transparent unit discs. We solved the *minimum perimeter circle formation* problem with a set of anonymous, oblivious, transparent fat-robots in a synchronized setting.

Contents

List of Figures **vi**

1 Introduction **1**

1.1 Motivation 1

1.2 Problem from 10000 feet 2

1.3 Definitions 4

2 Literature Review **9**

3 Uniform Circle Formation **18**

3.1 Definitions and Notations 19

3.2 Circle Formation 20

3.3 Uniform Transformation 23

3.3.1 Situation 1 23

3.3.2 Situation 2 25

3.4 Combining the two parts 27

4 Minimum Perimeter Circle Formation **30**

4.1 Problem Definition 30

4.2 Underlying Model 30

4.3 Overview 32

4.4 Algorithm 34
4.5 Correctness 40

5 Conclusion **42**

Bibliography **44**

List of Figures

3.1 Restriction 1: The movement of r_5 is constrained by the interior of its voronoi cell 22
3.2 Restriction 2: The movement of r_5 is constrained by smallest enclosing circle 22
3.3 Restriction 1-4: r_1, r_2, r_6 move on the boundary of the enclosing circle (Restriction 3-4): r_3, r_4, r_5 move in interior 22
3.4 Initial states of all the robots. The faulty robots is shown in blue color (although it is anonymous in theory) 25
3.5 Final states of all the robots. The faulty robots is shown in blue color (although it is anonymous in theory). As shown, after a finite number of iterations, they form a regular polygon. 26
3.6 Initial states of all the robots. Algorithm 4. 28
3.7 Final states of all the robots. Algorithm 4. They form a regular polygon. 29

List of Algorithms

1 Formation of an (arbitrary) circle (code executed by robot r_i) 21

2 Convergence towards a uniform transformation 23

3 uniform transformation with one faulty robot 24

4 uniform transformation with movement restriction in one direction 27

5 uniform circle formation (Combining the parts) 28

6 SECexpansion(r) - Expands the current SEC to make space for all the robots and also helps to get out of lock configuration. 34

7 CheckLock(r, T_r) - Checks if moving the robot r to position T_r will lead to a lock configuration or not. 35

8 ComputeDestination(r) - Computes the destination on the SEC for robots belonging to $C(m-1)$ for the initial formation of circle. 36

9 Converge(r) - After a circle is formed, this algorithm converges the robot to minimum perimeter circle. 38

10 MinimumPerimeterCircleFormation(r) - This is the main method called by every robot in it's computing phase which in turn uses the above algorithms to perform it's task. 39

Chapter 1

Introduction

From an engineering point of view, the problem of coordinating a set of autonomous, mobile robots for the purpose of cooperatively performing a task has been studied extensively over the past decade. As discussed in the subsequent sections, distributed approach to control a set of weak autonomous robots instead of few strong autonomous robots has many advantages. In this project we focus on *point-based robots* to form an uniform circle and then shift our focus to a more practical approach to model *robots as unit-discs*. In the remainder of the report, we will propose and discuss some distributed algorithms to control a set of autonomous robots in the different models described.

1.1 Motivation

An interesting trend has been observed in robotic research, both from engineering and behavioural viewpoints. They are readily moving away from the design and deployment of few, rather complex, usually expensive, application-specific robots. In fact, within this trend, the interest has shifted towards the

design and use of a large number of "generic" robots which are very simple, with very limited capabilities, and thus relatively inexpensive but capable of performing (together) rather complex tasks. The main idea is to let each robot execute a simple algorithm and plan its motion adaptively based on the observed movement of other robots, so that the robots as a group will achieve the given goal.

There is an increasing number of applications that benefit from having a team of autonomous robots to cooperate and complete various tasks in a self-organizing manner. Such application tasks may require, for example, that robots work in dangerous and harsh environments (e.g., for space, underwater or military purposes) or achieve high accuracy or speed (e.g., in nanotechnology, scientisc computing). It is usually desirable for the robots to be as simple as possible and have limited computing power, in order to be able to produce them fast in large numbers and cheap.

The advantages of such an approach are clear and many. They include: reduced costs (due to simpler engineering and construction costs, faster development and deployment time, etc); ease of system expandability (just add a few more robots) which in turns allows for incremental and on-demand deployment (use only as few robots as you need and when you need them); simple and affordable fault-tolerance capabilities (replace just the faulty robots); re-usability of the robots in different applications (reprogram the system to perform a different task).

1.2 Problem from 10000 feet

A fundamental problem that has drawn much attention recently is *gathering*, where a team of autonomous mobile robots must gather to a certain point or

region or form a certain formation (e.g., geometric shape) in the plane. The problem has been studied under various modeling assumptions; for example, asynchronous, semi-synchronous and synchronous settings have been considered. Robots may have a common coordination system, or have common sense of direction and use compasses to navigate in the plane; they may have stable memory or be history-oblivious. In all considered models, robots are equipped with a vision device (e.g., a camera) and their range of visibility is either limited or unlimited. Robots operate under the Look-Compute-Move cycle. Within a cycle, a robot takes a snapshot of the plane (Look), performs some local computation (Compute), and possibly decides to move to some other point in the plane (Move).

In particular, we address the problem of ***circle formation*** by a group of mobile robots. The problem of circle formation has interesting applications. For instance, consider the context of space exploration and the initial preparation of a zone. A group of robots could be sent and after landing at random locations, would self-organize to form the initial infrastructure for later expeditions. Also, pattern formation is the first step towards flocking, i.e., allowing a group of robots to move in formation. Moreover, the formation of geometrical patterns and flocking are both useful in themselves for the self-positioning of mobile base stations in a mobile ad-hoc network and self-deployment of sensor rings.

In layman terms, suppose that a schoolteacher wants her 100 children in the playground to form a circle so that, for instance, they can play a game. She might draw a circle on the ground as a guideline or even give each child a specific position to move to. What if the teacher does not provide such assistance? Even without such assistance, the children may still be able to form a sufficiently good approximation of a circle if each of them

moves adaptively based on the movement of other children and knowledge of the shape of a circle. If successful, this method can be called a *distributed solution to the circle formation problem for children*. A similar distributed approach can be used for controlling a group of multiple mobile robots.

Earlier we proposed two algorithms as a solution to uniform transformation problem by making a strong assumption in our model. In this project, we propose solution to the minimum circle formation problem with transparent fat-robots in a synchronous setting.

1.3 Definitions

We start with introducing some definitions that will be used in the rest of the report.

Definition 1.3.1. ***Robots***. We assume n synchronous/asynchronous, fault-free robots that can move along straight lines on the (infinite) plane. The robots can be *point-based* or *closed unit-discs*. They are identical and anonymous (i.e., they are indistinguishable). They do not have access to any global coordination system, but we assume *chirality*: the robots agree on the orientation of the axes. Robots are equipped with a 360-degree-angle vision device (e.g., camera) that enables the robots to take snapshots of the plane. The vision device has unlimited range and captures any point of the plane provided there is no obstacle (e.g., another robot if robots are non-transparent).

Definition 1.3.2. ***Computational Cycle***.The robots execute the same deterministic algorithm, which takes as input the observed positions of the robots within the visibility radius, and returns a destination point towards which the executing robot moves. A robot is initially in a waiting state (Wait); asynchronously and independently from the other robots, it observes

the environment in its area of visibility (Look); it calculates its destination point based only on the observed locations of the robots in its (Compute); it then moves towards that point (Move); after the move it goes back to a waiting state. The sequence: Wait Look Compute Move will be called a computation cycle (or briefly cycle) of a robot. The operations performed by the robots in each state will be now described in more details.

Wait The robot is idle. A robot cannot stay idle indefinitely unless it is faulty. At the beginning all the robots are in the Wait state.

Look The robot observes the world by activating its sensors which will return a *snapshot* of the positions of all other robots with respect to its local coordinate system. Each robot is viewed as a point, hence its position in the plane is given by its coordinates, and the result of the snapshot is just a set of coordinates in its local coordinate system: this set forms the *view of the world* of r . More formally, the view of the world of r at time t is defined as the last snapshot taken at a time smaller than or equal to t.

Compute The robot performs a local computation according to its deterministic, oblivious algorithm $\mathcal{A}$. The result of the computation is a destination point; if this point is the current location, the robot stays still (null movement).

Move If the point computed in the previous state is the current location, the robot does not move; otherwise it moves towards the destination point. The robot moves by an unpredictable amount of space, which is assumed neither infinite, nor infinitesimally small. Hence, the robot can only go towards its goal, but it cannot predict how far it will go

in the current cycle, because it can stop anytime during its movement; that is, a robot can stop before reaching its destination point.

Definition 1.3.3. ***Asynchronous Setting***. The robots are said to be in a asynchronous setting if the amount of time spent in observation, in computation, in movement, and in action is negligible but otherwise unpredictable. In particular, the robots do not (need to) have a common notion of time. Each robot makes steps at unpredictable time instants. The (global) time that passes between two successive steps of the same robot is finite; that is, any desired finite number of steps could have been made by any robot after some finite amount of time. In addition, we do not make any timing assumptions within a step: The time that passes after the robot has observed the positions of all others and before it starts moving is arbitrary, but finite. That is, the actual move of a robot may be based on a situation that lies arbitrarily far in the past, and therefore it may be totally different from the current situation. We feel that this assumption of asynchronicity within a step is important in a totally asynchronous environment, since we want to give each robot enough time to perform its local computation. In particular, the amount of time spent in *Wait, Look, Compute, Move*, and idle states is finite but otherwise unpredictable. As a result, the robots do not have a common notion of time, robots can be seen while moving, and computations can be made based on obsolete observations.

Definition 1.3.4. ***Synchronous Setting***. The robots are said to be in a synchronous setting if the robots execute their activities in an atomic and instantaneous fashion, we say that the robots are *atomically synchronized*, and that they move according to an schedule. In a synchronized setting, the amount of time spent in *Wait, Look, Compute, Move* cycle is finite and equal

for every robot. In particular, wait, look, compute and move states are all synchronized with a central clock.

Definition 1.3.5. ***Geometric Configuration***. A geometric configuration is a vector $G = \{c_1, c_2, ..., c_n\}$ where each c_i represents the center of the position of robot r_i on the plane. So, a configuration can be viewed as a snapshot of the robots on the plane. Note that the fact that robots are fat prohibits the formation of a configuration in which any two robots share more than a point in the plane. (Two robots share a point if the discs representing them touch each other.)

We say that a geometric configuration G is connected if between any two points of any two robots there is a polygonal line each of whose points belongs to some robot. Informally, a configuration is connected if every robot touches another robot and all robots form together a connected formation.

Definition 1.3.6. ***Visibility and fully visible configuration***. This definition is only valid for non-transparent robots. We say that point p in the plane is *visible* by a robot r_i (or equivalently, r_i can see p) if there is a point p_i in the circle bounding robot r_i such that the straight segment (p_i, p) does not contain any point of any other robot. So, a robot r_i can see another robot r_j if there is at least one point on the bounding circle of r_j that is visible by r_i . Given a geometric configuration G, robot r_i has *full visibility* in G if r_i can see all other $n-1$ robots. If all robots have full visibility in G, then configuration G is *fully visible*

Definition 1.3.7. ***Uniform Circle Formation***. The problem of uniform circle formation can be divided into two parts:

1. Forming a circle, possibly an non-uniform one. (Circle Formation Problem)

2. Positioning the robots evenly on the boundary, i.e., forming a regular polygon. (Uniform Transformation)

Definition 1.3.8. ***Circle Formation***. Given a group of n robots $r_1, r_2, ..., r_n$ with distinct positions and located arbitrarily on the plane, arrange them to eventually form a non-degenrate circle.

Definition 1.3.9. ***Uniform Transformation***. Given a group of n robots $r_1, r_2, ..., r_n$ with distinct positions and located arbitrarily on a plane on the boundary of some non-degenrate circle (i.e., with finite radius greater than zero), eventually arrange them at regular intervals of the boundary of the circle.

Definition 1.3.10. ***Minimum Perimeter Circle Formation***. Given a group of n fat robots $\{r_1, r_2, ..., r_n\}$ with distinct positions and located arbitrarily on a plane, eventually arrange them at a boundary of a circle such that the perimeter is minimum and the final configuration is connected and terminal.

Chapter 2

Literature Review

A vast amount of researches exist in the context of cooperative mobile robotics. Most of it uses diverse heuristics such as free market optimization [9] or swarm intelligence [22]. However, only few studies take the problem from a computational standpoint. This can be partly explained by the difficulty of the task, and the fact that heuristics are perceived as a way to circumvent that difficulty. Debest [6] briefly discusses the formation of a circle by a group of mobile robots as an illustration of self-stabilizing distributed algorithms. He discusses the problem, but does not really provide an algorithm. Sugihara and Suzuki [25] propose several algorithms for the formation of various geometrical patterns. They propose an algorithm for the formation of an approximation of a circle, based on heuristics. In some cases, the shape obtained with their algorithm is a Reuleaux triangle (a hybrid shape, between a triangle and a circle) rather than a circle. Suzuki and Yamashita [26] propose a non-oblivious algorithm for the formation of a regular polygon. In other words, the robots eventually reach a configuration in which they are arranged at regular intervals on the boundary of a circle. To achieve this, they must however require the robots to be able to remember all past actions. Under

the same model, Ando et al. [19] propose an algorithm by which robots with a limited range of vision gather to a point. Flocchini et al. [14] give an algorithm to solve the same problem in a slightly different model; dropping the assumption of instantaneous computation and movement, but assuming a common sense of direction as given by compasses. Flocchini et al. [13] study the solvability of the formation of arbitrary patterns, depending on how much common knowledge the robots initially have about a global coordinate system. Uny Cao et al. [27] provide a wide survey of researches in cooperative mobile robotics, and observe that only few researches take the problem from a computational point of view. This observation is later echoed by Flocchini et al. [14]. The remainder of the chapter summarizes the assumptions and results collected from various research papers.

From the works of G. Prencipe [23], given n robots with the following capabilities cannot solve the gathering problem:

- Autonomous
- Anonymous
- Homogeneous
- Oblivious
- Asynchronous, Synchoronized
- Unlimited Visibilty
- Dimensionless robots (point robots)
- Do not detect *multiplicity*
- Initial positions are all distinct

Under the above assumptions G. Prencipe [23] says that,

Theorem 2.0.11. *In both the asynchronous and the atomic time setting, there exists no deterministic oblivious algorithm that solves the gathering problem in a finite number of cycles, hence in finite time, for a set of $n \geq 2$ robots.*

In another work by M. Yamashita [26], after taking an extra assumption of multiplicity detection, they proved that gathering problem is solvable. The assumptions taken by M. Yamashita [26] are as follows:

- Autonomous
- Anonymous
- Homogeneous
- Oblivious
- Asynchronous, Synchoronized
- Unlimited Visibilty
- Dimensionless robots (point robots)
- **Detect *multiplicity***
- Initial positions are all distinct

Under the above assumptions, M. Yamashita [26] quotes,

Theorem 2.0.12. *There exists an oblivious algorithm for solving gathering problem in a finite number of steps for $n \geq 3$.*

Proof. It suffices to give an oblivious algorithm $\mathcal{A}$ that solves the gathering problem in a finite number of cycles. The idea is the following. Starting from distinct initial positions, we move the robots in such a way that eventually there will be exactly one position, say, p, that two or more robots occupy. Once such a distribution is reached, all robots that are not located at p move toward p in such a way that no two robots will occupy the same position at any location other than p. Then all robots eventually occupy p, solving gathering problem in finite time.

Such a distribution can be obtained if each robot, each time it becomes active, determines which of the following cases applies and moves to a new position (or remains stationary) as specified. Since a robots action is based only on the current robot distribution, this strategy can be implemented as an oblivious algorithm.

Case 1. $n = 3$ p_1, p_2 and p_3 denote the positions of the three robots

1. If $n = 3$ and p_1, p_2, p_3 are collinear with p_2 in the middle, then the robots at p_1 and p_3 move toward p_2 while the robot at p_2 remains stationary. Then eventually two robots occupy p_2.
2. If $n = 3$ and p_1, p_2 and p_3 form an isosceles triangle with $\|\overline{p_1p_2}\| = \|\overline{p_1p_3}\| \neq \|\overline{p_2p_3}\|$, then the robot at p_1 moves toward the foot of the perpendicular drop from its current position to p_2 p_3 in such a way that the robots do not form an equilateral triangle at any time, while the robots at p_2 and p_3 remain stationary. Then eventually the robots become collinear and the problem is reduced to part 1.
3. If $n = 3$ and the lengths of the three sides of triangle $p_1p_2p_3$ are all different, say, $\|\overline{p_1p_2}\| > \|\overline{p_1p_3}\| > \|\overline{p_2p_3}\|$, then the robot at p_3 moves toward the foot of the perpendicular drop from its current

position to p_1p_2 while the robots at p_1 and p_2 remain stationary. Then eventually the robots become collinear and the problem is reduced to part 1.

4. If $n = 3$ and p_1, p_2, p_3 form an equilateral triangle, then every robot moves towards the center of the triangle. Since all robots can move up to at least a constant distance ϵ greater than 0 in one step, if part 4 continues to hold then eventually either the robots meet at the center, or the triangle they form becomes no longer equilateral and the problem is reduced to part 2 or part 3.

Case 2. $\boldsymbol{n \geq 4}$ C_t denotes the smallest enclosing circle of the robots at time t

1. If $n \geq 4$ and there is exactly one robot r in the interior of C_t, then r moves toward the position of any one robot, say, r , on the circumference of C_t while all other robots remain stationary. Then eventually r and r occupy the same position.
2. If $n \geq 4$ and there are two or more robots in the interior of C_t , then these robots move toward the center of C_t while all other robots remain stationary (so that the center of C_t remains unchanged). Then eventually at least two robots reach the center.
3. If $n \geq 4$ and there are no robots in the interior of C_t , then every robot moves toward the center of C_t . Since all robots can move up to at least a constant distance ϵ greater than 0 in one step, if part 3 continues to hold, then eventually the radius of C_t becomes at most . Once this happens, then the next time some robot moves, say, at t' , either (i) two or more robots occupy the center of C_t or (ii) there is exactly one robot r at the center of C'_t , and therefore

there is a robot that is not on C_t (and the problem is reduced to part 1 or part 2) since a cycle passing through r and a point on C_t intersects with C_t at most at two points.

□

From the works of M. Yamashita [26], given n robots can solve the pattern formation problem under the following assumptions:

- Autonomous
- Anonymous
- Homogeneous
- ***nonoblivious***
- Asynchronous, Synchoronized
- Unlimited Visibilty
- Dimensionless robots (point robots)
- **Detect *multiplicity***
- Initial positions are all distinct

M. Yamashita [26] says that,

Theorem 2.0.13. *There exists an algorithm for solving formation problem for a predicate* π *iff either* $\pi = \pi_{point}$ *or* $\pi = \pi_{regular}$

Although, M. Yamashita [26] proves that regular pattern formation is solvable if the robots are *nonoblivious*, he poses the same problem by *oblivious* robots regardless of initial state as an open problem.

In a research by P. Widmayer [11] some possibility and impossibility results on pattern formation were given under the following assumptions:

- Autonomous
- Anonymous
- Homogeneous
- ***oblivious***
- Asynchronous, Synchoronized
- Unlimited Visibilty
- Dimensionless robots (point robots)

The results can be summarized as:

Theorem 2.0.14. *1. With common knowledge of two axes directions and orientations, the robots can form an arbitrary given pattern.*

2. With common knowledge on only one axis direction and orientation, the pattern formation problem is unsolvable when n is even; it can be solved if n is odd.

3. With common knowledge on only one axis direction, the robots can form an arbitrary pattern if n is odd.

4. With no common knowledge, the robots cannot form an arbitrary given pattern.

Theorem 2.0.15. *With common knowledge on only one axis direction and orientation, there exists no deterministic algorithm that allows an even number of robots to form an asymmetric pattern. Moreover, in this case they can*

only form symmetric patterns that have at least one symmetric axis not passing through a vertex of the input pattern.

Results on limited visibilty were also given in P. Widmayer [11]. These results can be summarized as:

Lemma 2.0.16. *If the visibility graph is disconnected, the pattern formation problem (or actually any problem) is unsolvable.*

Theorem 2.0.17. *There exists a deterministic algorithm that let the robots gather in one point in a finite number of movements, in the limited visibility setting and assuming common knowledge on direction and orientation of both axes.*

Up until the work of Czyzowicz et al. [4], the gathering problem was considered only under the assumption that robots are a point on the plane and are transparent, that is, a robot can see through another robot. These assumptions do not reflect reality, as real robots are not points, but instead they have a physical extent. Furthermore, robots are not transparent, that is, robots may block the view of other robots or robots may collide. Having this in mind, Czyzowicz et al. [4] initiated the study of the gathering problem with fat robots, that is, non-transparent unit-disks (disks of radius of 1 unit). As fat robots cannot occupy the same space on the plane, the gathering problem can no longer require robots to gather at the same point. Instead, per [4], gathering fat robots means forming a configuration for which the union of all discs representing them is connected. The model considered in [4] is the following: Robots operate in Look-Compute-Move cycles, they are identical, anonymous, history oblivious, non-transparent, and fat. They do not share a common coordination system and the only means of coordination is by vision; robots have unlimited visibly, unless their view is obstructed

by another robot. An asynchronous setting is considered, modeled by an adaptive adversary that can stop any robot for finite time, control the "speed" of any robot or cause robots moving into intersecting trajectories to collide. Under this model, the authors present solutions for the gathering problem for three and four robots. The proposed solutions consider exhaustively all possible classes of configurations in which robots may be found; a different gathering strategy corresponds to each possible case. As this approach cannot be generalized for larger number of robots (the cases grow exponentially as the number of robots increases), the authors left open the question of whether it is possible to solve gathering for any collection of $n \geq 5$ fat robots.

Work by C. Agathangelou [1] extended the work of Czyzowicz et al. [4] by giving a solution for $n \geq 5$ fat robots. With the additional assumption of *chirality*, C. Agathangelou [1] presented a distributed algorithm for the gathering problem for any number n of fat robots. Also, a solution to circle formation problem by *transparent fat* robots was given by S. Datta [5].

Chapter 3

Uniform Circle Formation

Uniform circle Formation problem can be divided into two subparts as follows:

1. Forming a circle, possibly an non-uniform one. (Circle Formation Problem)

2. Positioning the robots evenly on the boundary, i.e., forming a regular polygon. (Uniform Transformation)

The algorithms to solve the above two problems have been given separately in the upcoming sections. Starting with the definitions and notations required, algorithm for circle formation and uniform transformation are given.

A deterministic algorithm to form a non-uniform circle is provided. Unfortunately, the uniform transformation problem is only solved as a "convergence problem". But, assuming some constraints and power on the robots, the uniform transformation problem can be solved deterministically in finite time. These assumptions are:

- Robots have a common sense of "rotational" orientation. (chirality)

- The two oblivious algorithms presented separately assume the following:

    1. There is one faulty robot, i.e., a robot which cannot move when the robots are on the boundary of a circle.
    2. Robots can only move in the anti-clockwise direction. (All robots will move in the same direction as they have a common sense of orientation).

3.1 Definitions and Notations

Some definition and notations used in the presentation of the algorithm.

Definition 3.1.1. *Position.* Given a robot r_i, $p_i(t)$ denotes its position at time t, according to some global x-y coordinate system, and $p_i(0)$ is its initial position. P $(t) = p_i$ (t) $1 \leq$ i $\leq n$ denotes the multiset of the positions of all robots at time t. When this is not ambiguous, we sometimes mention a robot, implicitly referring to its position rather than the robot itself.

Definition 3.1.2. *Voronoi Diagram.* The Voronoi diagram Voronoi(P) of a set of points P $= p_1, p_2, ..., p_n$ is a subdivision of the plane into n cells, one for each point in P. The cells have the property that a point q belongs to the Voronoi cell of point p_i, denoted $V_{cell_{p_i}}$ (P), if and only if, for any other point $p_j \in$ P, dist(q, p_i) ¡ dist(q, p_j), where dist(p, q) is the Euclidean distance between p and q. In particular, the strict inequality means that points located on the boundary of the Voronoi diagram do not belong to any Voronoi cell. A Voronoi diagram is for instance depicted on Figure 3.1. Significantly more details about Voronoi diagrams and their principal applications are surveyed by Aurenhammer [2].

Definition 3.1.3. *Smallest enclosing circle.* The smallest enclosing circle of a set of points P is denoted by SEC(P). It can be defined by either two opposite points, or by at least three points. The smallest enclosing circle is unique, and can be computed in O(n log n) [24].

3.2 Circle Formation

Assumptions taken:

- Autonomous
- Anonymous
- Homogeneous
- Oblivious
- Asynchronous, Synchoronized
- Unlimited Visibilty
- Dimensionless robots (point robots)
- *multiplicity*
- Initial positions are all distinct

In addition to the above assumptions, we impose some restriction on the movement of the robots to solve the problem. Doing so ensures that (1) no two robots occupy the same position simultaneously (Restr. 1), and that (2) the smallest circle enclosing all robots remains invariant (Restr. 24). The restrictions are (they are also shown as figures):

1. A robot always moves toward a point that is inside its Voronoi cell.

2. No robot ever moves beyond the boundary of the smallest circle enclosing all robots.

3. All robots located on the boundary of the smallest enclosing circle remain on that boundary.

4. Robots located on the circumference of the smallest enclosing circle do not move unless there are at least three such robots with distinct positions.

"Algorithm 1" given by A. Konagaya [7] for circle formation. The given oblivious algorithm deterministically solves the circle formation problem in a finite number of cycles.

Algorithm 1 Formation of an (arbitrary) circle (code executed by robot r_i)

function $\mathcal{A}_{circle}$(P, p_i)

if $p_i \in$ SEC(P) **then**
  stay still {r_i already on boundary.}
else
  if $V_{cell_{p_i}}(P) \bigcap$ SEC(P) $\neq \Phi$ **then**
    target $\leftarrow V_{cell_{p_i}}(P) \bigcup$ SEC(P) $\bigcup$ Voronoi(P - {p_i})
    move to target
  else
    compute points in $V_{cell_{p_i}}(P)$ closest to SEC(P). {Voronoi cell of p_i inside circle}
    if exactly one candidate exists **then**
      move toward that point
    else
      select the coordinate with the greatest x-coordinate and then y-coordinate. {several candidates exists}
      move toward that point
    end if
  end if
end if

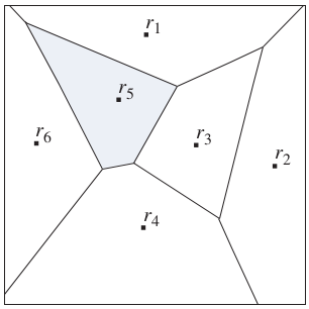

Figure 3.1: Restriction 1: The movement of r_5 is constrained by the interior of its voronoi cell

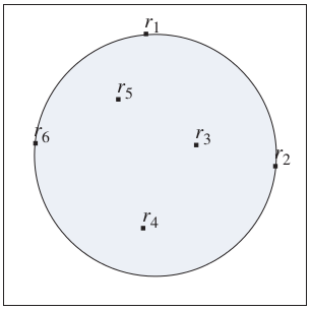

Figure 3.2: Restriction 2: The movement of r_5 is constrained by smallest enclosing circle

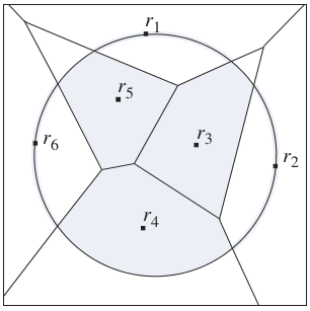

Figure 3.3: Restriction 1-4: r_1, r_2, r_6 move on the boundary of the enclosing circle (Restriction 3-4): r_3, r_4, r_5 move in interior

3.3 Uniform Transformation

The "Algorithm 2" given by A. Konagaya [7] converges to the uniform transformation. It may or may not ever.

Algorithm 2 Convergence towards a uniform transformation

function $\mathcal{A}_{uniform}$(P, p_i)

Require: Assume all robots are on the boundary of SEC(P)

1: prev(p_i) ← direct neighbour of p_i counterclockwise
2: next(p_i) ← direct neighbour of p_i clockwise
3: midpoint(p_i) ← midpoint of arc prev(p_i) & next(p_i)
4: target ← midpoint of midpoint(p_i)
5: move toward target.

Now the algorithms presented have been proposed for some special situations.

3.3.1 Situation 1

Assumming all the robots form a circle and suddenly exactly one of the robots stops moving (maybe it is faulty or intentional). Now, the given algorithm in "Algorithm 3" deterministically solves the uniform transformation problem in a finite number of cycles.

Proof. "Algorithm 3" deterministically solves the problem of uniform transformation. Consider the robot just prev of the "immobile" robot, say r_f. r_f robot will deterministically position itself in a finite number of cycles in its prescribed position as its neighbour robot is not moving, thus, fixing its target position for all the cycles. Now this r_f robot also becomes "immobile" as it is on its target position. Recursively considering the robot r_{f-1}, this will also deteministically position itself on the target position in a finite number of cycles. Similarly, all robots will move to their target position and will form

Algorithm 3 uniform transformation with one faulty robot

function $\mathcal{A}_{uniform}$(P, p_i)

Require: Assume all robots are on the boundary of SEC(P)

Ensure: One of the robots is faulty (cannot move)

if r_i is falty **then**
  This robot cannot move.
else
  next(r_i) ← direct neighbour of r_i clockwise
  n ← total number of robots
  requiredAngularDiff ← 2*π/n
  angularDiff ← (arc between next(r_i))/(radius of circle)
  if $requiredAngularDiff > angularDiff$ **then**
    target ← (requiredAngularDiff - angularDiff)*(radius of circle) in anticlockwise direction
    move toward target.
  else
    if $requiredAngularDiff < angularDiff$ **then**
      target ← (angularDiff - requiredAngularDiff)*(radius of circle) in clockwise direction
      move toward target
    else
      target ← currentPosition
      stand still.
    end if
  end if
end if

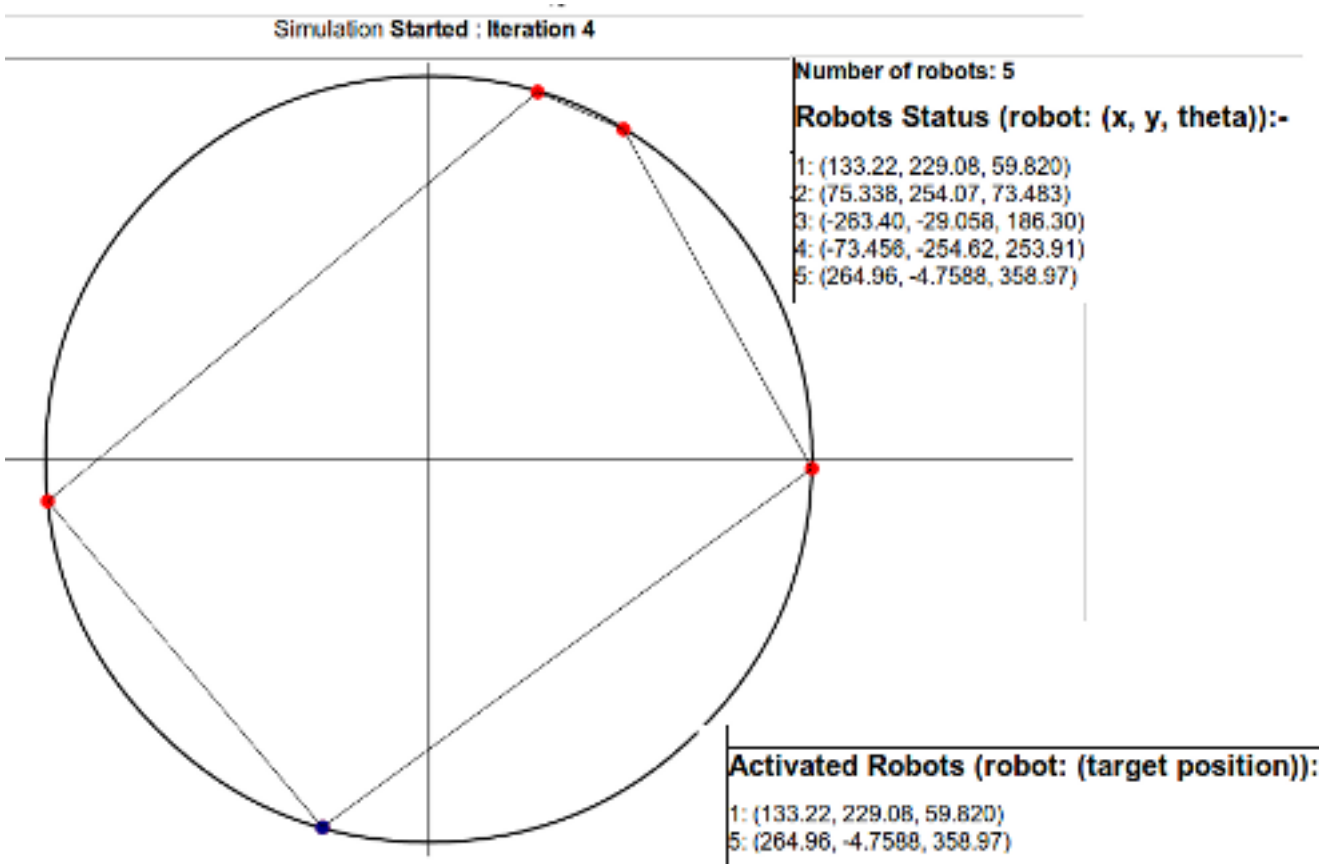

Figure 3.4: Initial states of all the robots. The faulty robots is shown in blue color (although it is anonymous in theory)

a regular-polygon solving the uniform transformation problem.

This algorithm was simulated, and as derived, the movement of the robots stopped after a finite number of cycles forming an uniform circle/regular polygon. This result is shown in the pictures.

□

3.3.2 Situation 2

Assumming all the robots form a circle and we restrict the movement in only counterclockwise direction. Then "Algorithm 4" deterministically solves the

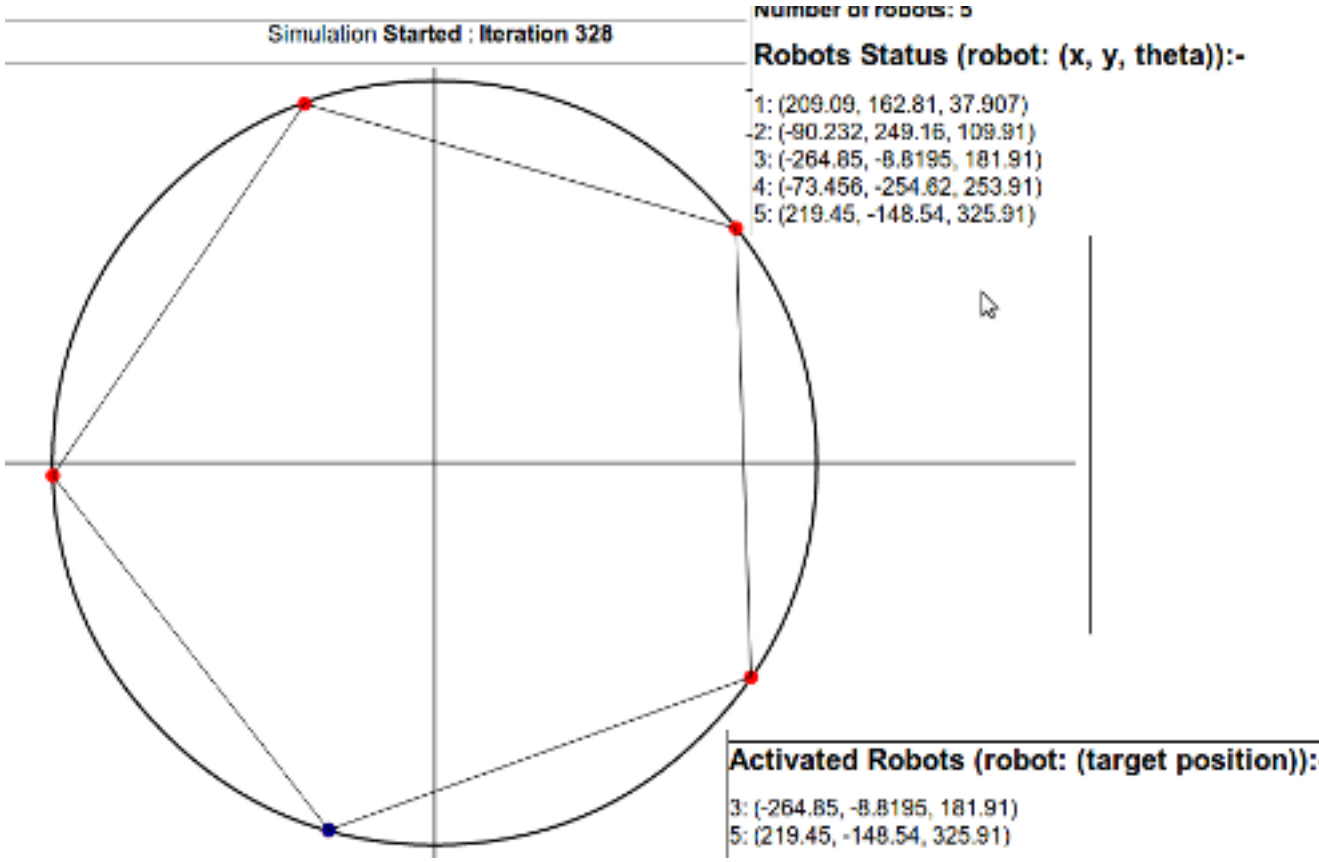

Figure 3.5: Final states of all the robots. The faulty robots is shown in blue color (although it is anonymous in theory). As shown, after a finite number of iterations, they form a regular polygon.

problem of uniform transformation.

Algorithm 4 uniform transformation with movement restriction in one direction

function $\mathcal{A}_{uniform}$(P, p_i)
Require: Assume all robots are on the boundary of SEC(P)
Ensure: Robots can only move counterclockwise
1: prev(p_i) ← direct neighbour of p_i counterclockwise
2: next(p_i) ← direct neighbour of p_i clockwise
3: midpoint(p_i) ← midpoint of arc prev(p_i) & next(p_i)
4: target ← midpoint of midpoint(p_i)
5: **if** target in counterclockwise direction **then**
6: Move the robot r_i to target
7: **else**
8: stand still.
9: **end if**

"Algorithm 4" deterministically solves the problem of uniform transformation. No rigorous proof hasn been worked out. But, this algorithm was simulated, and as in the diagrams shown, the movement of the robots stopped after a finite number of cycles forming an uniform circle/regular polygon.

3.4 Combining the two parts

Now the sub-parts constructed can be joined to make one solution to a bigger problem which is Uniform Circle Formation. This is shown in "Algorithm 5". When any robot is inside the circle, we run the algorithm for circle formation. If all the robots are on the boundary of the smallest enclosed circle, it will run one of the algorithms from 2, 3, 4 depending on the situation.

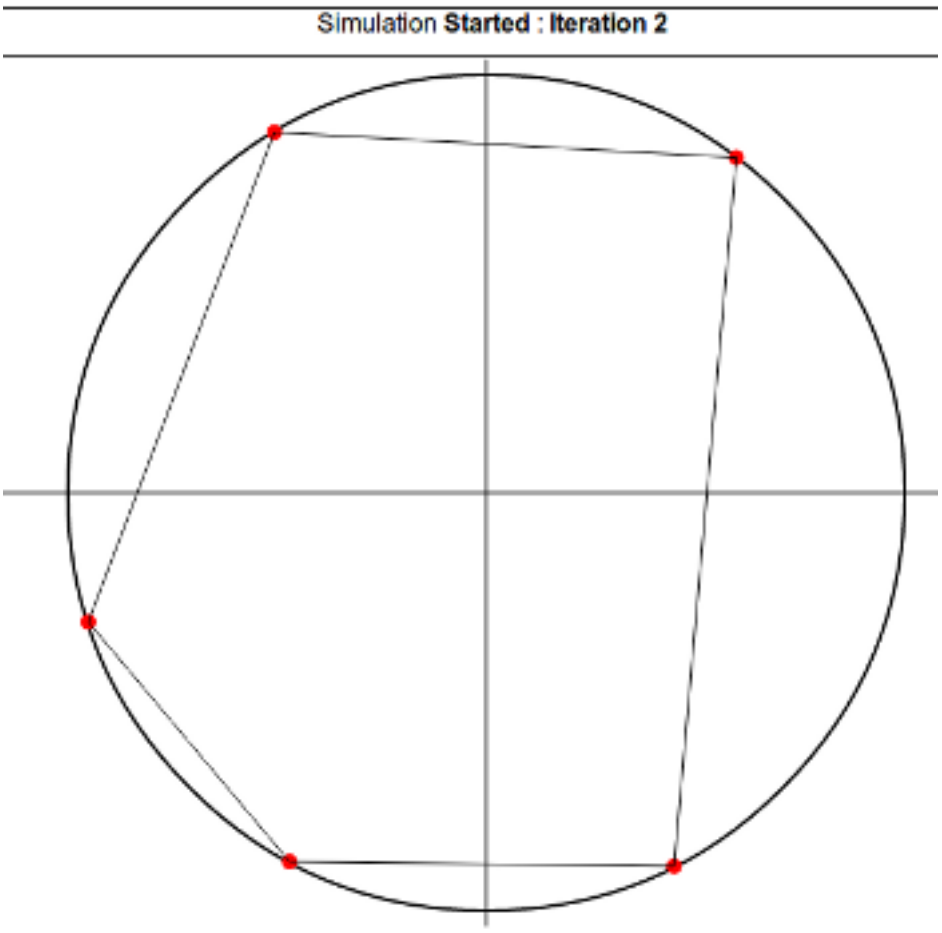

Figure 3.6: Initial states of all the robots. Algorithm 4.

Algorithm 5 uniform circle formation (Combining the parts)

function $\mathcal{A}_{uniform-circle}$(P, p_i)

1: **if** p_i is in the interior of SEC(P) **then**
2: run Algorithm 1 $\{\mathcal{A}_{circle}\}$(P, p_i)
3: **else**
4: Algorithm 2, 3, 4 depending on the situation {All robots are on SEC(P)} $\{\mathcal{A}_{uniform}\}$(P, p_i)
5: **end if**

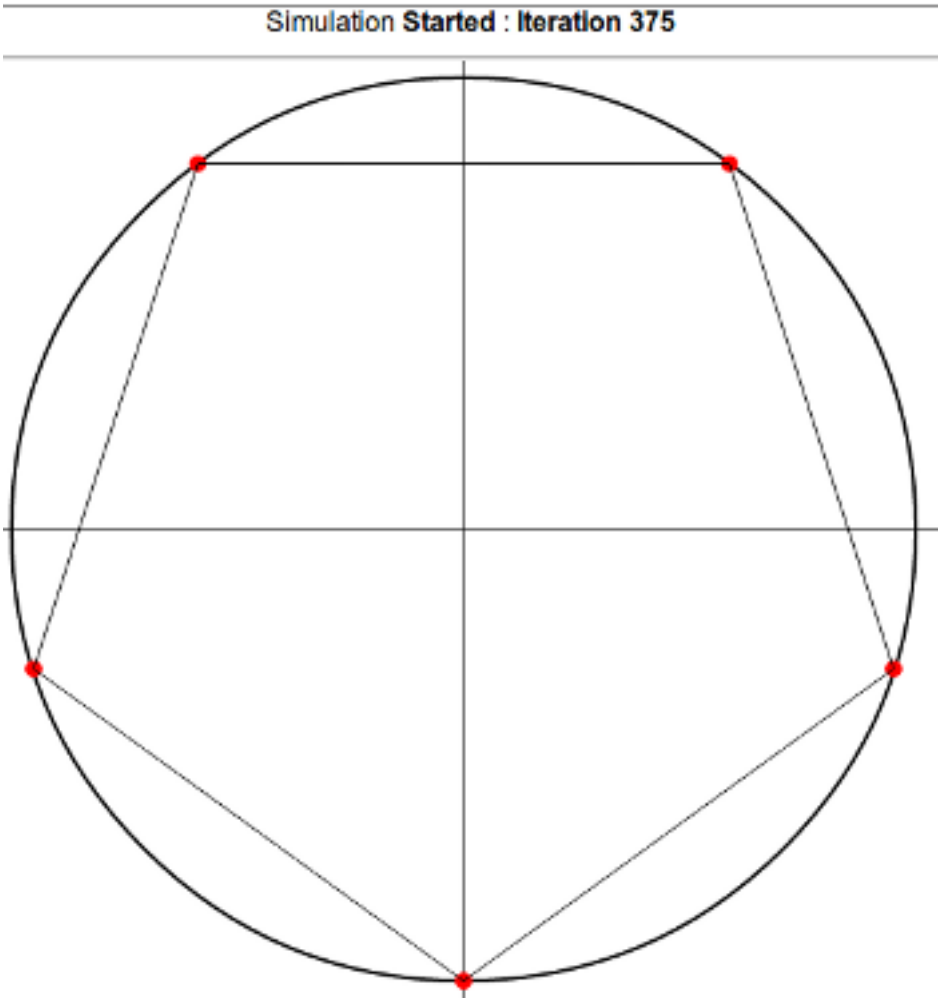

Figure 3.7: Final states of all the robots. Algorithm 4. They form a regular polygon.

Chapter 4

Minimum Perimeter Circle Formation

4.1 Problem Definition

In this chapter we address the problem of minimum perimeter circle formation. The problem statement is as follows:

Definition 4.1.1. ***Minimum Perimeter Circle Formation***. Given a group of n fat robots $\{r_1, r_2, ..., r_n\}$ with distinct positions and located arbitrarily on a plane, eventually arrange them at a boundary of a circle such that the perimeter is minimum and the final configuration is connected and terminal.

We will solve the problem using fat robots (closed unit-discs).

4.2 Underlying Model

As we know the number of robots are n, thus, the radius required to accomodate all the robots is: $\mathbf{Rad} = n/\pi$

We use the basic structure of *weak model* of robots and add some extra features which extend the model towards reality. Let $R = \{r_1, r_2, ..., r_n\}$ be a set of fat robots. A robot is represented by its center, i.e., by r_i we mean a robot whose center is c_i . The set of robots R deployed on the 2-D plane is described as follows:

- A robot can see up to a infinite distance around itself on the 2D plane.
- Robots are *transparent*, that is, every robot can see through every other robot. Thus, every robots is visible.
- The robots are autonomous.
- The robots do not have a common origin, no common x-y axis, but common sense of direction (chirality).
- Robots have a common unit of distance (as all robots are unit-discs, this can easily be assumed)
- Robots are anonymous and homogeneous in the sense that they are unable to uniquely identify themselves, neither with a unique identification number nor with some external distinctive mark (e.g. color, flag).
- The robots are oblivious in the sense that they can not remember any past action.
- A robot is a fat robot and represented as a unit disc.
- ***CORDA*** model is assumed for the robots. Under this model a robot in motion is visible to other robots.

- Robots can not communicate explicitly. The robots communicate only by means of observing other robots within its visibility range.
- Each robot executes a cycle of *look - compute - move synchronously.*
- One robot acts as an obstacle to another robot, that is, two robots cannot share more than one point on the 2D plane (The point is shared only when one robot "touches" another).
- As it is a synchronous model, at every step, robot can move atmost ϵ distance.

4.3 Overview

Let R be a set of stationary transparent fat robots, under the computation model described above. The objective is to move the robots in R in order to form a circle of radius Rad. First, the robots compute the Smallest Enclosing Circle (SEC) with R. Let P be the center of the SEC. Let $C(1)$ be the set of robots nearest to P. We call this set, the robots at 1^{st} level of distances. Similarly $C(2)$ is the set of robots 2^{nd} nearest to P . We call this set, the robots at 2^{nd} level of distances. Let there be m such levels of distances. The robots which are at i^{th} level of distance from P constitute the set $C(i)$ $(1 \leq i \leq m)$. The robots in $C(m)$ are actually on the SEC. We can visualize the configuration of robots as m concentric circles whose center is P. If any robot is at the center then it is considered to be on a circle with radius zero and denoted by $C(0)$. This set of circles is arranged according to their distances from P as $C = \{C(0), C(1), C(2), ..., C(m)\}$. Since the SEC is unique, the set C is also unambiguous. Each robot computes this sequence C. Now let us assume another concentric circle with radius Rad and call it

$C(Rad)$ which is actually the circle where the robots will lie in the terminal configuration.

We first form a circle by synchronous transparent fat robots. Then, when the circle is complete, the robots move in such a way that the circle starts to converge. When the robots form one big connected component with the least perimeter of the underlying circle, we say that the robots have deterministically solved the minimum perimeter circle formation problem. In the next section, you'll find all the algorithms to deterministically solve the above problem. Every robot calls method *MinimumPerimeterCircleFormation(r)* to form a minimum perimeter circle.

Let us introduce some notations and definition which will be used to explain the algorithm.

- $|A|$: Number of elements in set A.
- $Dist(A, B)$: The euclidean distance between two points A and B.
- T_r : Computed destination for r.
- $LN(r, i)$: i^{th} left (clockwise w.r.t. r) neighbor of the robot r on the circumference of the circle on which r lies.
- $RN(r, i)$: i^{th} right (anti clockwise w.r.t. r) neighbor of the robot r on the circumference of the circle on which r lies.
- $Poly(C(k))$: The convex polygon formed by the robots on the circumference of the circle $C(k) \in C$.
- $Projpt(r, C(k))$: The projected (radially inward or outward) point of the robot position r on the concentric circle $C(k) \in C$.

- $Maxe(Poly(C(k)))$: The unique longest edge of $Poly(C(k))$. (If there are more than one longest edge, then the next maximum length is found until we get a single edge of maximum length.) If R is not in symmetric configuration, then a longest unique edge is positively found.
- Component: set of robots touching each other. There can be multiple components in a configuration.
- CurrentSEC: The smallest enclosing circle.
- RequiredSEC: Circle with radius Rad.
- CurrentRadius: Radius of currentSEC.

Definition 4.3.1. ***Free Robot***. Let r be a robot on the circle $C(mi)$ $(1 \leq i \leq m1)$. r is projected (radially outward) on the circle $C(m)$. Let r' be the projected point. Let us denote the rectangular area with length $l = Dist(r, r')$ and width 2 as $rect(rr')$. r is said to be a *free robot* if $rect(rr')$ does not contain any part of other robot. A robot which is not free is called *locked robot*.

4.4 Algorithm

Algorithm 6 SECexpansion(r) - Expands the current SEC to make space for all the robots and also helps to get out of lock configuration.

```
1: if r ∈ C(m) then
2:     Move r ϵ distance radially outward from the center of current SEC.
3: else
4:     r does not move.
5: end if
```

Algorithm 7 CheckLock(r, T_r) - Checks if moving the robot r to position T_r will lead to a lock configuration or not.

1: **if** (m == 2) $\bigwedge$ (vacant points on $C(m)$ == Robots on $C(m-1)$) $\bigwedge$ ($\forall r_i \in C(m-1), r_i$ is not a free robot) $\bigwedge$ (r at T_r forms a symmetric configuration) **then**
2: return *true*;
3: **else**
4: return false;
5: **end if**

Algorithm 8 ComputeDestination(r) - Computes the destination on the SEC for robots belonging to $C(m-1)$ for the initial formation of circle.

Require: $r \in C(m-1)$

FoundDestination $\leftarrow$ *false*
if (FoundDestination == *false*) $\bigvee$ (($m == 1$) $\bigwedge$ ($|C(0)| = 1$)) **then**
  Compute $Poly(C(m))$; $e \leftarrow Maxe(Poly(C(m)))$; l $\leftarrow$ bisector of e;
  $u \leftarrow$ intersection point of l and $C(m)$ at that side of e where r lies;
  $T_r \leftarrow u$
  FoundDestination $\leftarrow$ *true*
else
  if $Projpt(r, C(m))$ is a vacant point **then**
    $T_r \leftarrow Projpt(r, C(m))$; FoundDestination $\leftarrow$ *true*
  else
    if ($m == 2$) $\bigwedge$ ($|C(1)| == 1$) $\bigwedge$ ($Projpt'(r, C(m))$ is a vacant point) **then**
      $T_r \leftarrow Projpt'(r, C(m))$; FoundDestination $\leftarrow$ *true*; $\{\}Projpt'(r, C(m))$ is the diametrically opposite point to $Projpt(r, C(m))$
    end if
    $r_{pt} \leftarrow Projpt(r, C(m))$; $d \leftarrow Projpt'(r, C(m))$;
    if FoundDestination == *false* **then**
      $i \leftarrow 1$; R $\leftarrow RN(r_{pt}, i)$; L $\leftarrow LN(r_{pt}, i)$;
      while $Dist(r_{pt}, R) == Dist(r_{pt}, L)$ **do**
        i++; R $\leftarrow RN(r_{pt}, i)$; L $\leftarrow LN(r_{pt}, i)$;
      end while
      if $Dist(r_{pt}, R) > Dist(r_{pt}, L)$ **then**
        $T_r \leftarrow$ the first vacant position at right side (anti-clockwise) of r_{pt} between r_{pt} and d; FoundDestination $\leftarrow$ *true*
      else
        $T_r \leftarrow$ the first vacant position at left side (clockwise) of r_{pt} between r_{pt} and d; FoundDestination $\leftarrow$ *true*
      end if
    end if

26: **if** FoundDestination == *false* **then**
27: $i \leftarrow 1$; R $\leftarrow RN(r, i)$; L $\leftarrow LN(r, i)$;
28: **while** $Dist(r, R) == Dist(r, L)$ **do**
29: i++; R $\leftarrow RN(r, i)$; L $\leftarrow LN(r, i)$;
30: **end while**
31: **if** $Dist(r, R) > Dist(r, L)$ **then**
32: $T_r \leftarrow$ the first vacant position at right side (anti-clockwise) of r_{pt} between r_{pt} and d; FoundDestination $\leftarrow$ *true*
33: **else**
34: $T_r \leftarrow$ the first vacant position at left side (clockwise) of r_{pt} between r_{pt} and d; FoundDestination $\leftarrow$ *true*
35: **end if**
36: **end if**
37: **if** FoundDestination == *false* **then**
38: $k \leftarrow 1$;
39: **while** (FoundDestination == *false*) $\bigwedge$ $(k < m)$ **do**
40: $r'_{pt} \leftarrow Projpt(r, C(m - k))$;$i \leftarrow 2$; R $\leftarrow RN(r'_{pt}, i)$; L $\leftarrow LN(r'_{pt}, i)$;
41: **while** $Dist(r'_{pt}, R) == Dist(r'_{pt}, L)$ **do**
42: i++; R $\leftarrow RN(r'_{pt}, i)$; L $\leftarrow LN(r'_{pt}, i)$;
43: **end while**
44: **if** $Dist(r'_{pt}, R) > Dist(r'_{pt}, L)$ **then**
45: $T_r \leftarrow$ the first vacant position at right side (anti-clockwise) of r_{pt} between r_{pt} and d; FoundDestination $\leftarrow$ *true*
46: **else**
47: $T_r \leftarrow$ the first vacant position at left side (clockwise) of r_{pt} between r_{pt} and d; FoundDestination $\leftarrow$ *true*
48: **end if**
49: **if** FoundDestination == *false* **then**
50: k++;
51: **end if**
52: **end while**
53: **end if**
54: **if** FoundDestination == *false* **then**
55: Draw tangent τ at r;
56: τ intersects $C(m)$ at point L_p and R_p;
57: $T_r \leftarrow L_p$; or $T_r \leftarrow R_p$;
58: **end if**
59: **end if**
60: **end if**
61: return T_r;

Algorithm 9 Converge(r) - After a circle is formed, this algorithm converges the robot to minimum perimeter circle.

Require: All robots are on SEC

if CurrentSEC == RequiredSEC **then**
  Do nothing; return;
else
  if Any robot is tangent to any other robot **then**
    if robot r is the *rightmost*(anti-clockwise) of the component it belongs to **then**
      $d \leftarrow ((2 * \pi * CurrentRadius) - (2 * n))/n$
      $T_r \leftarrow$ Point on SEC at distance d from r's right neihbour.
      Move r to T_r (Only atmost ϵ distance will covered on the SEC)
    end if
  else
    $d \leftarrow$ maximum distance all robots can move radially inwards until any one robot becomes tangent to another robot.
    $d_{final} = minimum(\epsilon, d)$;
    Move distance d_{final} radially inwards.
  end if
end if

Algorithm 10 MinimumPerimeterCircleFormation(r) - This is the main method called by every robot in it's computing phase which in turn uses the above algorithms to perform it's task.

```
1: if All Robots on SEC then
2:    Call Converge(r);
3: else
4:    if CurrentSEC < RequiredSEC then
5:       Call SECexpansion(r);
6:    else
7:       if r ∈ C(m − 1) then
8:          T_r ← ComputeDestination(r)
9:          if (CheckLock(r, T_r)) ⋁ (Line joining r and T_r intersect C(m − 1)
    or any other robot) then
10:            Call SECexpansion(r);
11:         else
12:            r moves to T_r
13:         end if
14:      else
15:         r does not do anything.
16:      end if
17:   end if
18: end if
```

4.5 Correctness

We'll prove that every algorithm achieves it task and finally the method MinimumPerimeterCircleFormation(r) forms the minimum perimeter circle.

Lemma 4.5.1. *The radius of CurrentSEC is never decreased by the execution of SECexpansion(r). Also, it will not infinitely expand the SEC.*

Proof. According to the algorithm SECexpansion(r), a robot either moves away from the center of the currentSEC or stay still. Hence, the radius of the circle either increases or remains same. Also, it is only called a limited number of times as the condition to call the method is:

- Radius of current SEC is smaller than the required SEC. Thus, as soon as the current SEC is greater than the required SEC, the method will not be called anymore.
- There is a lock configuration, i.e., the inner robots may not be able to find a destination on the SECbecause of limited space and symmetry. When the SEC is big enough, the condition that robots on inner circle is equal to the vacant points on the SEC will not be true and the method will not be called anymore.

Thus, the method SECexpansion(r) is callled only a limited number of times and will never lead to infinite expansion of SEC. □

Definition 4.5.2. ***Eligible Robot*** [5]. A robot is called an eligible robot, if it finds a unique destination (using ComputeDestination(r)) on the circumference of SEC.

Lemma 4.5.3. *ComputeDestination(r) finds a unique destination for every robot.*

Proof. Every robot in R is an eligible robot as the circumference of SEC will be greater than the space required to accomodate all the robots. □

Lemma 4.5.4. *Converge(r) reduces the radius of the current SEC and terminates after a finite number of steps.*

Proof. converge(r) moves the robots radially inwards. Also, all the robots move at the same time and the same distance because of the synchronization assumption and the fact that all the robots are on the SEC. Thus, the radius of the current SEC decreases with execution of converge(r). As after a point of time when we reach the RequiredSEC, converge(r) will not be able to move the robots inside anymore and thus will terminate. □

Theorem 4.5.5. *MinimumPerimeterCircleFormation(r) forms a circle with minimum perimeter after a finite time.*

Proof. According to the lemmas above 4.5.1, 4.5.3 and 4.5.4, every robot finds a place on the SEC in a finite number of steps and thus, all the fat-robots form a circle. When the circle is formed, MinimumPerimeterCircleFormation(r) in-turn calls the converge(r) method which converges the circle to required-SEC in finite number of steps. Thus, a circle with minimum perimeter is formed. □

Chapter 5

Conclusion

Results of the previous semester can be summarized as follows:

- Results from references were collected about the possibilities and impossibilities of the solutions for sub problems in pattern formation by autonomous, anonymous, oblivious/non-oblivious and homogeneous robots.
- Uniform Circle Formation problem can be sub-divided into two parts(Circle Formation and Uniform Transformation problems) and they can be solved separately to get the solution to the bigger problem.
- Circle Formation Problem can be solved for n $\geq$ 3 robots in a finite number of cycles with oblivious robots.
- Currently, no deterministic algorithm for oblivious, anonymous mobile robots exist to solve the Uniform transformation problem which terminates in a *finite number of cycles*.
- Taking a stronger assumption on the model makes the Uniform transformation problem solvable by oblivious robots in a finite number of

cycles.

- ***First assumption***: Exactly one robot on the circle is "immobile". Now, the Uniform transformation problem is solvable by oblivious robots in a finite number of cycles by the algorithm *Algorithm 3* for $n \geq 2$.

- ***Second assumption***: Robots can only move in one direction (anti-clockwise). Again, the Uniform transformation problem is solvable by oblivious robots in a finite number of cycles by the algorithm *Algorithm 3* for $n \geq 3$.

The project was further extended to fat-robots trying to solve the minimum perimeter circle formation problem. A determinitic solution was given for the problem in a synchronized CORDA model with transparent fat-robots having an additional assumption of chirality.

Bibliography

[1] Chrysovalandis Agathangelou, Chryssis Georgiou, and Marios Mavronicolas. A distributed algorithm for gathering many fat mobile robots in the plane. In *Proceedings of the 2013 ACM Symposium on Principles of Distributed Computing*, PODC '13, pages 250–259, New York, NY, USA, 2013. ACM.

[2] F. Aurenhammer. Voronoi diagramsa survey of a fundamental geometric data structure. *ACM Comput. Surv*, 23(3):345–405, 1991.

[3] Klmn Bolla, Tams Kovacs, and Gbor Fazekas. Gathering of fat robots with limited visibility and without global navigation. In Leszek Rutkowski, Marcin Korytkowski, Rafa Scherer, Ryszard Tadeusiewicz, LotfiA. Zadeh, and JacekM. Zurada, editors, *Swarm and Evolutionary Computation*, volume 7269 of *Lecture Notes in Computer Science*, pages 30–38. Springer Berlin Heidelberg, 2012.

[4] Jurek Czyzowicz, Leszek Gasieniec, and Andrzej Pelc. Gathering few fat mobile robots in the plane. *Theoretical Computer Science*, 410(67):481 – 499, 2009. Principles of Distributed Systems.

[5] Suparno Datta, Ayan Dutta, Sruti Gan Chaudhuri, and Krishnendu Mukhopadhyaya. Circle formation by asynchronous transparent fat

robots. In Chittaranjan Hota and PradipK. Srimani, editors, *Distributed Computing and Internet Technology*, volume 7753 of *Lecture Notes in Computer Science*, pages 195–207. Springer Berlin Heidelberg, 2013.

[6] X. A. Debest. Remark about self-stabilizing systems. *Commun. ACM*, 38(2):115–117, 1995.

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

[8] Stphane Devismes, Anissa Lamani, Franck Petit, Pascal Raymond, and Sbastien Tixeuil. Optimal grid exploration by asynchronous oblivious robots. In AndraW. Richa and Christian Scheideler, editors, *Stabilization, Safety, and Security of Distributed Systems*, volume 7596 of *Lecture Notes in Computer Science*, pages 64–76. Springer Berlin Heidelberg, 2012.

[9] M Bernardine Dias and Anthony (Tony) Stentz. A free market architecture for distributed control of a multirobot system. In *6th International Conference on Intelligent Autonomous Systems (IAS-6)*, pages 115–122, July 2000.

[10] Ayan Dutta, Sruti Gan Chaudhuri, Suparno Datta, and Krishnendu Mukhopadhyaya. Circle formation by asynchronous fat robots with limited visibility. In R. Ramanujam and Srini Ramaswamy, editors, *Distributed Computing and Internet Technology*, volume 7154 of *Lecture*

Notes in Computer Science, pages 83–93. Springer Berlin Heidelberg, 2012.

[11] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Distributed coordination of a set of autonomous mobile robots. In *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*, pages 480–485, 2000.

[12] Paola Flocchini, David Ilcinkas, Andrzej Pelc, and Nicola Santoro. Computing without communicating: Ring exploration by asynchronous oblivious robots. *Algorithmica*, 65(3):562–583, 2013.

[13] Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Peter Widmayer. Hard tasks for weak robots: The role of common knowledge in pattern formation by autonomous mobile robots. In *Algorithms and Computation*, volume 1741 of *Lecture Notes in Computer Science*, pages 93–102. Springer Berlin Heidelberg, 1999.

[14] Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Peter Widmayer. Gathering of asynchronous oblivious robots with limited visibility. In Afonso Ferreira and Horst Reichel, editors, *STACS 2001*, volume 2010 of *Lecture Notes in Computer Science*, pages 247–258. Springer Berlin Heidelberg, 2001.

[15] Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Peter Widmayer. Gathering of asynchronous robots with limited visibility. *Theoretical Computer Science*, 337(13):147 – 168, 2005.

[16] Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Peter Widmayer. Arbitrary pattern formation by asynchronous, anonymous, oblivious robots, 2008.

[17] Nao Fujinaga, Yukiko Yamauchi, Shuji Kijima, and Masafumi Yamashita. Asynchronous pattern formation by anonymous oblivious mobile robots. In MarcosK. Aguilera, editor, *Distributed Computing*, volume 7611 of *Lecture Notes in Computer Science*, pages 312–325. Springer Berlin Heidelberg, 2012.

[18] Vincenzo Gervasi and Giuseppe Prencipe. Coordination without communication: the case of the flocking problem. *Discrete Applied Mathematics*, 144(3):324 – 344, 2004. Fun with Algorithms 2.

[19] I. Suzuki H. Ando, Y. Oasa and M. Yamashita. Distributed memoryless point convergence algorithm for mobile robots with limited visibility. *EEE Trans. on Robotics and Automation*, 15(5):818–828, 1999.

[20] Branislav Katreniak. Biangular circle formation by asynchronous mobile robots. In Andrzej Pelc and Michel Raynal, editors, *Structural Information and Communication Complexity*, volume 3499 of *Lecture Notes in Computer Science*, pages 185–199. Springer Berlin Heidelberg, 2005.

[21] Ralf Klasing, Euripides Markou, and Andrzej Pelc. Gathering asynchronous oblivious mobile robots in a ring. In Tetsuo Asano, editor, *Algorithms and Computation*, volume 4288 of *Lecture Notes in Computer Science*, pages 744–753. Springer Berlin Heidelberg, 2006.

[22] J.-B. Billeter M. J. B. Krieger and L. Keller. Ant-like task allocation and recruitment in cooperative robots. *Nature*, 406(6799):992–995, 2000.

[23] Giuseppe Prencipe. Impossibility of gathering by a set of autonomous mobile robots. *Theoretical Computer Science*, 384(23):222 – 231, 2007. Structural Information and Communication Complexity (SIROCCO 2005).

[24] S. Skyum. A simple algorithm for computing the smallest enclosing circle. *Information Processing Letters*, 37(3):121–125, 1991.

[25] K. Sugihara and I. Suzuki. Distributed algorithms for formation of geometric patterns with many mobile robots. *Journal on Robotics Systems*, 3(13):127–139, 1996.

[26] Ichiro Suzuki and Masafumi Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing*, 28:1347–1363, 1999.

[27] A. S. Fukunaga Y. Uny Cao and A. B. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, (4):1–23, 1997.