Pattern Formation by Robots with Inaccurate Movements

Kaustav Bose[1]
Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Kolkata, India

Archak Das
Department of Mathematics, Jadavpur University, Kolkata, India

Buddhadeb Sau
Department of Mathematics, Jadavpur University, Kolkata, India

Abstract

Arbitrary Pattern Formation is a fundamental problem in autonomous mobile robot systems. The problem asks to design a distributed algorithm that moves a team of autonomous, anonymous and identical mobile robots to form any arbitrary pattern F given as input. In this paper, we study the problem for robots whose movements can be inaccurate. Our movement model assumes errors in both direction and extent of the intended movement. Forming the given pattern exactly is not possible in this setting. So we require that the robots must form a configuration which is close to the given pattern F. We call this the Approximate Arbitrary Pattern Formation problem. With no agreement in coordinate system, the problem is unsolvable, even by fully synchronous robots, if the initial configuration 1) has rotational symmetry and there is no robot at the center of rotation or 2) has reflectional symmetry and there is no robot on the reflection axis. From all other initial configurations, we solve the problem by 1) oblivious, silent and semi-synchronous robots and 2) oblivious, asynchronous robots that can communicate using externally visible lights.

2012 ACM Subject Classification Theory of computation → Distributed algorithms; Theory of computation → Computational geometry; Computing methodologies → Distributed algorithms; Computing methodologies → Robotic planning

Keywords and phrases Distributed Algorithm, Mobile Robots, Movement Error, Approximate Arbitrary Pattern Formation, Look-Compute-Move, Minimum Enclosing Circle

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2021.10

Related Version *Full Version*: `https://arxiv.org/abs/2010.09667`

Funding *Archak Das*: supported by University Grants Commission, India.

Acknowledgements We would like to thank the anonymous reviewers for their comments and suggestions which helped us to improve the presentation of the paper.

1 Introduction

A robot swarm is a distributed system of autonomous mobile robots that collaboratively execute some complex tasks. Distributed coordination of robot swarms has attracted considerable research interest. Early investigations of these problems were experimental in nature with the main emphasis being on designing good heuristics. However, the last two decades have seen a series of theoretical studies on the computability and complexity issues related to distributed computing by robot swarms. These studies are aimed at providing provably correct algorithmic solutions to fundamental coordination problems. The robots are assumed to be *anonymous* (they have no unique identifiers that they can use in a

[1] This work was done when the author was at Jadavpur University, Kolkata, India.

computation), *homogeneous* (they execute the same distributed algorithm) and *identical* (they are indistinguishable by their appearance). The robots do not have access to any global coordinate system. They have either no memory or very little memory available to remember past observations and calculations. Also, they either have no means of direct communication or have some very weak communication mechanism (e.g., an externally visible light that can assume a small number of predefined colors). The model assumes robots with such weak features because the theoretical studies usually intend to find the minimum capabilities necessary for the robots to solve a given problem. The objective of this approach is to obtain a clear picture of the relationship between different features and capabilities of the robots (such as memory, communication, sensing, synchronization, agreement among local coordinate systems etc.) and their exact role in solvability of fundamental problems. Adopting such restrictive model also makes sense from a practical perspective since the individual units of robot swarms are low-cost generic robots with limited capabilities. Although certain assumptions, such as obliviousness (having no memory of past observations and calculations), may seem to be overly restrictive even for such weak robots, there are specific motivations for these assumptions. For example, the assumption of oblivious robots ensures self-stabilization. This is because any algorithm that works correctly for oblivious robots is inherently self-stabilizing as it tolerates errors that alter the local states of the robots. While the robots are assumed to be very weak with respect to memory, communication etc., certain aspects of the model are overly strong. In particular, the assumed mobility features of the robots are very strong. Two standard models regarding the movement of the robots are Rigid and Non-Rigid. In Rigid, if a robot x wants to go to any point y, then it can move to exactly that point in one step. This means that the robots are assumed to be able to execute error-free movements in any direction and by any amount. Certain studies also permit the robots to move along curved trajectories. The algorithms in this model rely on the accurate execution of the movements and are not robust to movement errors that real life robots are susceptible to. Furthermore, the error-free movements of the robots have surprising theoretical consequences as shown in the remarkable results obtained in [11]. A "positional encoding" technique was developed in [11] that allows a robot, that has very limited or no memory to store data, to implicitly store unbounded amount of information by encoding the data in the binary representation of its distance from another robot or some other object, e.g., the walls of the room inside which it is deployed. Exact movements allow the robots to preserve and update the data. This gives the robots remarkable computational power that allows them to solve complex problems which appear to be unsolvable by robots with limited or no memory, e.g., constructing a map of a complex art gallery by an oblivious robot. Obviously these techniques are impossible to implement in practice. Also, for problems that we expect to be unsolvable by real life robots with certain restrictions in memory, communication etc., it may become difficult or impossible to theoretically establish a hardness or impossibility result due to the strong model. The Non-Rigid model assumes that a robot may stop before reaching its intended destination. However, $\exists$ a constant $\delta > 0$ such that if the destination is at most δ apart, the robot will reach it; otherwise, it will move towards the destination by at least δ. Notice that in the Non-Rigid model, 1) the movement is still error-free if the destination is close enough, i.e., within δ, and 2) there is no error whatsoever in the direction of the movement even if the destination is far away. In [1], it was shown that these two properties allow robots to implement positional encoding even in the Non-Rigid model. This motivates us to consider a new movement model allowing inaccurate movements.

We consider a movement model that assumes errors in both direction and extent of the intended movement. Also, the errors can occur no matter what the extent of the attempted movement is. In this model, we study the Arbitrary Pattern Formation problem.

Arbitrary Pattern Formation is a fundamental robot coordination problem that has been extensively studied in the literature [12, 14, 9, 8, 6, 13, 5, 10, 15, 2, 4]. The objective of the problem is to design a distributed algorithm that allows the robots to form any pattern F given as input. This problem is well-studied in the literature in the Rigid and Non-Rigid model. However, the techniques used in these algorithms are not readily portable in our setting. For example, in most of these algorithms, the minimum enclosing circle of the configuration plays an important role. The center of the minimum enclosing circle is set as the origin of the coordinate system with respect to which the pattern is to be formed. So the minimum enclosing circle is kept invariant throughout the algorithm. The robots inside the minimum enclosing circle move to form the part of the pattern inside the circle, without disturbing it. For the pattern points on the minimum enclosing circle, robots from the inside may have to move on to the circle. Also, the robots on the minimum enclosing circle, in order to reposition themselves in accordance with the pattern to be formed, will move along the circumference so that the minimum enclosing circle does not change. Notice that while moving along the circle, an error prone robot might skid off the circle. Also, when a robot from the inside attempts to move exactly on to the circle, it may move out of the circle due to the error in movement. In both cases, the minimum enclosing circle will change and the progress made by the algorithm will be lost. In fact, we face difficulty at a more fundamental level: exactly forming an arbitrary pattern is impossible by robots with inaccurate movements. Therefore, we consider a relaxed version of the problem called Approximate Arbitrary Pattern Formation where the robots are required to form an approximation of the input pattern F. We show that with no agreement in coordinate system, the problem is unsolvable, even by fully synchronous robots, if the initial configuration 1) has rotational symmetry and there is no robot at the center of rotation, or 2) has reflectional symmetry and there is no robot on the reflection axis. From all other initial configurations, we solve the problem in $\mathcal{OBLOT} + \mathcal{SSYNC}$ (the robots are oblivious, silent and semi-synchronous) and $\mathcal{FCOM} + \mathcal{ASYNC}$ (the robots are oblivious, asynchronous and can communicate using externally visible lights).

Movement error was previously considered in [7], but in the context of the Convergence problem which requires the robots to converge towards a single point. The error model in [7] also considers errors in both direction and extent of the intended movement. However, there is some difference between the error model of [7] and the one introduced in this paper. In particular, the maximum possible error in direction is independent of the extent of the intended movement in [7]. In our model, the maximum possible error in both direction and extent, depend upon the extent of the intended movement. We believe that this is a reasonable assumption as the error is expected to be less if the destination of the intended movement is not far away.

2 Robot Model

A set of n mobile computational entities, called *robots*, are initially positioned at distinct points in the plane. The robots are anonymous, identical, autonomous and homogeneous. The robots are modeled as dimensionless points in the plane. They do not have access to any global coordinate system. Each robot has its own local coordinate system centered at its current position. There is no consistency among the local coordinate systems of the robots except for a common unit of distance. We call this the *standard unit of distance*. Based on the memory and communication capabilities, we consider two standard models: $\mathcal{OBLOT}$ and $\mathcal{FCOM}$. In $\mathcal{OBLOT}$, the robots are *silent* (they have no means of communication) and

oblivious (they have no memory of past observations and computations). In $\mathcal{FCOM}$, each robot is equipped with a light which can assume a constant number of colors and is only visible to other robots. The lights serve as a weak communication mechanism. The robots, when active, operate according to the so-called Look-Compute-Move cycles. 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 (and their lights in case of $\mathcal{FCOM}$) of the robots. Based on the perceived configuration, the robot performs computations according to a deterministic algorithm to decide a destination point (and a color in case of $\mathcal{FCOM}$). Based on the outcome of the algorithm, the robot (sets its light to the computed color in case of $\mathcal{FCOM}$, and) either remains stationary or attempts to move to the computed destination. Based on the activation and timing of the robots, there are three types of schedulers. In $\mathcal{FSYNC}$ or fully synchronous, time can be logically divided into global rounds. In each round, all the robots are activated and they perform their actions at the same time. $\mathcal{SSYNC}$ or semi-synchronous coincides with $\mathcal{FSYNC}$, with the only difference that not all robots are necessarily activated in each round. The most general model is $\mathcal{ASYNC}$ or asynchronous where there are no synchronicity assumptions.

We now describe our movement model. There are known constants $0 < \lambda < 1$, $0 < \Delta < 1$, such that if a robot at x attempts to move to y, then it will reach a point z where $d(z, y) < \mu(x, y)d(x, y)$ where $\mu(x, y) = \min\{\Delta, \lambda d(x, y)\}$. Here $d(x, y)$ denotes (the numerical value of) the distance between the points x and y measured in the standard unit of distance. The movement of the robot will be along the straight line joining x and z. We denote by $\mathcal{Z}(x, y)$ the set of all points where a robot may reach if it attempts to move from x to y. So $\mathcal{Z}(x, y)$ is the open disk $\{z \in \mathbb{R}^2 \mid d(z, y) < \mu(x, y)d(x, y)\}$ (see Fig. 1a). We denote by $error_d(x, y)$ and $error_a(x, y)$ the supremums of the possible distance errors (i.e., the deviation from the intended amount of distance to be traveled) and angle errors (i.e., the angular deviation from the intended trajectory) respectively when a robot intends to travel from x to y. Notice that $error_d(x, y)$ is equal to the radius of $\mathcal{Z}(x, y)$ and $error_a(x, y)$ is equal to the angle between $line(x, y)$ and a tangent on $\mathcal{Z}(x, y)$ passing through x. Hence, $error_d(x, y) = \mu(x, y)d(x, y)$ and $error_a(x, y) = sin^{-1}(\frac{\mu(x,y)d(x,y)}{d(x,y)}) = sin^{-1}(\mu(x, y))$. Also notice that 1) $error_d(x, y)$ increases with $d(x, y)$, and 2) $error_a(x, y)$ increases with $d(x, y)$ only up to a certain value, i.e., $sin^{-1}(\Delta)$ and then remains constant (see Fig. 1b). So, $error_a(x, y) \leq sin^{-1}(\Delta)$, for any x, y.

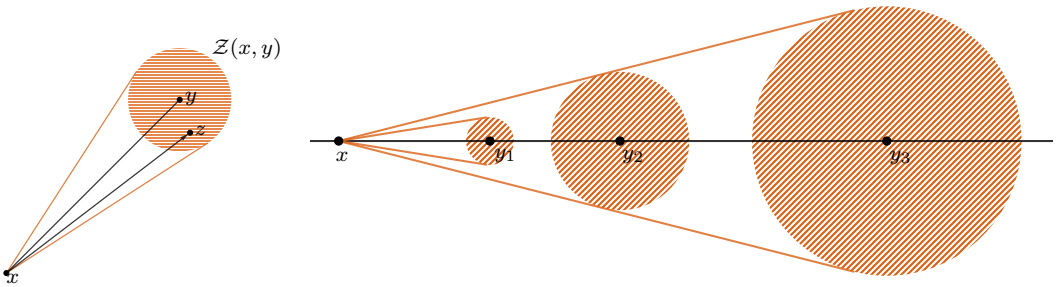

Figure 1 a) If a robot attempts to move from x to y, then it will reach at some point z in the shaded region $\mathcal{Z}(x, y)$. b) If a robot attempts to move from x to y_i, then it will reach at some point in $\mathcal{Z}(x, y_i)$ which is the shaded region around y_i. Observe that $error_d(x, y_1) < error_d(x, y_2) < error_d(x, y_3)$, but $error_a(x, y_1) < error_a(x, y_2) = error_a(x, y_3)$.

3 Definitions and Notations

We denote the configuration of robots by $R = \{r_1, r_2, \ldots, r_n\}$ where each r_i denotes a robot as well as the point in the plane where it is situated. The input pattern given to the robots will be denoted by $F = \{f_1, f_2, \ldots f_n\}$ where each f_i is an element from $\mathbb{R}^2$. Given two points x and y in the Euclidean plane, let $d(x, y)$ denote the distance between the points x and y measured in the standard unit of distance. We denote by $line(x, y)$ the straight line passing through x and y. By $seg(x, y)$ ($\overline{seg}(x, y)$) we denote the line segment joining x and y excluding (resp. including) the end points. If ℓ_1 and ℓ_2 are two parallel lines, then $\mathcal{S}(\ell_1, \ell_2)$ denotes the open region between these two lines. For any point c in the Euclidean plane and a length l, $C(c, l) = \{z \in \mathbb{R}^2 \mid d(c, z) = l\}$, $B(c, l) = \{z \in \mathbb{R}^2 \mid d(c, z) < l\}$ and $\overline{B}(c, l) = \{z \in \mathbb{R}^2 \mid d(c, z) \leq l\} = B(c, l) \cup C(c, l)$. If C is a circle then $encl(C)$ and $\overline{encl}(C)$ respectively denote the open and closed region enclosed by C. Also, $ext(C) = \mathbb{R}^2 \setminus \overline{encl}(C)$ and $\overline{ext}(C) = \mathbb{R}^2 \setminus encl(C)$. Hence, $\overline{encl}(C) = encl(C) \cup C$ and $\overline{ext}(C) = ext(C) \cup C$. Let x, y be two points in the plane and $d(x, y) > l$. Suppose that the tangents from x to $C(y, l)$ touches $C(y, l)$ at a and b. The $Cone(x, B(y, l))$ is the open region enclosed by $B(y, l), \overline{seg}(x, a)$ and $\overline{seg}(x, b)$, as shown in Fig. 2. Also, $\angle Cone(x, B(y, l)) = \angle axb$. We denote by $\mathcal{F}(x, y)$ the family of circles passing through x and y. The center of all the circles lie on the perpendicular bisector of $\overline{seg}(x, y)$. If $C_1, C_2 \in \mathcal{F}(x, y)$ and c_1, c_2 be their centers respectively, then $(C_1, C_2)_{\mathcal{F}(x,y)}$ and $[C_1, C_2]_{\mathcal{F}(x,y)}$ will denote respectively the family of circles $\{C \in \mathcal{F}(x, y) \mid c \in seg(c_1, c_2),$ where c is the center of $C\}$ and $\{C \in \mathcal{F}(x, y) \mid c \in \overline{seg}(c_1, c_2),$ where c is the center of $C\}$.

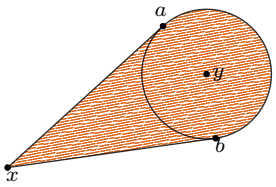

Figure 2 $Cone(x, B(y, l))$ is defined as the shaded open region enclosed by $B(y, l), \overline{seg}(x, a)$ and $\overline{seg}(x, b)$.

For a set P of points in the plane, $C(P)$ and $c(P)$ will respectively denote the minimum enclosing circle of P (i.e., the smallest circle C such that $P \subset \overline{encl}(C)$) and its center. The smallest enclosing circle $C(P)$ is unique and can be computed in linear time. For P, with $2 \leq |P| \leq 3$, $CC(P)$ denotes the circumcircle of P defined as the following. If $P = \{p_1, p_2\}$, $CC(P)$ is the circle having $\overline{seg}(p_1, p_2)$ as the diameter and if $P = \{p_1, p_2, p_3\}$ and the three points are not collinear, $CC(P)$ is the unique circle passing through p_1, p_2 and p_3.

▶ **Property 1.** *If $P' \subseteq P$ such that 1) P' consists of two points or P' consists of three points that form an acute angled triangle, and 2) $P \subset \overline{encl}(CC(P'))$, then $C(P) = CC(P')$. Conversely, for any P, $\exists P' \subseteq P$ so that 1) P' consists of two points or P' consists of three points that form an acute angled triangle and 2) $C(P) = CC(P')$.*

From Property 1 it follows that $C(P)$ passes either through two points of P that are on the same diameter (antipodal points), or through at least three points so that some three of them form an acute angled or right angled triangle. A point $p \in P$ is said to be *critical* if $C(P) \neq C(P \setminus \{p\})$. Note that $p \in P$ is critical only if $p \in C(P)$.

▶ **Property 2.** *If $|P \cap C(P)| \geq 4$ then there exists at least one point from $P \cap C(P)$ which is not critical.*

Consider all concentric circles that are centered at $c(P)$ and passes through at least one point of P. Let $C^i_{\downarrow}(P)$ ($C^i_{\uparrow}(P)$) denote the ith ($i \geq 1$) of these circles so that $C^{i+1}_{\downarrow}(P) \subset encl(C^i_{\downarrow}(P))$ (resp. $C^i_{\uparrow}(P) \subset encl(C^{i+1}_{\uparrow}(P))$). We shall denote $c(P)$ by $C^0_{\uparrow}(P)$. So we have $C^1_{\downarrow}(P) = C(P)$ and if there is a point at $c(P)$, then $C^1_{\uparrow}(P) = c(P) = C^0_{\uparrow}(P)$. We say that a configuration of robots R is *symmetry safe* if one of the following three conditions hold (see Fig. 3).

1. i) there is some non-critical robot on $C(R)$, hence $|R \cap C(R)| \geq 3$, ii) there is no robot at $c(R)$, iii) $|R \cap C^1_{\uparrow}(R)| = 1$ and $|R \cap C^2_{\uparrow}(R)| = 1$, iv) if $R \cap C^1_{\uparrow}(R) = \{r_1\}$ and $R \cap C^2_{\uparrow}(R) = \{r_2\}$, then $r_1, r_2, c(R)$ are not collinear.
2. i) all robots on $C(R)$ are critical and $R \cap C(R) = \{r_1, r_2, r_3\}$, ii) $\Delta r_1 r_2 r_3$ is scalene, i.e., all three sides have different lengths.
3. i) all robots on $C(R)$ are critical and $R \cap C(R) = \{r_1, r_2\}$, ii) $|R \cap C^1_{\uparrow}(R)| = 1$, iii) if $R \cap C^1_{\uparrow}(R) = \{r\}$, $r \notin line(r_1, r_2) \cup \ell$, where ℓ is the line passing through $c(R)$ and perpendicular to $line(r_1, r_2)$.

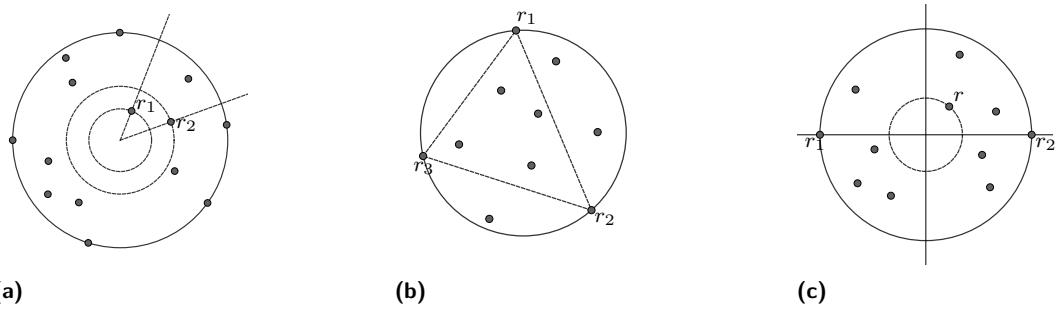

Figure 3 Illustrations of symmetry safe configurations.

We shall say that a configuration R of robots has an *unbreakable symmetry* if one of the following is true: i) R has rotational symmetry with no robot at $c(R)$, ii) R has reflectional symmetry with respect to a line ℓ with no robot on ℓ.

4 Approximate Arbitrary Pattern Formation

The Arbitrary Pattern Formation problem in its standard form is the following. Each robot of a team of n robots is given a pattern F as input which is a list of n distinct elements from $\mathbb{R}^2$. The given input F is exactly same for each robot. The problem asks for a distributed algorithm that guides the robots to a configuration that is similar to F with respect to translation, reflection, rotation and uniform scaling. We refer to this version of the problem as the Exact Arbitrary Pattern Formation problem, highlighting the fact that the configuration of the robots is required to be exactly similar to the input pattern. However, it is not difficult to see that Exact Arbitrary Pattern Formation is unsolvable in our model where the robot movements are inaccurate.

▶ **Theorem 1.** *Exact Arbitrary Pattern Formation is unsolvable by robots with inaccurate movements.*

Therefore, we introduce a relaxed version of the problem called the Approximate Arbitrary Pattern Formation. Intuitively, we want the robots to form a pattern that is close to the given pattern, but may not be exactly similar to it. Formally, the robots are given as input a pattern F and a number $0 < \epsilon < 1$. The number ϵ is small enough so

that the distance between no two pattern points is less than $2\epsilon D$ where D is the diameter of $C(F)$. Given the input (F, ϵ), the problem requires the robots to a form a configuration $R = \{r_1, \ldots, r_n\}$ such that there exists an embedding (subject to translation, reflection, rotation and uniform scaling) of the pattern F on the plane, say $P = \{p_1, \ldots, p_n\}$, such that $d(p_i, r_i) \leq \epsilon\overline{D}$ for all $i = 1, \ldots, n$, where $\overline{D}$ is the diameter of $C(P)$. In this case, we say that *the configuration R is ϵ-close to the pattern F*. Recall that the number ϵ is such that the disks $B(f_i, \epsilon D)$ are disjoint. Since P is similar to F, disks $B(p_i, \epsilon\overline{D})$ are also disjoint. The problem requires that exactly one robot is placed inside each disk. Furthermore, the movements should be collisionless.

It is well known that Arbitrary Pattern Formation is unsolvable if the initial configuration has an unbreakable symmetry. It can be shown that this also holds for Approximate Arbitrary Pattern Formation (See Appendix A for proof).

▶ **Theorem 2.** *Approximate Arbitrary Pattern Formation is deterministically unsolvable, even with Rigid movements, if the initial configuration has unbreakable symmetries.*

5 The Algorithm for Semi-Synchronous Robots

In this section, we present an algorithm that solves Approximate Arbitrary Pattern Formation in $\mathcal{OBLOT} + \mathcal{SSYNC}$ from any initial configuration that does not have any unbreakable symmetries. The algorithm works in three phases which we shall describe in the next three subsections. For each phase, we shall first present the idea behind the approach and then give a brief description of the algorithm. More detailed description along with formal proofs can be found in the full version [3] of the paper.

5.1 Phase 1

Motive and Overview

The goal of Phase 1 is to create a configuration which is asymmetric and in which all robots on its minimum enclosing circle are critical. Phase 1 consists of three subphases, namely Subphase 1.1, Subphase 1.2 and Subphase 1.3. If the configuration is symmetric, our first step would be to get rid of the symmetry. Since the initial configuration cannot have any unbreakable symmetries, it will be possible to choose some unique robot from the configuration. We can remove the symmetry by appropriately moving this robot. This is done in Subphase 1.1. Once we have an asymmetric configuration, the next objective is to bring inside some non-critical robots from the minimum enclosing circle so that all the remaining robots on the minimum enclosing circle are critical. However, we have to make sure that these moves do not create new symmetries in the configuration. For this, we first make the configuration symmetry safe, i.e., have unique robots r_1 and r_2 respectively closest and second closest from the center of the minimum enclosing circle such that r_1 and r_2 are not on the same diameter. This is done in Subphase 1.2. After this, in Subphase 1.3, we start bringing inside the robots from the circumference. The movements of the robots should be such that r_1 and r_2 remain the unique closest and second closest robot from the center. This ensures that these movements do not create any symmetries. The two properties that we achieved in Phase 1, namely, having an asymmetric configuration and not having any non-critical robot on the minimum enclosing circle, will play crucial role in our approach and hence, will be preserved during the rest of the algorithm. This will be the case even if the target pattern F is symmetric or has non-critical robots on its minimum enclosing circle. This is not a problem as we are not required to exactly form the pattern F. Any pattern F can be approximated by a pattern that is asymmetric and has no non-critical points on its minimum enclosing circle.

Brief Description of the Algorithm

We shall now briefly describe the algorithm. The algorithm is in Phase 1 if $\neg\mathtt{u} \wedge (\neg\mathtt{a} \vee \neg\mathtt{c})$ holds, where $\mathtt{a}$ = "R is an asymmetric configuration", $\mathtt{u}$ = "R has an unbreakable symmetry", $\mathtt{c}$ = "all robots on $C(R)$ are critical". The objective is to create a configuration with $\mathtt{a} \wedge \mathtt{c}$. The algorithm is in Subphase 1.1 if $\neg\mathtt{u} \wedge \neg\mathtt{a}$ holds, in Subphase 1.2 if $\mathtt{a} \wedge \neg\mathtt{s} \wedge \neg\mathtt{c}$ holds ($\mathtt{s}$ = "R is symmetry safe") and in Subphase 1.3 if $\mathtt{s} \wedge \neg\mathtt{c}$ holds.

First we describe Subphase 1.1. We have $\neg\mathtt{u}\wedge\neg\mathtt{a}$. Our objective is to create an asymmetric configuration, i.e., have $\mathtt{a}$. As mentioned earlier, we will remove the symmetry by moving exactly one robot of the configuration, while all other robots will remain stationary. The fact that we have $\neg\mathtt{u}$, will allow us to select one such robot from the configuration. To describe the algorithm, we have to consider the following four cases. Case 1 consists of the configurations in Subphase 1.1 where there is a robot at $c(R)$. Now consider the cases where there is no robot at $c(R)$. Notice that in this case, R cannot have a rotational symmetry because $\neg\mathtt{u}$ holds. So R has a reflectional symmetry with respect to a unique line ℓ as reflectional symmetry with respect to two different lines imply rotational symmetry. Since $\neg\mathtt{u}$ holds, there are robots on ℓ. If there is a non-critical robot on ℓ then we call it Case 2. For the remaining cases where there is no non-critical robot on ℓ, we call it Case 3 if there are more than 2 robots on $C(R)$ and Case 4 if there are exactly 2 robots on $C(R)$.

In Case 1, we have a robot r at $x = c(R)$. In this case, r will move away from the center and all other robots will remain static. The destination y chosen by the robot r should satisfy the following conditions: (1) $\mathcal{Z}(x, y) \subset encl(C^2_{\uparrow}(R)) \setminus \{c(R)\}$, (2) $\mathcal{Z}(x, y) \cap \ell = \emptyset$ for any reflection axis ℓ of $R \setminus \{r\}$. It is easy to see that such an y exists. Furthermore, r can easily compute such an y.

In Case 2, there is no robot at $c(R)$, R has reflectional symmetry with respect to a unique line ℓ and there is at least one non-critical robot on ℓ. If there are more than one non-critical robots on ℓ, we can single out one of them using the concept of *view* of a robot (see Appendix A for details). In particular, all robots on ℓ will have distinct views (because otherwise R will have rotational symmetry) and hence we have a unique non-critical robot r with minimum view. Only r will move in this case. Suppose that r is at point x. The destination y chosen by r should satisfy the following conditions: (1) if $x \in C^i_{\uparrow}(R)$, then $Cone(x, \mathcal{Z}(x, y)) \subset encl(C^i_{\uparrow}(R)) \setminus \overline{encl}(C^{i-1}_{\uparrow}(R))$, (2) $\mathcal{Z}(x, y) \cap \ell' = \emptyset$ for any reflection axis ℓ' of $R \setminus \{r\}$. Such points clearly exist and r can easily compute one.

In Case 3, we have no robot at $c(R)$, R has reflectional symmetry with respect to a unique line ℓ, there is no non-critical robot on ℓ and $C(R)$ has at least 3 robots on it. In this case, it can be shown that there is exactly one robot on ℓ and it is on $C(R)$. Call this robot r. Let x denote its position. Let r_1, r_2 be the two robots (specular with respect to ℓ) on $C(R)$ such that $\angle rc(R)r_1 = \angle rc(R)r_2 = max\{\angle rc(R)r'' \mid r'' \in R \cap C(R)\}$. It can be shown that $\frac{\pi}{2} < \angle rc(R)r_1 = \angle rc(R)r_2 < \pi$. Only r will move in this case and the rest will remain static. Here the robot will move outside of the current minimum enclosing circle. The chosen destination y should satisfy the following conditions: (1) $\mathcal{Z}(x, y) \cap \ell = \emptyset$, (2) $Cone(x, \mathcal{Z}(x, y)) \subset ext(C(R)) \cap encl(C') \cap \mathcal{H}$ where C' is the largest circle from $\{C \in \mathcal{F}(r_1, r_2) \mid R \subset \overline{encl}(C)\}$ and $\mathcal{H}$ is the open half-plane delimited by $line(r_1, r_2)$ that contains x, (3) $\mathcal{Z}(x, y) \cap C_i = \emptyset$, where $C_i = C(r_i, d(r_1, r_2)), i = 1, 2$, (4) $\mathcal{Z}(x, y) \subset \mathcal{S}(L_1, L_2)$, where L_i is the line parallel to ℓ and passing through r_i, $i = 1, 2$. Again, it is staightforward to see that such an y should exist and r can easily compute one.

In Case 4, we have no robot at $c(R)$, R has reflectional symmetry with respect to a unique line ℓ, there is no non-critical robot on ℓ and $C(R)$ has exactly 2 robots on it. In this case, it can be shown that there is no robot on $\ell \cap encl(C(R))$ and there are two

antipodal robots on ℓ, say r and r'. Also, the views of r and r' must be different. So let r be the robot with minimum view. Only r will move in this case. Let ℓ' be the line perpendicular to ℓ and passing through r. For each $r'' \in R \setminus \{r, r'\}$, consider the line passing through r'' and perpendicular to $seg(r'', r')$. Consider the points of intersection of these lines with ℓ'. Let P_1, P_2 (specular with respect to ℓ) be the two of these points that are closest to ℓ. Let L_1, L_2 be the lines parallel to ℓ and passing through P_1, P_2 respectively. Assuming that r is at point x, the destination y chosen by r should satisfy the following conditions: (1) $Cone(x, \mathcal{Z}(x, y)) \subset ext(C(R))$, (2) $\mathcal{Z}(x, y) \cap \ell = \emptyset$, (3) $\mathcal{Z}(x, y) \subset \mathcal{S}(L_1, L_2)$, (4) $\mathcal{Z}(x, y) \cap C(c, d(c, r')) = \emptyset$, where $c = c(R \setminus \{r, r'\})$.

It can be shown that the movements described for Subphase 1.1 will lead to an asymmetric configuration. The algorithm will be in Subphase 1.2 if $\mathtt{a} \wedge \neg \mathtt{s} \wedge \neg \mathtt{c}$ holds. Then our goal would be to make the configuration symmetry safe. This can be easily done. When $\mathtt{s} \wedge \neg \mathtt{c}$ holds, we are in Subphase 1.3 . Then our objective would be to have $\mathtt{a} \wedge \mathtt{c}$. As the configuration is asymmetric (as $\mathtt{s} \implies \mathtt{a}$), there is a robot with minimum view among all the non-critical robots lying on $C(R)$. This robot will move inside. Continuing in this manner, non-critical robots on $C(R)$ will sequentially move inside until we obtain $\mathtt{a} \wedge \mathtt{c}$.

5.2 Phase 2

Motive and Overview

Phase 1 was a preprocessing step where a configuration was prepared in which there is no symmetry and all robots on the minimum enclosing circle are critical. Actual formation of the pattern will be done in two steps, in Phase 2 and Phase 3. In Phase 2, the robots on the minimum enclosing circle will reposition themselves according to the target pattern and then in Phase 3, the robots inside the minimum enclosing circle will move to complete the pattern. The standard approach to solve the Arbitrary Pattern Formation problem, however, is exactly the opposite. Usually, the part of the pattern inside the minimum enclosing circle is first formed and then the pattern points on the minimum enclosing circle are occupied by robots. In this approach, the minimum enclosing circle is kept invariant throughout the algorithm. Keeping the minimum enclosing circle fixed is important because it helps to fix the coordinate system with respect to which the pattern is formed. During the second step, a robot on the minimum enclosing circle may have to move to another point on the circle. In order to keep the minimum enclosing circle unchanged, it has to move exactly along the circumference. However, it is not possible to execute such movement in our model. An error in movement in this step will change the minimum enclosing circle and the progress made by the algorithm will be lost. Placing the robots at the correct positions on the minimum enclosing circle is a difficult issue in our model. In fact, it can be proved that it is impossible to deterministically obtain a configuration with ≥ 4 robots on the minimum enclosing circle if the initial configuration has < 4 robots on the minimum enclosing circle. For this reason, we shall work with 2 or 3 (critical) robots on the minimum enclosing circle as obtained from Phase 1 (or may be from the beginning). So in Phase 2, we start with an asymmetric configuration where all robots on the minimum enclosing circle are critical. The objective of this phase is to move these critical robots so that their relative positions on the minimum enclosing circle is consistent with the target pattern. For this, we shall choose a set of two or three pattern points from the minimum enclosing circle of the target pattern. We shall call this set the bounding structure of the target pattern. Essentially, the objective of Phase 2 is to approximate this structure by the critical robots.

The Bounding Structure

If Algorithm 1 is applied on the target pattern F, then we obtain a set $B_F \subseteq C(F) \cap F$ of pattern points such that B_F is a minimal set of points of $C(F) \cap F$ such that $CC(B_F) = C(F)$. By minimal set we mean that no proper subset of B_F has this property. By Property 1, B_F either consists of two antipodal points or three points that form an acute angled triangle. We call B_F the *bounding structure* of F (see Fig. 4a). Recall that each robot computes the same bounding structure since the input $F = \{f_1, f_2, \dots f_n\}$ is same for all robots. We say that the bounding structure of F is formed by the robots if one of the following holds.

1. B_F has exactly two points, $C(R)$ has exactly two robots on it and R is symmetry safe.
2. B_F has exactly three points and $C(R)$ has exactly three robots on it (see Fig. 4). Let $B_F = \{f_{i_1}, f_{i_2}, f_{i_3}\}$ and $C(R) \cap R = \{r_1, r_2, r_3\}$. R is symmetry safe (i.e. $\Delta r_1 r_2 r_3$ is scalene) and furthermore, if $seg(r_1, r_2)$ is the largest side of the triangle formed by r_1, r_2, r_3 and $seg(f_{i_1}, f_{i_2})$ is a largest side of the triangle formed by $f_{i_1}, f_{i_2}, f_{i_3}$, then $\exists$ an embedding $f_i \mapsto P_i$ of F on the plane identifying $seg(f_{i_1}, f_{i_2})$ with $seg(r_1, r_2)$ so that i) $r_3 \in B(P_{i_3}, \epsilon D)$, ($D$ = diameter of $C(P_1, \dots, P_n)$) and ii) $B(P_i, \epsilon D) \cap encl(CC(r_1, r_2, r_3)) \neq \emptyset$ for all $i \in \{1, \dots, n\}$

Algorithm 1 Algorithm producing the bounding structure of a pattern.

Input : A pattern $F = \{f_1, \dots, f_n\}$
Let $C(F) \cap F = \{f_{j_1}, \dots, f_{j_k}\}$, where $j_1 < \dots < j_k$
$B_F \leftarrow \{f_{j_1}, \dots, f_{j_k}\}$
for $l \in 1, \dots, k$ **do**
  if f_{j_l} *is non-critical in* F **then**
    $F \leftarrow F \setminus \{f_{j_l}\}$
    $B_F \leftarrow B_F \setminus \{f_{j_l}\}$
Return B_F

Brief Description of the Algorithm

The algorithm is in Phase 2 if $\mathtt{a} \wedge \mathtt{c} \wedge \neg \mathtt{b}$ holds ($\mathtt{b}$ = "the bounding structure is formed"). The objective is to have $\mathtt{b}$. We describe the algorithm for the following cases: $C(R)$ has three robots and the bounding structure also has three points (Case 1), $C(R)$ has three robots and the bounding structure has two points (Case 2), $C(R)$ has two robots and the bounding structure has three points (Case 3) and $C(R)$ has two robots and the bounding structure also has two points (Case 4).

First consider Case 1. Here the goal is to transform the triangle of the robots on $C(R)$ so that the bounding structure of F is formed. Let $C(R) \cap R = \{r_1, r_2, r_3\}$. If $\Delta r_1 r_2 r_3$ is not scalene, then we shall make it so by using similar techniques from Subphase 1.1, Case 3. So now assume that $\Delta r_1 r_2 r_3$ is scalene. Let $\overline{seg}(r_1, r_2)$ be the largest side of $\Delta r_1 r_2 r_3$. In that case, r_3 will be called the *transformer robot*. This robot will move to form the bounding structure of F. Let L be the perpendicular bisector of $\overline{seg}(r_1, r_2)$. Since no two sides of the triangle are of equal length, $r_3 \notin L$. Let $\mathcal{H}$ be the open half-plane delimited by L that contains r_3. Without loss of generality, assume that $r_1 \in \mathcal{H}$. Let L_1 be the line parallel to L and passing through r_1. Let $\mathcal{H}'$ be the open half-plane delimited by L_1 that contains L. Since $\Delta r_1 r_2 r_3$ is acute angled, $r_3 \in \mathcal{H}'$. Let $\mathcal{H}''$ be the open half-plane delimited by $line(r_1, r_2)$ that contains r_3. Let $C_1 = C(r_1, d(r_1, r_2))$ and $C_2 = C(r_2, d(r_2, r_1))$. Since $\overline{seg}(r_1, r_2)$ is (strictly) the largest side of $\Delta r_1 r_2 r_3$, $r_3 \in encl(C_1) \cap encl(C_2)$. If $C_3 = CC(r_1, r_2)$, then $r_3 \in ext(C_3)$ as $\Delta r_1 r_2 r_3$ is acute angled. Now take the largest side of the bounding structure B_F. In case of a tie, use the ordering of the points in the input F to choose one of them. Embed

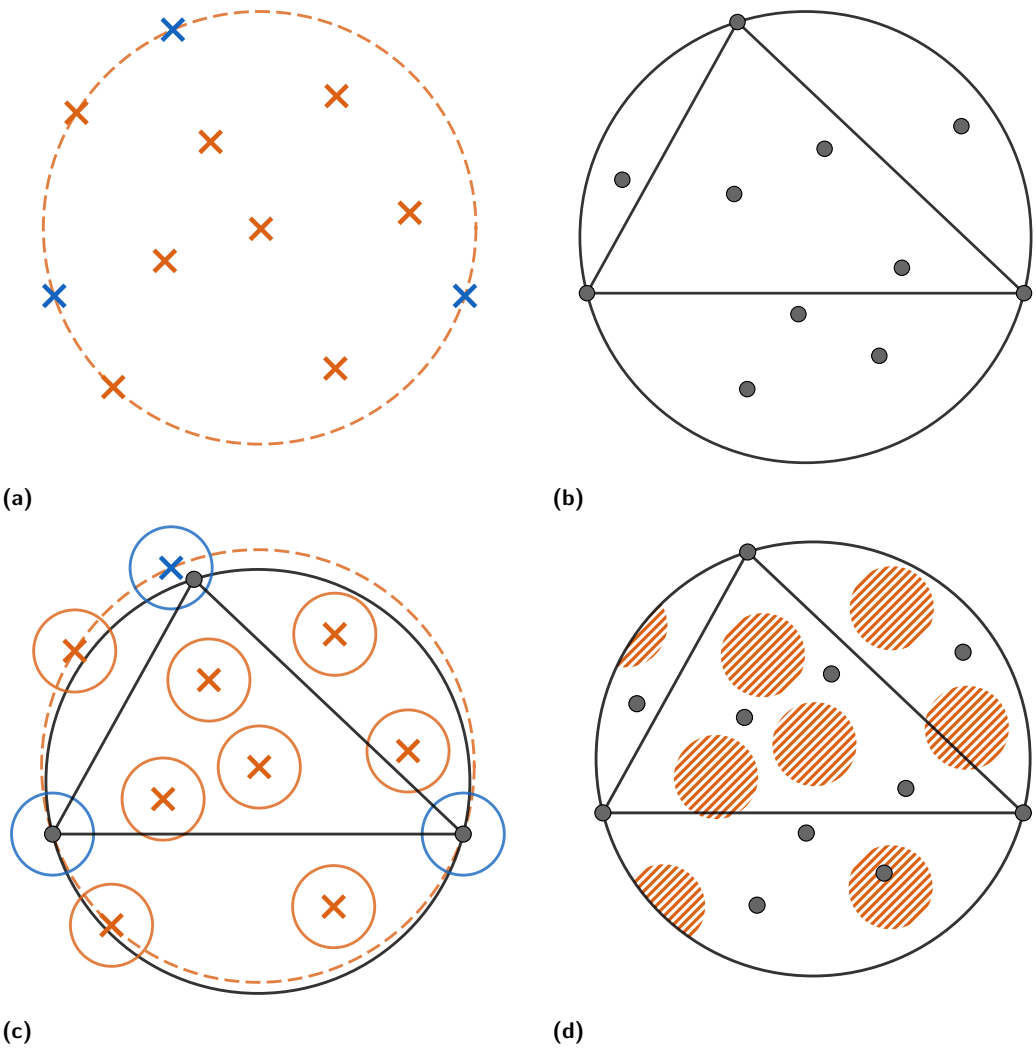

Figure 4 a) The input pattern F. The bounding structure b_F consists of the blue pattern points. b)-c) The bounding structure is formed by the robots. d) To obtain a final configuration, each shaded region must have a robot inside it.

the bounding structure B_F on the plane identifying this side with $\overline{seg}(r_1, r_2)$ so that the third point of the bounding structure is mapped to a point $P \in \overline{\mathcal{H}} \cap \mathcal{H}''$. Since the bounding structure is acute angled, $P \in \mathcal{H}'$. Also, $P \in ext(C_3)$ for the same reason. Furthermore, since a largest side of the bounding structure is identified with $\overline{seg}(r_1, r_2)$, $P \in \overline{encl(C_1)} \cap \overline{encl(C_2)}$. So we have $r_3 \in \mathcal{H} \cap \mathcal{H}' \cap \mathcal{H}'' \cap encl(C_1) \cap encl(C_2) \cap ext(C_3) = \mathcal{U}_{blue}$ (the blue open region in Fig. 5a) and $P \in \overline{\mathcal{H}} \cap \mathcal{H}' \cap \mathcal{H}'' \cap \overline{encl(C_1)} \cap \overline{encl(C_2)} \cap ext(C_3) = \mathcal{U}'_{blue}$. Notice that $\mathcal{U}'_{blue}$ consists of the open region $\mathcal{U}_{blue}$ and some parts of its boundary. Our objective is to move the robot r_3 to a point near P. The entire trajectory of the movement should lie inside the region $\mathcal{U}_{blue}$. However, before this movement, we have to make sure that the configuration satisfies some desirable properties described in the following. Let C_4 be the circle passing through r_1, r_2 and the point in $\mathcal{H}''$ where C_1 and C_2 intersect each other. We shall say that the transformer robot is *eligible to move* if $R \cap encl(C(R)) \subset encl(C_3) \cap encl(C_4) = \mathcal{U}_{red}$ (the red region in Fig. 5b). The transformer robot will not move until it becomes eligible.

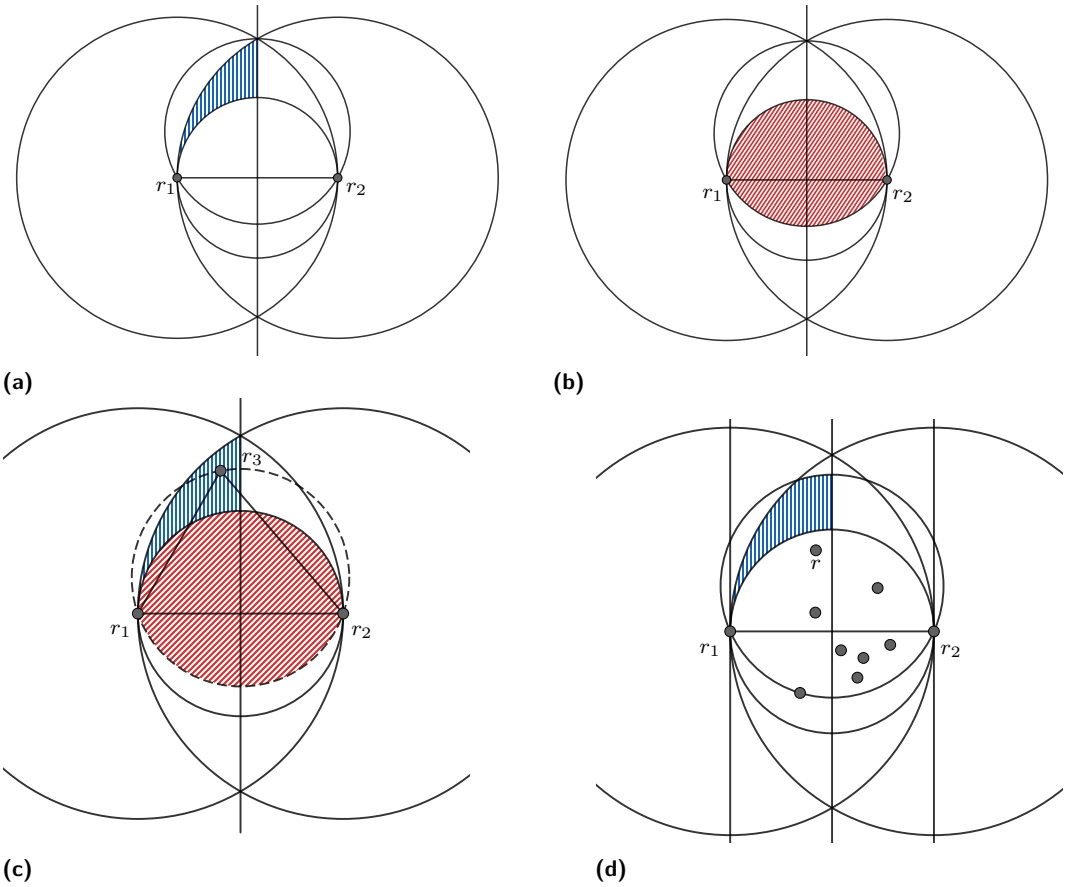

Figure 5 a)-b)Illustrations for Phase 2, Case 1. c) Illustrations for Phase 2, Case 2. d) Illustrations for Phase 2, Case 3.

So the robots in $encl(C(R))$ that are not in $\mathcal{U}_{red}$, should sequentially move inside this region first. Notice that during these movements, the configuration remains asymmetric (as $\Delta r_1 r_2 r_3$ remains scalene) and also r_3 remains the transformer robot. So when we have $R \cap encl(C(R)) \subset \mathcal{U}_{red}$, r_3 will become eligible to move. Now it has to move inside the region $B(P, \epsilon D) \cap \mathcal{U}_{blue}$. However, it is important that its trajectory lies inside $\mathcal{U}_{blue}$. This is because it can be shown that as long as r_3 stays inside the "safe region" $\mathcal{U}_{blue}$, it remains as the transformer robot. This can be done by a movement scheme described in Appendix B that allows a robot to move close to a destination point through a safe region.

In Case 2, $C(R)$ has exactly three robots and the bounding structure has exactly two points. Let $C(R) \cap R = \{r_1, r_2, r_3\}$. As before, $\Delta r_1 r_2 r_3$ will be made scalene. Let $\overline{seg}(r_1, r_2)$ be the largest side. Then r_3 is the transformer robot. The plan is to move r_3 inward so that it is no longer on the minimum enclosing circle. Let $C_1, C_2, C_3, \mathcal{H}, \mathcal{H}', \mathcal{H}''$ denote the same as in Case 1. As before, we have $r_3 \in \mathcal{H} \cap \mathcal{H}' \cap \mathcal{H}'' \cap encl(C_1) \cap encl(C_2) \cap ext(C_3) = \mathcal{U}_{blue}$ (blue region in Fig. 5c). We shall say that the transformer robot is *eligible to move* if 1) $R \cap encl(C(R)) \subset encl(C_3) \cap encl(C(R))$ (red region in Fig. 5c) and 2) $R \setminus \{r_3\}$ is a symmetry safe configuration. The robots in $encl(C(R))$ that are not already in $encl(C_3)$ will move inside it. Then we have $C(R \setminus \{r_3\}) = C_3$ and it passes through only r_1 and r_2. So $R \setminus \{r_3\}$ will be symmetry safe if there is a unique robot closest to O, the midpoint of $seg(r_1, r_2)$, and it is not on $seg(r_1, r_2)$ or its perpendicular bisector. This can be achieved easily. When r_3 becomes eligible to move, it will move inside $encl(C_3)$. During its movement, when it has not

entered $encl(C_3)$, its trajectory should remain inside $\mathcal{U}_{blue}$. Also, when it enters $encl(C_3)$, it should remain in $ext(C)$ where $C = C^1_\uparrow(R \setminus \{r_3\})$. So its entire trajectory should be inside the region $\mathcal{H} \cap \mathcal{H}' \cap \mathcal{H}'' \cap encl(C_1) \cap encl(C_2) \cap ext(C)$ and it should not collide with any robot upon entering $encl(C_3)$. This can be done by the scheme from Appendix B.

In Case 3, $C(R)$ has exactly two robots and the bounding structure consists of exactly three points. Let $C(R) \cap R = \{r_1, r_2\}$. Here the strategy is to move outward one of the robots from $encl(C(R))$, say r, so that the minimum enclosing circle becomes the circumcircle of r, r_1 and r_2. We shall call r the transformer robot. The robot farthest from $c(R)$ will be chosen as the transformer robot. In case of a tie, it is broken using the asymmetry of the configuration. Let $\mathcal{H}$ be the open half plane delimited by $line(r_1, r_2)$ that contains r. Let L_1 and L_2 be the lines perpendicular to $line(r_1, r_2)$ and passing through respectively r_1 and r_2. Let L be the perpendicular bisector of $seg(r_1, r_2)$. Without loss of generality, assume that $r \in \mathcal{S}(L_1, L) \cup L$. Let $C_1 = C(r_1, d(r_1, r_2))$, $C_2 = C(r_2, d(r_2, r_1))$ and $C_3 = CC(r_1, r_2)$. Let C_4 be the largest circle from the family $\{C \in \mathcal{F}(r_1, r_2) \mid \text{center of } C \text{ lies in } \mathcal{H} \text{ and } R \subset \overline{encl}(C)\}$. The algorithm asks r to move into the region $encl(C_1) \cap encl(C_2) \cap ext(C_3) \cap encl(C_4) \cap \mathcal{H} \cap \mathcal{S}(L_1, L)$ (the blue region in Fig. 5d). Again, this can be done by the scheme described in Appendix B.

In Case 4, $C(R)$ has two exactly robots and the bounding structure also has exactly two points. The only time $\neg\mathtt{b}$ may hold is when the configuration is not symmetry safe. So we have to make the configuration symmetry safe by previously discussed techniques.

5.3 Phase 3

Motive and Overview

The algorithm is in Phase 3 if $\mathtt{b}$ holds. The objective of this phase is to form the pattern approximately. Notice that when $\mathtt{b}$ holds, the configuration is symmetry safe and hence asymmetric. This will allow the robots to agree on a coordinate system in which the target will be formed (approximately). During this process, $\mathtt{b}$ has to be preserved because otherwise the agreement in coordinate system will be lost.

The termination condition of the algorithm is that both $\mathtt{b}$ holds (i.e., it is a Phase 3 configuration) and the configuration is ϵ-close to F. Therefore, even if the initial configuration is ϵ-close to F (i.e., the pattern F is already formed approximately), the algorithm will still go through the earlier phases to have $\mathtt{b}$ and then approximately form the pattern while preserving $\mathtt{b}$. The reason why we take this approach is because in general, even if the configuration is ϵ-close to F, the robots may not be able to efficiently identify this. This is a basic difficultly of the problem. However, when $\mathtt{b}$ holds there is a way to fix a particular embedding of F in the plane and then the only thing to check is whether there are robots close to each point of the embedding. For Phase 3, there are two cases to consider: B_F has exactly two points (Case 1) and B_F has exactly three points (Case 2).

Brief Description of the Algorithm

We shall only discuss Case 1 because its techniques can be used to solve Case 2 as well. Case 2 and all the omitted details of Case 1 can be found in the full version [3] of the paper. So for Case 1, let us first describe how we shall fix a common coordinate system. Let $\{r_1, r_2\} = C(R) \cap R$. Let $\ell = line(r_1, r_2)$ and ℓ' be the line passing through $c(R)$ and perpendicular to ℓ. Let r_l be the unique robot closest to $c(R)$. Also it is in $encl(C(R)) \setminus (\ell \cup \ell')$. Such a robot exists because $\mathtt{b}$ holds. We set a *global coordinate system* whose center is at $c(R)$, X axis along ℓ, Y axis along ℓ'. The positive directions of X and Y axis are such that r_l lies in the positive quadrant. Now we choose an embedding of the pattern F that will be approximated. Perform a coordinate transformation (rotation) on the target

pattern F so that the bounding structure is along the X axis. Let F' denote the input after this transformation. Consider the pattern points on $C^1_\uparrow(F')$ except the points of the bounding structure (notice that $C^1_\uparrow(F')$ may have points from the bounding structure when $C^1_\uparrow(F') = C^1_\downarrow(F')$). Reflect the pattern with respect to X axis or Y axis or both, if required, so that at least one of them is in the closed positive quadrant ($X \geq 0, Y \geq 0$). Let F'' denote the pattern thus obtained. Therefore, if $\{f_i, f_j\}$ be the bounding structure, then we have 1) f_i, f_j on the X axis and 2) at least one point from $C^1_\uparrow(F'') \cap (F'' \setminus \{f_i, f_j\})$ in the closed positive quadrant. Each robot applies coordinate transformations on F and obtains the same pattern F''. Let f_l denote the first pattern point from $C^1_\uparrow(F'') \cap (F'' \setminus \{f_i, f_j\})$ that is in the closed positive quadrant. The pattern F'' is mapped in the plane in the global coordinate system and scaled so that the bounding structure is mapped onto $\overline{seg}(r_1, r_2)$. These points are called the *target points*. T denotes the set of target points. Notice that the robot r_l, being the unique robot on $C^1_\uparrow(R)$ and also being in an open quadrant (defined by $\ell \cup \ell'$), plays crucial role in fixing the common coordinate system. This will be preserved throughout the algorithm. In particular, r_l will remain in such a position even in the final configuration. The target point that r_l will approximate in the final configuration will be the target point corresponding to f_l. Let us call it t_l. Now t_l is on $C^1_\uparrow(T)$ and in the closed positive quadrant. As r_l is in the open quadrant, it does not need to move out of it to approximate t_l. Now as r_l needs to remain the closest robot from the center, we will define a circle C_l, that depends only on the position of t_l, and require that in the final configuration we have r_l inside this circle and all robots are outside the circle. If D is the diameter of $C(T)$, i.e., $D = d(r_1, r_2)$, then define the circle C_l as (see Fig. 6) i) if $t_l \in C^1_\uparrow(T) = c(T)$, then $C_l = C(c(T), \epsilon D)$, ii) if $t_l \in C^1_\uparrow(T) = C(T)$, then $C_l = C(c(T), (1-\epsilon)\frac{D}{2})$, iii) otherwise, $C_l = C^1_\uparrow(T)$.

We shall say that a target point $t \neq t_l$ is *realized* by a robot r, if r is the unique closest robot to t and $r \in B(t, \epsilon D) \cap \overline{ext}(C_l) \cap \overline{encl}(C(R))$. We shall say that t_l is *realized* by a robot r if all target points $t \neq t_l$ are realized, r is the robot closest to t_l and $r \in B(t, \epsilon D) \cap encl(C_l)$. Hence, if t_l is realized then it implies that all target points are realized, i.e., the given pattern is formed. We call this the final configuration (see also Fig. 6). Now the objective is to realize all the target points. This will be done in the following way. First the robot r_l moves inside $encl(C_l)$, if not already there. The movement should be such that $\mathtt{s}$ remains true. Then the robots from $R \setminus \{r_l\}$ will sequentially realize all the target points of $T \setminus \{t_l\}$ preserving $\mathtt{s}$. These movements are complicated and are described in the full version [3]. When the target points of $T \setminus \{t_l\}$ are realized, the robot r_l will then realize t_l. Again, $\mathtt{s}$ should remain true and r_l should remain as the unique robot closest to $c(R)$.

5.4 The Main Result

Recall that a configuration with $\neg\mathtt{u} \wedge (\neg\mathtt{a} \vee \neg\mathtt{c})$ is in Phase 1, a configuration with $\mathtt{a} \wedge \mathtt{c} \wedge \neg\mathtt{b}$ is in Phase 2, and a configuration with $\mathtt{b}$ is in Phase 3. It is easy to see that any configuration with $\neg\mathtt{u}$ belongs to one of the three phases. Phase 1 terminates with $\mathtt{a} \wedge \mathtt{c}$ which is either a Phase 2 or Phase 3 configuration. Phase 2 terminates with $\mathtt{b}$ which is a Phase 3 configuration. A final configuration is formed in Phase 3. Hence the algorithm solves the problem in $\mathcal{OBLOT} + \mathcal{SSYNC}$ from any configuration which is $\neg\mathtt{u}$.

6 The Algorithm for Asynchronous Robots

Let us denote the algorithm presented in Section 5 as **A**. It works in $\mathcal{OBLOT} + \mathcal{SSYNC}$. Notice that a feature of this algorithm is that it is *sequential* in the following sense. At any round during the execution of the algorithm, at most one robot decides to move.

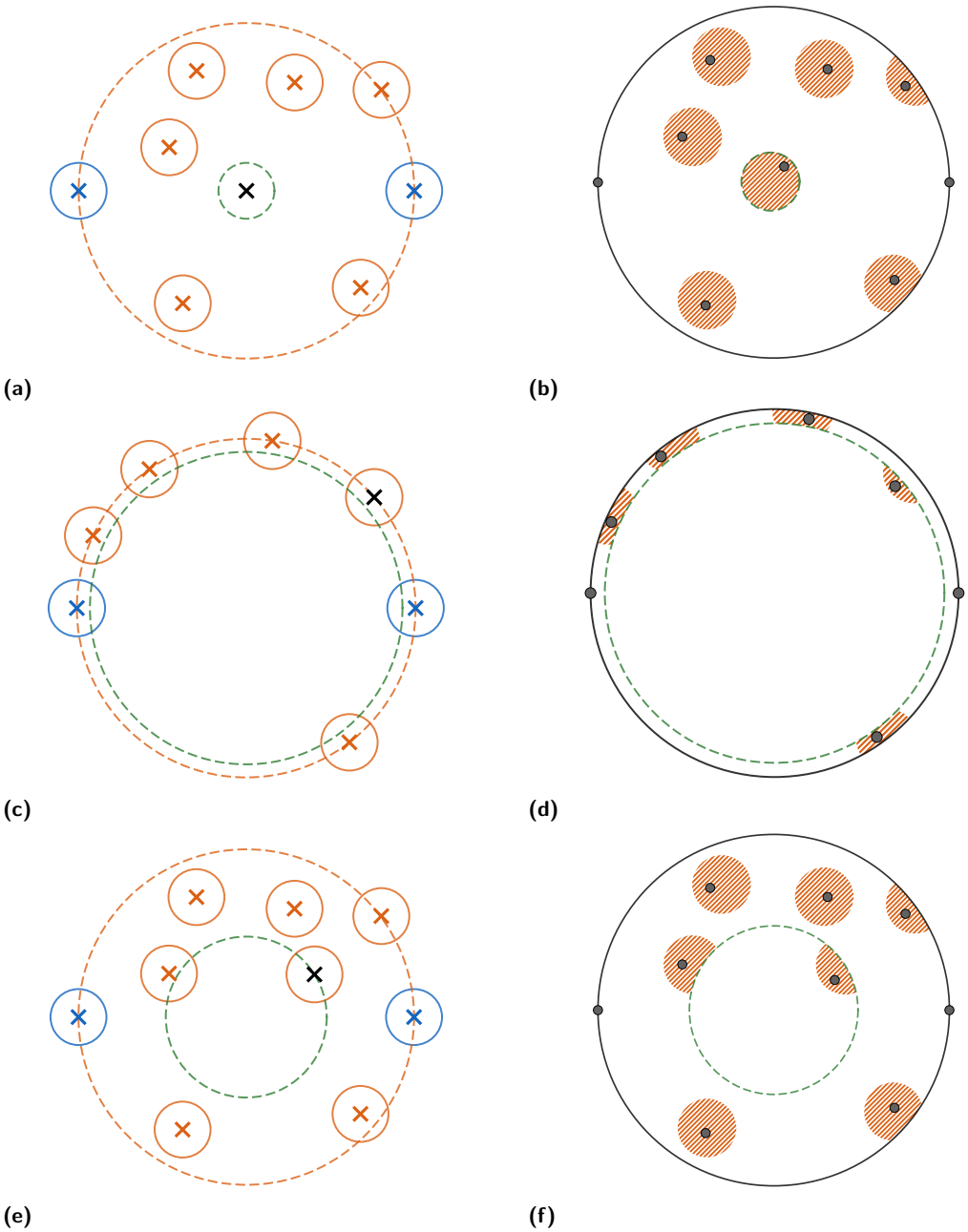

Figure 6 Illustrations for Phase 3, Case 1. In each row, the input pattern F is shown on the left and a final configuration approximating F is shown on the right. In each case, points of the bounding structure are shown in blue, t_l is shown in black and the green circle represents C_l.

This immediately gives an algorithm that works in $\mathcal{FCOM} + \mathcal{ASYNC}$ using two colors $\{\texttt{busy}, \texttt{idle}\}$. The algorithm $\mathbf{A}$ can be seen as a function that maps the snapshot taken by a robot to a movement instruction. We now construct an algorithm $\mathbf{A}'$ from $\mathbf{A}$ with two colors $\{busy, idle\}$ in the following way. Initially the colors of all robots are set to `idle`. If any robot finds some robot with light set to `busy`, then it does nothing. Otherwise, it applies $\mathbf{A}$ on its snapshot (ignoring colors). If $\mathbf{A}$ returns a non-null move, it sets its light to `busy` and moves accordingly. If $\mathbf{A}$ returns a null move, it sets its light to `idle` (recall that it does not know what its present color is) and does not make any move. It is easy to see that $\mathbf{A}'$ solves the problem in $\mathcal{FCOM} + \mathcal{ASYNC}$.

7 Concluding Remarks

We have introduced a model for robots with inaccurate movements. In this model, we have presented algorithms for Approximate Arbitrary Pattern Formation in $\mathcal{OBLOT} + \mathcal{SSYNC}$ and $\mathcal{FCOM} + \mathcal{ASYNC}$. Solving the problem in $\mathcal{OBLOT} + \mathcal{ASYNC}$ is an interesting open problem. The main difficulty of the $\mathcal{ASYNC}$ setting is that a robot can see another robot while the later is moving. How will a robot identify whether a robot in its snapshot is static or moving? In $\mathcal{FCOM}$, a robot used the color `busy` to inform others that it is moving. But this is not possible in $\mathcal{OBLOT}$. Usually such difficulties are handled in a different way in $\mathcal{OBLOT} + \mathcal{ASYNC}$. Suppose that a robot r has to move to a point P. Other robots also know this and conclude that r has completed its movement by simply observing that r has moved to P. But notice that in our case, moving exactly to P is impossible with erroneous movements. Even when r is close to P, it can not be decided whether it is still moving or not. Consider a particular situation in our algorithm where r is moving outside the smallest enclosing circle (as in Phase 1 and Phase 2), i.e., the smallest enclosing circle is changing as r is moving. If we cannot ascertain if r is moving or not, then we cannot ascertain if the smallest enclosing circle is stable or changing. Recall that the center of the smallest enclosing circle is the origin of the coordinate system with respect to which the pattern will be formed. So with a changing smallest enclosing circle, the coordinate system is also changing. So it is crucial to distinguish between moving and static robots. A possible approach in this setting could be that the robots may predict a bound on how much the coordinate system can perturb and act accordingly.

We did not consider multiplicities (points with multiple robots) in the input pattern. Since two robots cannot be brought to the same point in our model, a multiplicity can be interpreted in this case as multiple robots very close to each other. Our algorithm can be adapted to handle inputs with multiplicities. In this work, we modeled the robots as dimensionless points. Another interesting direction for future research would be to consider robots with physical extent.

References

1 Kaustav Bose, Ranendu Adhikary, Manash Kumar Kundu, and Buddhadeb Sau. Positional encoding by robots with non-rigid movements. In *Proc. of 26th International Colloquium on Structural Information and Communication Complexity, SIROCCO 2019, L'Aquila, Italy*, volume 11639 of *LNCS*, pages 94–108. Springer, 2019. doi:10.1007/978-3-030-24922-9_7.

2 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.

3 Kaustav Bose, Archak Das, and Buddhadeb Sau. Pattern formation by robots with inaccurate movements. *arXiv*, abs/2010.09667, 2020. arXiv:2010.09667.

4 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.

5 Quentin Bramas and Sébastien Tixeuil. Brief announcement: Probabilistic asynchronous arbitrary pattern formation. In *Proc. of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, Chicago, IL, USA*, pages 443–445, 2016. doi:10.1145/2933057.2933074.

6 Serafino Cicerone, Gabriele Di Stefano, and Alfredo Navarra. Asynchronous arbitrary pattern formation: the effects of a rigorous approach. *Distributed Computing*, pages 1–42, 2018. doi:10.1007/s00446-018-0325-7.

7 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.

8 Yoann Dieudonné, Franck Petit, and Vincent Villain. Leader election problem versus pattern formation problem. In *Proc. of 24th International Symposium on Distributed Computing, DISC 2010, Cambridge, MA, USA*, pages 267–281, 2010. doi:10.1007/978-3-642-15763-9_26.

9 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.

10 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.

11 Giuseppe Antonio Di Luna, Paola Flocchini, Nicola Santoro, Giovanni Viglietta, and Masafumi Yamashita. Meeting in a polygon by anonymous oblivious robots. *Distributed Comput.*, 33(5):445–469, 2020. doi:10.1007/s00446-019-00362-2.

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

13 Ramachandran Vaidyanathan, Gokarna Sharma, and Jerry L. Trahan. On fast pattern formation by autonomous robots. In *Proc. of 20th International Symposium on Stabilization, Safety, and Security of Distributed Systems, SSS 2018, Tokyo, Japan*, pages 203–220, 2018. doi:10.1007/978-3-030-03232-6_14.

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

15 Yukiko Yamauchi and Masafumi Yamashita. Randomized pattern formation algorithm for asynchronous oblivious mobile robots. In *Proc. of 28th International Symposium on Distributed Computing, DISC 2014, Austin, TX, USA*, pages 137–151, 2014. doi:10.1007/978-3-662-45174-8_10.

A Symmetries and Basic Impossibilities

We first present the concept of *view* (defined similarly as in [6]) of a point in a pattern or a robot in a configuration. The view of a point/robot can be used to determine whether the pattern/configuration is symmetric or asymmetric. Let $R = \{r_1, \ldots, r_n\}$ be a configuration of robots or a pattern of points. A map $\varphi : R \to R$ is called an *isometry* or *distance preserving* if $d(\varphi(r_i), \varphi(r_j)) = d(r_i, r_j)$ for any $r_i, r_j \in R$. R is said to be *asymmetric* if R admits only the identity isometry, and otherwise it is called *symmetric*. The possible symmetries that a symmetric pattern/configuration can admit are reflections and rotations. For any $r \in R$, its *clockwise view*, denoted by $\mathcal{V}^{\circlearrowright}(r)$, is a string of $n+1$ elements from $\mathbb{R}^2$ defined as the following. For $r \neq c(R)$, consider the polar coordinates of the points/robots in the coordinate system with origin at $c(R)$, $\overrightarrow{c(R)r}$ as the reference axis and the angles measured in clockwise

direction. The first element of the string $\mathcal{V}^{\circlearrowright}(r)$ is the coordinates of r and next n elements are the coordinates of the n points/robots ordered lexicographically. For $r = c(R)$, all $n+1$ elements are taken $(0,0)$. The *counterclockwise view* $\mathcal{V}^{\circlearrowleft}(r)$ is defined analogously. Among $\mathcal{V}^{\circlearrowright}(r)$ and $\mathcal{V}^{\circlearrowleft}(r)$, the one that is lexicographically smaller is called the *view* of r and is denoted as $\mathcal{V}(r)$. In a configuration, each robot can compute its view as well as the views of all other robots. Hence, the following properties can be used by the robots to detect whether the configuration is symmetric or not [6]: 1) R admits a reflectional symmetry if and only if there exist two points $r_i, r_j \in R$, $r_i, r_j \neq c(R)$, not necessarily distinct, such that $\mathcal{V}^{\circlearrowright}(r_i) = \mathcal{V}^{\circlearrowleft}(r_j)$, 2) R admits a rotational symmetry if and only if there exist two points $r_i, r_j \in R$, $r_i \neq r_j$, $r_i, r_j \neq c(R)$, such that $\mathcal{V}^{\circlearrowright}(r_i) = \mathcal{V}^{\circlearrowright}(r_j)$.

A problem that is closely related is the Leader Election problem where a unique robot from the team is to be elected as the leader. It is a well-known result [6] that Leader Election is deterministically solvable if and only if the initial configuration R does not have i) rotational symmetry with no robot at $c(R)$ or ii) reflectional symmetry with respect to a line ℓ with no robot on ℓ. We call the symmetries i) and ii) *unbreakable symmetries*. If a configuration does not have such symmetries, then it can be shown that the robots can use the views to elect a unique leader. It is well-known [6] that Exact Arbitrary Pattern Formation is deterministically unsolvable, even with Rigid movements, if the initial configuration has unbreakable symmetries. The same result holds for Approximate Arbitrary Pattern Formation as stated in Theorem 2

Proof of Theorem 2

Proof. For any configuration of robots R, define $\gamma(r)$ for any $r \in R$ as $\gamma(r) = \Sigma_{r' \in R \setminus \{r\}} d(r, r')$. Let R_0 be an initial configuration of n robots that has an unbreakable symmetry. For the sake of contradiction, assume that there is a distributed algorithm $\mathcal{A}$ that solves Approximate Arbitrary Pattern Formation for any input (F, ϵ) from this configuration, i.e., it forms a configuration that is ϵ-close to F. Consider the following input pattern $F = \{f_1, f_2, \ldots, f_n\}$, where f_1, f_2, f_3 form an isosceles triangle with $d(f_1, f_2) = d(f_1, f_3) > d(f_2, f_3)$ and $f_4, \ldots, f_n$ are arranged on the smaller side of the triangle. If $d(f_1, f_2) = d(f_1, f_3)$ is sufficiently large compared to $d(f_2, f_3)$ and ϵ is sufficiently small, then for any configuration R' of robots that is ϵ-close to F, we have $\gamma(r_1) > \gamma(r)$ for all $r \in R' \setminus \{r_1\}$, where r_1 is the robot approximating f_1. This property can be used to elect r_1 as the leader. Hence, Approximate Arbitrary Pattern Formation can be used to solve Leader Election from the initial configuration R_0. This is a contradiction to the fact that Leader Election is deterministically unsolvable from a configuration with unbreakable symmetries [6]. ◀

B Moving Through Safe Zone

In this section, we present some movement strategies that will be used several times in the main algorithm. Suppose that a robot needs to move to or close to some point in the plane. If the point is far away from the robot and it attempts to reach it in one step, the error would be very large and it will miss the target by a large distance. As a result, it may reach a point which causes the configuration to loose some desired property. Also, due to the large deviation from the intended trajectory, it may collide with other robots. So the robot needs to move towards its target in multiple steps and move through a "safe" region where it does not collide with any robot and the desired properties of the configuration are preserved. We first discuss the following problem. Let x_0 and y be two points in the plane so that $d(x_0, y) > l$. Suppose that a robot r is initially at x_0 and the objective is that it has to move

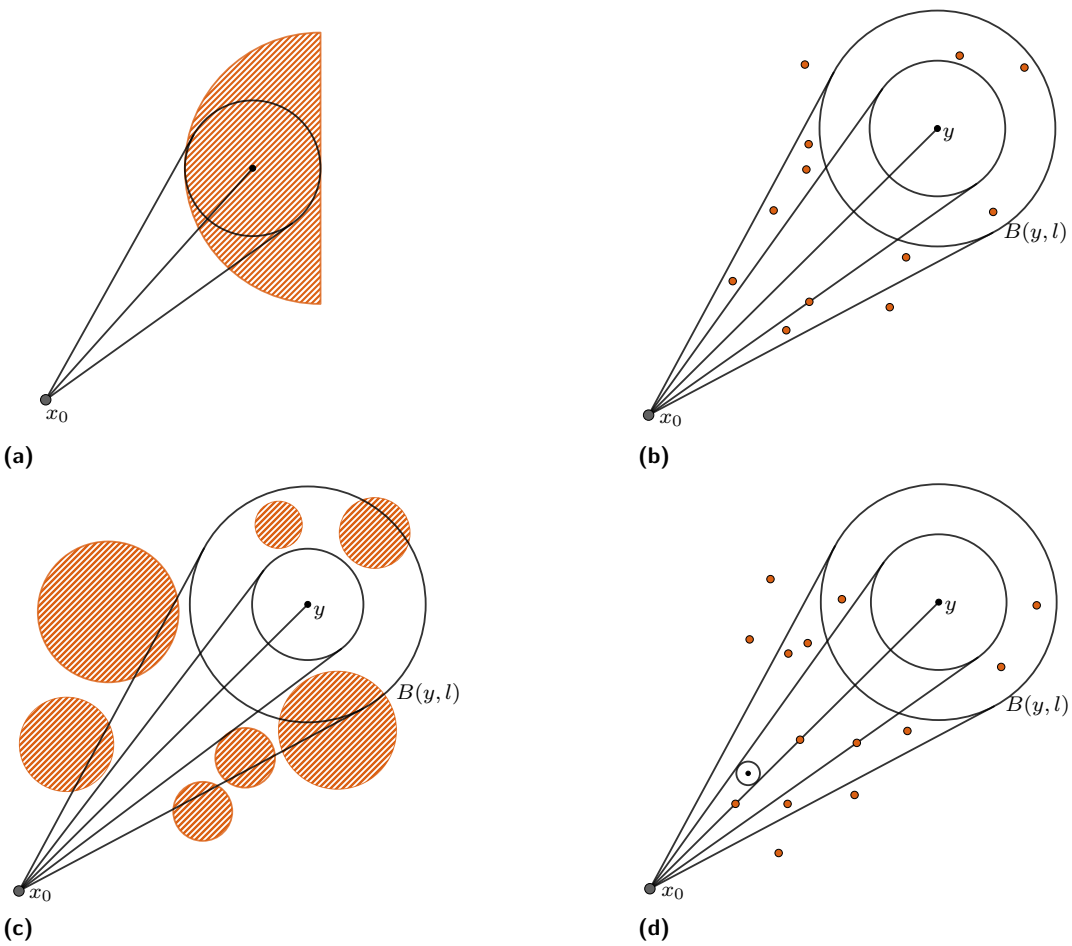

Figure 7 Some variants of Algorithm 2. a) Starting from x_0, the robot has to move inside the shaded region. b) Starting from x_0, the robot has to move inside $B(y,l)$ avoiding point obstacles. There are no obstacles on the line segment joining x_0 and y. c) Starting from x_0, the robot has to move inside $B(y,l)$ avoiding disk shaped obstacles. The line segment joining x_0 and y does not intersect any obstacle. d) Starting from x_0, the robot has to move inside $B(y,l)$ avoiding point obstacles. There are some obstacles on the line segment joining x_0 and y.

to a point inside $B(y,l)$ via a trajectory which lies inside $Cone(x_0, B(y,l))$. A pseudocode description of an algorithm that solves the problem is presented in Algorithm 2. Proof of correctness of the algorithm can be found in the full version [3] of the paper.

Algorithm 2 Algorithm for moving through a safe zone.

Input : A point y on the plane and a distance l
$r \leftarrow$ myself
if $d(r,y) \geq l$ **then**
  if $d(r,y) = l$ *or* $\frac{l}{d(r,y)} \geq sin(error_a(r,y))$ **then**
    Move to y
  else
    $p \leftarrow$ point on $seg(r,y)$ so that $\frac{l}{d(r,y)} = sin(error_a(r,p))$
    Move to p

We now discuss some variants of the problem. They can be solved using the movement strategy of Algorithm 2 subject to some modifications.

1. Suppose that the robot r is required to move inside some region other than a disk. Assume that the region is enclosed by some line segments and circular arcs. We can easily solve this problem using the same movement strategy, e.g., by fixing some disk $B(y, l)$ inside the region and following Algorithm 2. See Fig. 7a.
2. Now consider the situation where the robot r, starting from x_0, have to get inside a disk $B(y, l)$, but there are some point obstacles that it needs to avoid. Let $\mathcal{O} \subset \mathbb{R}^2$ be the set of obstacles. However, there are no obstacles on $\overline{seg}(x_0, y)$. Again a similar approach will work. Instead of $B(y, l)$, the robot r just needs to consider $B(y, l')$ where $l' \in (0, l]$ is the largest possible length such that $Cone(r, B(y, l')) \cap \mathcal{O} = \emptyset$. See Fig. 7b.
3. Instead of point obstacles, now consider disk shaped obstacles. Assume that none of the obstacles intersect $\overline{seg}(x_0, y)$. The same approach as in the previous problem would work here too. See Fig. 7c.
4. Now again consider point obstacles, but this time there might be some obstacles lying on $\overline{seg}(x_0, y)$. Let $\mathcal{O}' = \mathcal{O} \cap \overline{seg}(x_0, y)$. The robot will move to a point $x' \in Cone(x_0, B(y, l))$ so that there is no obstacle on $\overline{seg}(x', y)$. For this, it will move so that it reaches a point in $Cone(x_0, B(y, l')) \setminus \overline{seg}(x_0, y)$ where $l' \in (0, l]$ is the largest possible length such that $Cone(x_0, B(y, l')) \cap (\mathcal{O} \setminus \mathcal{O}') = \emptyset$. See Fig. 7d.