Arbitrary Pattern Formation on Infinite Grid by Asynchronous Oblivious Robots ⋆

Kaustav Bose[0000−0003−3579−1941], Ranendu Adhikary[0000−0002−9473−2645], Manash Kumar Kundu[0000−0003−4179−8293], and Buddhadeb Sau

Department of Mathematics, Jadavpur University, Kolkata, India
{kaustavbose.rs, ranenduadhikary.rs, manashkrkundu.rs }@jadavpuruniversity.in, bsau@math.jdvu.ac.in

Abstract. The ARBITRARY PATTERN FORMATION problem asks to design a distributed algorithm that allows a set of autonomous mobile robots to form any specific but arbitrary geometric pattern given as input. The problem has been extensively studied in literature in continuous domains. This paper investigates a discrete version of the problem where the robots are operating on a two dimensional infinite grid. The robots are assumed to be autonomous, identical, anonymous and oblivious. They operate in Look-Compute-Move cycles under a fully asynchronous scheduler. The robots do not agree on any common global coordinate system or chirality. We have shown that a set of robots can form any arbitrary pattern, if their starting configuration is asymmetric.

Keywords: Distributed algorithm · Autonomous robots · Arbitrary Pattern Formation · Grid · Asynchronous · Look-Compute-Move cycle.

1 Introduction

1.1 Motivation

Distributed coordination of autonomous mobile robot systems has attracted considerable research interest in recent years owing to its potential applications in a wide range of real-world problems. The problem of forming an arbitrary geometric pattern is a fundamental coordination task for autonomous robot swarms. The pattern formation problem has been extensively investigated in continuous domains under different assumptions. In the continuous setting, the robots are assumed to be able to execute accurate movements in arbitrary directions and by arbitrarily small amounts. Hence, even in densely crowded situations, the robots can maneuver avoiding collisions. Certain models also permit the robots to move along curved trajectories, in particular, circumference of a circle. For robots with weak mechanical capabilities, it may not be possible to execute such intricate movements with precision. This motivates us to consider the problem in a grid based terrain where the movements of the robots are restricted only

---

⋆ This is the full version of the paper, with the same title and authors, that was accepted in the 13th International Conference and Workshops on Algorithms and Computation (WALCOM 2019), February 27 - March 02, 2019, Guwahati, India

along grid lines and only by a unit distance in each step. Grid type floor layouts can be easily implemented in real life robot navigation systems using magnets or optical guidances.

1.2 Earlier Works

The ARBITRARY PATTERN FORMATION problem was first studied by Suzuki and Yamashita [12,13] on the Euclidean plane. In these papers, a complete characterization of the class of formable patterns has been provided for autonomous and anonymous robots with an unbounded amount of memory. They characterized the class of formable patterns by using the notion of *symmetricity* which is essentially the order of the cyclic group that acts on the initial configuration. In [8], Flocchini et. al. investigated the solvability of the problem for fully asynchronous and oblivious robots. Initially, the robots are in arbitrary positions, with the only requirement that no two robots are in the same position. They showed that if the robots have no common agreement on coordinate system, they cannot form an arbitrary pattern. If the robots have one-axis agreement, then any odd number of robots can form an arbitrary pattern, but an even number of robots cannot, in the worst case. If the robots agree on both X and Y axes, then any set of robots can form any pattern. They also proved that it is possible to elect a leader for $n \geq 3$ robots if it is possible to form any pattern. In [6,7], the authors studied the relationship between ARBITRARY PATTERN FORMATION and LEADER ELECTION among robots in asynchronous scheduler. They provided algorithms to form an arbitrary pattern starting from any geometric configuration wherein the leader election is possible. More precisely, their solutions work for four or more robots with chirality and for at least five robots without chirality. Combined with the result in [8], they deduced that ARBITRARY PATTERN FORMATION and LEADER ELECTION are equivalent, i.e., it is possible to solve ARBITRARY PATTERN FORMATION for $n \geq 4$ with chirality (resp. $n \geq 5$ without chirality) if and only if LEADER ELECTION is solvable. While all the previous works considered robots with unlimited visibility, Yamauchi et. al. [14] first studied the problem with limited visibility. Randomized pattern formation algorithms were studied in [2,15]. In [5], Das et al. investigated the problem of forming a sequence of patterns in a given order. In [3,9], the problem was studied allowing the pattern to have multiplicities. In [4,9] the so-called EMBEDDED PATTERN FORMATION problem was studied where the pattern to be formed is provided as a set of visible points in the plane. Recently in [10], the pattern formation problem was studied on an infinite grid for robots with limited visibility. The problem was studied in synchronous setting for robots with constant size memory, and having a common coordinate system. Furthermore, robots were given a fixed point on the grid so that they can form a connected configuration containing it. Other specific types of formation problems that have been studied in the infinite grid set up, are the *Gathering* problem [11], i.e., the point formation problem and the *Mutual Visibility* problem [1], where a set of opaque robots have to form a pattern in which no three robots are collinear.

The paper is organized as follows. In Section 2, some basic definitions and a formal description of the model and the problem are presented. In Section 3, we present and solve a preliminary problem that will be used in the main algorithm, which is described with formal proof in Section 4.

2 Model and Definition

2.1 The Model

Robots: The robots are *autonomous* (there is no central control), *homogeneous* (they execute the same distributed algorithm), *anonymous* (they have no unique identifiers), *identical* (they are indistinguishable by their appearance) and *oblivious* (they have no memory of past configurations and previous actions). The robots cannot explicitly communicate with each other. The robots have *global visibility* which means that they can observe the entire grid and the positions of all the robots. The robots do not have access to any common global coordinate system. In particular, they do not have a common notion of direction or chirality. Each robot has its own local view of the world with respect to its local Cartesian coordinate system. All the robots are initially positioned on distinct grid points.

Movement: The movement of the robots are restricted only along grid lines from one grid point to one of its four neighboring grid points. Traditionally in discrete domains, robot movements are assumed to be instantaneous. For simplicity of analysis, we also assume the movements to be instantaneous. This implies that the robots are always seen on grid points, not on edges. However, our strategy will also work without this assumption (by asking the robots to wait i.e, do nothing, if they see a robot on a grid edge).

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

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

2.2 Basic Geometric Definitions

Consider a team of a finite number of robots placed on the vertices of a simple undirected connected graph $G = (V, E)$. Define a function $f : V \longrightarrow \mathbb{N} \cup \{0\}$, where $f(v)$ is the number of robots on the vertex v[1]. The pair (G, f) is called a *configuration of robots on G*, or simply a *configuration*. Given a configuration of robots $\mathscr{C}$, let $\mathscr{R}$ denote the smallest grid-aligned rectangle that contains all the robots.

---

[1] Since we have assumed that the robots are initially positioned on distinct grid points and our algorithm guarantees collisionless movements, $f(v)$ is always either 0 or 1.

An *automorphism* of a graph $G = (V, E)$ is a bijection $\varphi : V \longrightarrow V$ such that for all $u, v \in V$, u, v are adjacent if and only if $\varphi(u), \varphi(v)$ are adjacent. The set of all automorphisms of G forms a group, called the *automorphism group* of G and is denoted by $Aut(G)$. The definition of automorphism of graphs can be extended to robot configurations on graphs. An *automorphism of a configuration* (G, f) is an automorphism φ of G such that $f(v) = f(\varphi(v))$ for all $v \in V$. The set of all automorphisms of (G, f) also forms a group that will be denoted by $Aut(G, f)$. We shall refer to an automorphism of a configuration as a *symmetry*. We shall call a symmetry *trivial* if $\varphi(v) = v$, for all $v \in V$ with $f(v) \neq 0$. If a configuration admits no non-trivial symmetries, then it is called an *asymmetric configuration*, and otherwise, a *symmetric configuration*.

An infinite path is the graph $P = (\mathbb{Z}, E)$, where $E = \{(i, i+1) \mid i \in \mathbb{Z}\}$. An infinite grid can be defined as the Cartesian product $G = P \times P$. Assume that the infinite grid G is embedded in the Cartesian plane $\mathbb{R}^2$. It is not difficult to see that $Aut(G)$ is generated by three types of automorphisms: translations, reflections and rotations. A translation shifts all the vertices of G by the same amount. Since a configuration (G, f) has only finite number of robots, it is not difficult to see that $Aut(G, f)$ has no translations. Reflections are defined by an axis of reflection. The axis can be horizontal or vertical or diagonal. The angle of rotation can be of 90 or 180 degrees, and the center of a rotation can be a vertex, or the center of an edge, or the center of the unit square.

The solvability of the arbitrary pattern formation problem depends on the symmetries of the initial configuration of the robots. This paper exclusively considers only asymmetric initial configurations. Some impossibility results regarding symmetric configurations are briefly discussed in Section 5.

2.3 The Arbitrary Pattern Formation Problem

A swarm of k robots is arbitrarily deployed on the vertices of the infinite grid. We assume that the initial configuration $\mathscr{C}_{init}$ is asymmetric, and no two robots are in the same position. The goal of the ARBITRARY PATTERN FORMATION problem is to design a distributed algorithm that guides the robots to form an arbitrary geometric pattern $\mathscr{C}_{target}$. The pattern $\mathscr{C}_{target}$ is a set of k (distinct) vertices in the grid given in an arbitrary Cartesian coordinate system. The pattern $\mathscr{C}_{target}$ is given to all robots in the system as input. Due to absence of a common global coordinate system, the robots decide that the pattern is formed when their present configuration becomes 'similar' to $\mathscr{C}_{target}$ with respect to translations, rotations, reflections. We say that a pattern formation algorithm is *collision-free*, if, at any time t, there are no two robots that occupy the same grid point. Avoiding collisions is a necessary requirement of the problem under this model. This is because, if two robots at any point in time, occupy the same grid point, they can not be deterministically separated thereafter, as they both execute the same deterministic algorithm.

3 Pattern Formation on a Finite Grid

In this section, we will discuss a related problem that will be used in the main algorithm. Consider a set of k robots deployed on an $m \times n$ finite grid. Starting from any

arbitrary (symmetric or asymmetric) configuration, they are required to form a given arbitrary pattern. Unlike our original problem, we assume that the robots agree on a common global coordinate system. The input $\mathscr{C}_{target}$ is also given in this coordinate system. Hence, the given input corresponds to a fixed set $\mathscr{T}$ of k grid points on the grid and our problem is to place a robot on each of these grid points. All the other assumptions from our original problem, stated in Section 2.1, are retained.

We shall first consider the case where $m = 1$, i.e., the grid is just a discretized line segment. Since the robots have a common global coordinate system, they agree on left and right. Hence, in the starting configuration, the robots can be labeled as $r_1, \ldots, r_k$ from left to right. If we can devise a swap-free (the act of two adjacent robots exchanging their positions is called a swap) and collision-free movement strategy, then the labels will remain unchanged throughout the algorithm. Note that, in the asynchronous setting, a collision-free algorithm is necessarily swap-free. We can also label the grid points in $\mathscr{T}$ as $t_1, \ldots, t_k$ from left to right. Our strategy is to simply ask each r_i to go to t_i. In order to avoid collisions, a robot will move to an adjacent grid point only if it is empty. A pseudocode description of the strategy is given in Algorithm 1.

Algorithm 1: Pattern formation on a $1 \times n$ grid

---
1 **Procedure** PFONPATH()
2       $s \in \{left, right\}$
3       $r_i \leftarrow$ me
4   **if** *I am not at t_i* **then**
5   **if** *t_i is on my s* **then**
6               $u \leftarrow$ the adjacent grid point on my s
7   **if** *u is empty* **then**
8   Move to u

Theorem 1. *Algorithm* PFONPATH() *is correct.*

Proof. We first show that the algorithm does not lead to a collision. A collision can only occur in a situation shown in Fig. 1a, where the algorithm asks both r_i and r_{i+1} to move to B. This implies that t_i is somewhere on the right of A, while t_{i+1} is on the left of C. But this is impossible, since t_i and t_{i+1} are two distinct grid points with t_{i+1} on the right of t_i.

It remains to show that following this strategy, each r_i will be able to reach t_i. The algorithm can only fail if a deadlock is created. This can only happen if we have two robots, r_i and r_{i+1}, adjacent to each other, where r_i wants to move towards right and r_{i+1} is either at t_{i+1} or wants to move towards left. Again this is impossible by the same arguments as earlier. □

Now we consider the general $m \times n$ finite grid. An $m \times n$ finite grid can be seen as a coiled up path as shown in Fig. 1b. To be precise, an $m \times n$ grid has a spanning subgraph isomorphic to the finite path P_{mn}. But there are many such spanning subgraphs. The common global coordinate system allows the robots to agree on a particular subgraph as
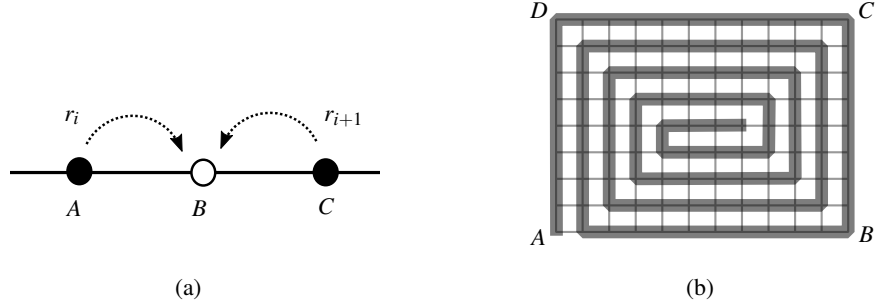
(a) (b)

Fig. 1: a) Illustration supporting the proof of Theorem 1. b) A coiled up path in a finite grid.

shown in Fig. 1b. Hence, pattern formation on a finite grid reduces to pattern formation on a path, which can be solved by algorithm PFONPATH().

4   The Main Algorithm

Consider a configuration $\mathscr{C}$, where $\mathscr{R} = ABCD$ is an $m \times n$ rectangle with $|AB| = n \geq m = |AD|$ (See Fig. 2). Here, the size of a side is defined as the number of grid points on it. If all robots in $\mathscr{C}$ lie on one grid line, then $\mathscr{R}$ is just a line segment. In this case, when we say $\mathscr{R} = ABCD$, it is to be regarded as a $1 \times n$ 'rectangle' with $A = D$, $B = C$ and $|AD| = |BC| = 1$. Let us first assume that $ABCD$ is a non-square rectangle with $|AB| = n > m = |AD| > 1$. We associate a binary string of length mn to each corner of $\mathscr{R}$. The binary string associated to a corner A is defined as follows. Scan the grid from A along the shorter side AD to D and sequentially all grid lines parallel to AD in the same direction. For
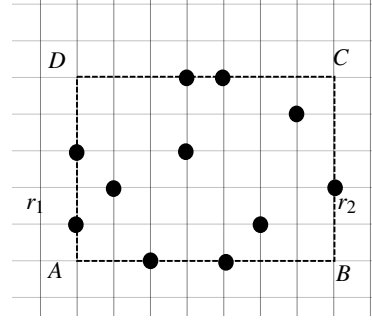

Fig. 2: In this configuration, the lexicographically largest string is $\lambda_{AD} = 010100001000100000001011000010100 00000010001000$. The head and the tail are respectively r_1 and r_2.

each grid point, put a 0 or 1 according to whether it is empty or occupied. We denote this string by λ_{AD}. The three other strings λ_{BC}, λ_{CB} and λ_{DA} are defined similarly. If $ABCD$ is a square, i.e., $m = n$, then we have to associate two strings to each corner. In that case, the two sequences associated with A will be denoted by λ_{AD} and λ_{AB}. If any two of these strings are equal, then it implies that the configuration has a (reflectional or rotational) symmetry. Hence, if the configuration is asymmetric, then all the strings are distinct and we can find a unique lexicographically largest string. Assume that λ_{AD} is the lexicographically largest string. Then A will be called the *leading corner*. Once we have the unique lexicographically largest string λ_{AD}, the robots can agree on a com-

mon coordinate system as follows. The leading corner A is taken as origin and X-axis $= \overrightarrow{AB}$, Y-axis $= \overrightarrow{AD}$. Unless mentioned otherwise, any asymmetric configuration $\mathscr{C}$ will be expressed in this coordinate system. In the case where $m = 1$, λ_{AD} and λ_{DA} essentially refers to the same string. Hence, in this case, we have only two binary strings to compare. Again, if they are equal then the configuration is symmetric. Hence, if the configuration is asymmetric [2], then we shall have a leading corner, say $A = D$. Then A will be taken as origin and X-axis $= \overrightarrow{AB}$. But there will be no agreement on the Y-axis. However, as all the robots lie on the X-axis, the points in $\mathscr{C}$ can still be unambiguously expressed in coordinates.

Therefore, we see that in an asymmetric configuration, all the robots can agree on a common global coordinate system. By 'up' (resp. 'right') and 'down' (resp. 'left'), we shall refer to the positive and negative direction of Y (resp. X) axis of this coordinate system respectively. Also, given any asymmetric configuration $\mathscr{C}$, the robots corresponding to the first and the last 1 in the lexicographically largest string, will be called the *head* and the *tail* respectively. The remaining robots will be called *interior* robots. $\mathscr{C}'$ and $\mathscr{C}''$ will denote the sets $\mathscr{C} \setminus \{tail\}$ and $\mathscr{C} \setminus \{head, tail\}$ respectively.

The configuration $\mathscr{C}_{target}$, given to the robots as an input, is expressed in some arbitrary coordinate system. We can take the smallest enclosing rectangle of $\mathscr{C}_{target}$, call it $\mathscr{R}_{target}$. Assume that $\mathscr{R}_{target}$ is an $M \times N$ rectangle, with $N \geq M$. Now associate binary strings to its corners in the same manner as we did for $\mathscr{R}$. We shall assume that $\mathscr{C}_{target}$ is expressed in a coordinate system where the origin is the leading corner and the positive Y axis is along the side corresponding to the lexicographically largest string. No generality is lost, as the robots can always perform such a coordinate transformation on the input. However, unlike the previous case, we may not have a unique lexicographically largest string. This is because, the configuration $\mathscr{C}_{target}$ can have symmetries. In that case, for the coordinate transformation, we have to choose one among the largest strings to define the coordinate system. Notice that any choice leads to the same set of values. Therefore, in general, we shall assume that the origin of the coordinate system of $\mathscr{C}_{target}$ is *one of the* leading corners, and the positive Y axis is along the side corresponding to *one of the* lexicographically largest strings. We shall call this coordinate system the *canonical coordinate system*. Given $\mathscr{C}_{target}$ in the canonical coordinate system, we define $h_{target}, t_{target} \in \mathscr{C}_{target}$ as the points, corresponding to the first and the last 1 of the binary string that starts from the origin and goes along the Y axis respectively. Also, define $\mathscr{C}'_{target} = \mathscr{C}_{target} \setminus \{t_{target}\}$ and $\mathscr{C}''_{target} = \mathscr{C} \setminus \{h_{target}, t_{target}\}$.

We can logically divide the algorithm into seven phases. The starting configuration of the robots can fall into any one of the phases. These phases will be described in detail in the following subsections. Since the robots are oblivious, in each LOOK-COMPUTE-MOVE cycle, it has to infer from the perceived configuration, which phase it is currently in. It does so by checking if certain conditions are fulfilled or not. These conditions can be expressed in terms of Boolean variables listed in Fig. 3.

The main algorithmic difficulty of the problem arises from the restrictions imposed on the movements of the robots. In the continuous setting the robots can freely move in any direction by arbitrarily small amounts and in some models, along any curve.

---

[2] In the $m = 1$ case, we already have a reflectional symmetry with respect to $\overleftrightarrow{AB}$. But this is a trivial symmetry, and is to be ignored by our definition of asymmetric configurations.

C_0	$\mathscr{C} = \mathscr{C}_{target}$
C_1	$\mathscr{C}' = \mathscr{C}'_{target}$
C_2	Y-coordinate of the tail in $\mathscr{C}$ = Y-coordinate of t_{target} in $\mathscr{C}_{target}$
C_3	$n \geq \max\{M, m\} + 2$
C_4	$n \geq 2 \cdot \max\{N, H\}$, where H is the length of the horizontal side of the smallest enclosing rectangle of $\mathscr{C}'$
C_5	The head in $\mathscr{C}$ is at the origin
C_6	$m \geq \max\{M, V\} + 1$, where V is the length of the vertical side of the smallest enclosing rectangle of $\mathscr{C}'$
C_7	$\mathscr{C}'' = \mathscr{C}''_{target}$
C_8	$\mathscr{C}'$ has a non-trivial reflectional symmetry with respect to a horizontal line

Fig. 3: The Boolean variable on the left is true if and only if the condition on the right is satisfied.

Therefore, techniques used in the previous works on continuous space (e.g., [7,8,6]) are not immediately portable in the discrete setting. Collision less movement is a major challenge in the grid model due to movement restriction. To resolve this, the tail expands the initial smallest enclosing rectangle (in Phase 1 and 3) making enough room for the interior robots to reconfigure themselves inside the rectangle without colliding. Our main idea is to utilize the asymmetry of the configuration to reach an agreement on a coordinate system, and try to keep the coordinate system invariant during the movements. To achieve this, in the first three phases, the head is put at the origin and the smallest enclosing rectangle is large enough so that the interior robots are confined in an appropriately small finite subgrid. Any movement by the interior robots restricted inside the finite subgrid keeps the coordinate system unaltered. So in Phase 4, the interior robots will rearrange themselves inside the finite subgrid to partially form the given pattern. In the final three phases, the head and the tail will move to their prescribed positions. Despite the apparent simplicity of the final three phases, designing movements is somewhat complicated as the coordinate system may change or the agreement in the coordinate system may be lost in some cases in the final phases.

4.1 Phase 1

A robot infers that it is in Phase 1 if $\neg(C_1 \wedge C_2) \wedge \neg(C_3 \wedge C_4)$ is true [3]. In this case, the tail will move to the right and all other robots will remain static. Our aim is to make both C_3 and C_4 true.

Theorem 2. *If we have an asymmetric configuration $\mathscr{C}$ in phase 1 at some time t, then*

1. *after one move by the tail towards right, the new configuration is still asymmetric and the coordinate system remains unchanged.*
2. *after one move by the tail towards right, we have $\neg(C_1 \wedge C_2) = true$.*
3. *after finite number of moves by the tail, we shall have $(C_3 \wedge C_4) = true$.*

---

[3] $A \wedge B$ is true if and only if both A and B are true. $\neg A$ is true if and only if A is false.

4. *after finite number of moves by the tail, Phase 1 completes with* $\neg(C_1 \wedge C_2) \wedge (C_3 \wedge C_4) = true$.

Proof. 1) Let the smallest enclosing rectangle at time t be $\mathscr{R}^{old} = ABCD$, with $|AD| = m$ and $|AB| = n$, $n \geq m$. Since $\mathscr{C}$ is asymmetric, we must have $n > 2$. Let λ_{AD} be the lexicographically largest string. Hence the tail r is on the edge BC. After a move by the tail towards right, the smallest enclosing rectangle becomes $\mathscr{R}^{new} = AB'C'D$, with r now on the edge $B'C'$.

We already have $\lambda_{AD}^{old} > \lambda_{DA}^{old}$. Suppose that in $\mathscr{C}$, r is the only robot on BC. Then it is easy to see that $\lambda_{AD}^{new} > \lambda_{DA}^{new}$. Now assume that there are multiple robots on BC. Suppose that r corresponds to the xth and yth term in λ_{AD}^{old} and λ_{DA}^{old} respectively.

Case-1 : Let $x = y$. This means that m is odd and r is on the middle point of BC. Since the xth term is the last non-zero term in λ_{AD}^{old}, we must have $\lambda_{AD}^{old}|_{x-1} > \lambda_{DA}^{old}|_{x-1}$, where $\lambda|_p$ is the string obtained by taking the first p terms of λ. Therefore, $\lambda_{AD}^{new}|_{x-1} > \lambda_{DA}^{new}|_{x-1}$, which implies $\lambda_{AD}^{new} > \lambda_{DA}^{new}$.

Case-2 : Let $x < y$. Let $x = m(n-1) + b$. Since $x < y$, we have $b \leq \lfloor \frac{n}{2} \rfloor$. Since r is the tail, the $(mn - m + 1)$th to $(mn - m + b)$th terms of λ_{DA}^{old} are all 0. The same is true for λ_{DA}^{new}. Since there were multiple robots on BC at time t, there is at least one 1 among the $(mn - m + 1)$th to $(mn - m + b)$th terms in λ_{AD}^{new}. Hence, $\lambda_{AD}^{new} > \lambda_{DA}^{new}$.

Case-3 : If $x > y$, then r is encountered strictly earlier, as we scan $\mathscr{R}^{old}$ according to the string λ_{DA}^{old}, than λ_{AD}^{old}. This implies that $\lambda_{DA}^{old}|_y > \lambda_{DA}^{new}|_y$ and $\lambda_{AD}^{old}|_y = \lambda_{AD}^{new}|_y$. But as $\lambda_{AD}^{old} > \lambda_{DA}^{old}$, we have $\lambda_{AD}^{old}|_y \geq \lambda_{DA}^{old}|_y$. Hence, we have $\lambda_{AD}^{new}|_y > \lambda_{DA}^{new}|_y$, and so $\lambda_{AD}^{new} > \lambda_{DA}^{new}$.

Therefore, we have $\lambda_{AD}^{new} > \lambda_{DA}^{new}$. Now we compare λ_{AD}^{new} and $\lambda_{C'B'}^{new}$. Clearly, r corresponds to the first 1 in λ_{CB}^{old} (and also $\lambda_{C'B'}^{new}$). Suppose it is the xth term λ_{CB}^{old}. If the first 1 in λ_{AD}^{old} appears before the xth term, then we easily have $\lambda_{AD}^{new} > \lambda_{C'B'}^{new}$. So let the xth term be the first 1 in λ_{AD}^{old}. Let the second 1 in λ_{AD}^{old} be the yth term. Again, if the second 1 in λ_{CB}^{old} appears beyond the yth term, then we are done. So assume that the second 1 in λ_{CB}^{old} is the yth term. Hence, $\lambda_{AD}^{old}|_y = \lambda_{CB}^{old}|_y$. Now $\lambda_{C'B'}^{new}$ is obtained by inserting a string of 0's of length m in λ_{CB}^{old} after the xth term. Hence, we clearly have $\lambda_{AD}^{new}|_y = \lambda_{AD}^{old}|_y > \lambda_{C'B'}^{new}|_y$. Thus, $\lambda_{AD}^{new} > \lambda_{C'B'}^{new}$. We can similarly show that $\lambda_{AD}^{new} > \lambda_{B'C'}^{new}$.

Hence, λ_{AD}^{new} is lexicographically strictly larger than λ_{DA}^{new}, $\lambda_{B'C'}^{new}$ and $\lambda_{C'B'}^{new}$. Since, $\mathscr{R}^{new} = AB'C'D$ is a non-square rectangle, we only have these four binary strings to consider. Hence, we can clearly see that the new configuration is still asymmetric and the coordinate system is unchanged.

2) Since only the tail moves, the value of C_1 remains unchanged. As the tail moves to the right, its Y-coordinate and hence C_2 is also unchanged. Therefore, $\neg(C_1 \wedge C_2)$ remains true after the move.

3) It follows from 1) that the tail remains invariant throughout phase 1. Clearly, after a finite number of moves by the tail towards right, both C_3 and C_4 will become true.

4) Follows from 2) and 3). $\qquad\square$

4.2 Phase 2

The algorithm is in phase 2, when either $C_3 \wedge C_4 \wedge \neg C_5 \wedge \neg C_7$ or $\neg C_2 \wedge C_3 \wedge C_4 \wedge \neg C_5 \wedge C_7$ is true. Our aim is to take the head to the origin. Hence, in this phase, the head will move down towards the origin.

Theorem 3. *If we have an asymmetric configuration $\mathscr{C}$ in phase 2 at some time t, then*

1. *after one move by the head downwards, the new configuration is still asymmetric and the coordinate system is unchanged.*
2. *after finite number of moves by the head, C_5 becomes true.*
3. *after finite number of moves by the head, phase 2 completes with $C_3 \wedge C_4 \wedge C_5 \wedge \neg C_7$ or $\neg C_2 \wedge C_3 \wedge C_4 \wedge C_5 \wedge C_7$ true.*

4.3 Phase 3

The algorithm is in phase 3, if $C_3 \wedge C_4 \wedge C_5 \wedge \neg C_6 \wedge \neg C_7$ is true. In this phase, there are two cases to consider. The robots will check if C_8 is true or false. Let us first consider the case where C_8 is false. In this case, the tail will move upwards and the rest will remain static.

Theorem 4. *If we have an asymmetric configuration $\mathscr{C}$ in phase 3 at some time t with $C_8 = false$, then*

1. *after one move by the tail upwards, the new configuration is still asymmetric and the coordinate system is unchanged.*
2. *after one move by the tail upwards, we still have $C_4 \wedge C_5 \wedge \neg C_7 = true$.*
3. *after finite number of moves by the tail, we shall have $C_3 \wedge C_4 \wedge C_5 \wedge C_6 \wedge \neg C_7 = true$.*

Proof. Let the smallest enclosing rectangle at time t be $\mathscr{R}^{old} = ABCD$, with $|AD| = m$ and $|AB| = n$, $n > m$. Let λ_{AD}^{old} be the lexicographically largest string.

Case 1 : Suppose the smallest enclosing rectangle remains unchanged after the move, i.e., $\mathscr{R}^{new} = ABCD$. Since $C_4 \wedge C_5$ is true, it is easy to see that $\lambda_{AD}^{new} > \lambda_{BC}^{new}$ and $\lambda_{AD}^{new} > \lambda_{CB}^{new}$. Finally, $\lambda_{AD}^{new} > \lambda_{DA}^{new}$ follows from the fact that $C_4 \wedge C_5 \wedge \neg C_8$ is true. To see this, note that $n \geq 2H$ as C_4 is true. This implies that the first $p = m \lfloor \frac{n}{2} \rfloor$ terms of both λ_{AD}^{old} and λ_{DA}^{old} contains all terms corresponding to the robots in $\mathscr{C}'$. Since C_8 is false, we must have $\lambda_{AD}^{old}\big|_p > \lambda_{DA}^{old}\big|_p$, and hence $\lambda_{AD}^{new}\big|_p > \lambda_{DA}^{new}\big|_p$. So, $\lambda_{AD}^{new} > \lambda_{DA}^{new}$.

Case 2 : Suppose that the tail is at C at time t, and after a move upwards, we have $\mathscr{R}^{new} = ABC'D'$, $|AD'| = m+1$. Since $n \geq m+2 \Rightarrow n > m+1 \Rightarrow |AB| > |AD'|$, the smallest enclosing rectangle is still not a square. This implies that we still need to consider only four binary strings and it is easy to see that $\lambda_{AD'}$ is strictly largest among them.

Therefore, we have proved 1). It is easy to see 2), i.e., $C_4 \wedge C_5 \wedge \neg C_7$ is true after the move. However, C_3 might become false after the move described in case 2. Then the phase changes to phase 1. Since, before the move, we had $C_3 = true$, $n \geq m+2$ and $n \geq M+2$. If C_3 becomes false after the move, we have $n < \max\{M, m+1\}+2 \Rightarrow n \leq m+2$

(as $n \geq M+2$). Since we also have $n \geq m+2$, we get $n = m+2 \Rightarrow m = n-2$. If $M \leq m$, after the move, C_6 becomes true as $m+1 \geq V+1$ and $m+1 \geq M+1$. If $m < M$, then $n-2 < M \Rightarrow n < M+2$, a contradiction. Therefore, after the move, we have $C_6 =$ true, i.e., we have $\neg C_3 \wedge C_4 \wedge C_5 \wedge C_6 \wedge \neg C_7 =$ true. So we are in phase 1, and the tail will move rightwards. After one rightwards move, we shall have $C_3 \wedge C_4 \wedge C_5 \wedge C_6 \wedge \neg C_7 =$ true. $\qquad\square$

(a) Case 1

(b) Case 2

Fig. 4: Illustration of phase 3 with $C_8 =$ true.

Now assume that C_8 is true, i.e., $\mathscr{C}'$ has a non-trivial reflectional symmetry with respect to a horizontal line L. Again, let the smallest enclosing rectangle be $\mathscr{R}^{old} = ABCD$, with $|AD| = m$ and $|AB| = n$, $n > m$. Let λ_{AD}^{old} be the lexicographically largest string. Let E be the point of BC where it intersects with L. Let the smallest enclosing rectangle of $\mathscr{C}'$ be $\mathscr{R}' = AB'C'D'$. There are two cases to consider: $D \neq D'$ (Case 1) and $D = D'$ (Case 2).

Case 1: See Fig.4a. In this case, the tail r will move upwards. Clearly, after finite number of moves by the tail $C_6 =$ true is achieved. However, C_3 may become false, but is recovered after one move rightwards as explained previously.

Case 2: See Fig.4b. Since the configuration is asymmetric and λ_{AD}^{old} is the largest string, the tail r must be in $[B,E)$. In this case, r will move downwards. When r goes below B, the coordinate system flips. The new coordinate system has origin at D, X-axis $= \overrightarrow{DC}$ and Y-axis $= \overrightarrow{DA}$. Clearly, the case is reduced to the situation similar to case 1.

Theorem 5. *If we have an asymmetric configuration $\mathscr{C}$ in phase 3 at some time t with C_8 true, then after finite number of moves by the tail, we have $C_3 \wedge C_4 \wedge C_5 \wedge C_6 \wedge \neg C_7 =$ true.*

4.4 Phase 4

If the configuration satisfies $C_3 \wedge C_4 \wedge C_5 \wedge C_6 \wedge \neg C_7 = $ true, then the algorithm is in phase 4. In this phase, the head and the tail will remain static. Let $\mathscr{F}$ be the subgrid of $\mathscr{R}$ of size $(m-1) \times \lfloor \frac{n}{2} \rfloor$ with coinciding bottom-left corners (See Fig. 5). $\mathscr{F}$ can be considered as a finite line segment $\mathscr{L}$ as shown in Fig. 5. The interior robots will execute the protocol PFONPATH() on $\mathscr{L}$ to achieve $\mathscr{C}'' = \mathscr{C}''_{target}$, i.e., $C_7 = $ true.

Fig. 5: A configuration in phase 4.

Theorem 6. *If we have an asymmetric configuration $\mathscr{C}$ at some time t, with $C_3 \wedge C_4 \wedge C_5 \wedge C_6 \wedge \neg C_7 = $ true, then*

1. *after any move by an interior robot according to* PFONPATH()*, the new configuration is still asymmetric and the coordinate system is unchanged.*
2. *after any move by an interior robot according to* PFONPATH()*, we still have $C_3 \wedge C_4 \wedge C_5 \wedge C_6 = $ true.*
3. *after finite number of moves by the interior robots, we shall have $C_7 = $ true.*
4. *after finite number of moves by the interior robots, Phase 4 completes with $C_3 \wedge C_4 \wedge C_5 \wedge C_6 \wedge C_7 = $ true.*

Proof. It is easy to see that $C_3 \wedge C_4 \wedge C_5 \wedge C_6$ is true after any move. Let $\mathscr{R}^{old} = \mathscr{R}^{new} = ABCD$. Let λ_{AD}^{old} be the lexicographically largest string in $\mathscr{R}^{old}$. $C_5 = $ true implies that A is occupied. $C_4 \wedge C_6 = $ true implies that B, D are empty and C is occupied by the tail. Hence, $\lambda_{AD}^{new} > \lambda_{BC}^{new}$ and $\lambda_{AD}^{new} > \lambda_{DA}^{new}$. Also, it implies from $C_4 = $ true that $\lambda_{AD}^{new} > \lambda_{CB}^{new}$. The rest are easily seen. □

4.5 Phase 5

The algorithm is in phase 5, if $\neg C_2 \wedge C_3 \wedge C_4 \wedge C_5 \wedge C_7$ is true. In this phase, the tail will move along the vertical grid line in order to make C_2 true.

If we have an asymmetric configuration $\mathscr{C}$ which is in phase 5, then depending on whether C_8 is true or false, there are two cases to consider.

Let the smallest enclosing rectangle of $\mathscr{C}$ be $\mathscr{R} = ABCD$, with $|AD| = m$ and $|AB| = n$, $n > m$. Let A be the leading corner, and hence X-axis $= \overrightarrow{AB}$ and Y-axis $= \overrightarrow{AD}$. Now, let us plot the points of $\mathscr{C}_{target}$ in this coordinate system. Except h_{target} and t_{target}, all other

points of $\mathscr{C}_{target}$ are occupied by the robots. Let the smallest enclosing rectangle of $\mathscr{C}'$ be $\mathscr{R}' = AB'C'D'$ (See Fig. 6). Hence, the tail is currently on BC, and all the remaining robots are inside the region $AB'C'D'$.

Case-1 First, assume that C_8 is true. Since the head is at A, D' is also occupied due to the symmetry. Let C'' be the grid point where the grid lines $\overleftrightarrow{D'C'}$ and $\overleftrightarrow{BC}$ intersect. Let C''' be the middle point of BC''. C''' is a grid point if $|BC''|$ is odd. If the tail is on BC'', then $C = C''$ and $D = D'$. Note that the tail can not be on $[C''',C'']$, because then we shall have $\lambda_{DA} \geq \lambda_{AD}$. Hence, the tail is on $[B,C''']$ or (C'',∞).

In this phase, we want to make C_2 true. This means that the tail needs to go to the grid point on $\overleftrightarrow{BC''}$ that is on the same horizontal line with t_{target}. Let us call this point $\tilde{t}_{target}$. Consider the case where $\tilde{t}_{target} \in [B,C'']$. In this case, the upper left corner of the smallest enclosing rectangle of $\mathscr{C}_{target}$ is D', which is occupied by a robot. Since the input is given in canonical coordinates, the bottom left corner (origin) of the smallest enclosing rectangle of $\mathscr{C}_{target}$, i.e., A, must be the leading corner. Therefore, A must be occupied in the final configuration. Since A is already occupied, it implies that C_1 is currently true. Also note that $\tilde{t}_{target} \notin (C''',C'')$, as A is the leading corner in $\mathscr{C}_{target}$. Hence, $\tilde{t}_{target}$ is on $[B,C''']$ or (C'',∞).

Case 1A: tail $\in [B,C''')$ and $\tilde{t}_{target} \in [B,C''']$

The tail will move towards $\tilde{t}_{target}$. During the movements, the coordinate system remains invariant. However, if $\tilde{t}_{target}$ is at C''', a horizontal symmetry will be created when it reaches $\tilde{t}_{target}$.

Case 1B: tail $\in (C'',\infty)$ and $\tilde{t}_{target} \in (C'',\infty)$

The tail will move towards $\tilde{t}_{target}$. Again it is easy to see that the coordinate system remains invariant during the movements.

Case 1C: tail $\in (C'',\infty)$ and $\tilde{t}_{target} \in [B,C''']$

In this case, the tail will move downwards. When r reaches C'', the coordinate system flips. The new coordinate system has origin at D', X-axis $= \overrightarrow{D'C''}$ and Y-axis $= \overrightarrow{D'A}$. In the new coordinate system, r requires to place itself in $[C'',C''']$. Hence, the case is reduced to the situation similar to case 1A. Thus r achieves $C_2 = $ true without going beneath C'''.

Case 1D: tail $\in [B,C''')$ and $\tilde{t}_{target} \in (C'',\infty)$

In this case, the tail will move downwards. When r goes beneath B, the coordinate system flips. The new coordinate system has origin at D', X-axis $= \overrightarrow{D'C''}$ and Y-axis $= \overrightarrow{D'A}$. Clearly, the case is reduced to the situation similar to case 1B.

Case-2 Now assume that C_8 is false. It is easy to see that where ever $\tilde{t}_{target}$ is on $\overleftrightarrow{BC}$, the binary string attached to A is lexicographically strictly largest as the tail moves towards it. Hence, the movement of the tail in phase 5 does not change the coordinate system. Clearly, after finite number of moves, C_2 becomes true. Then phase 5 is completed with $C_2 \wedge C_3 \wedge C_4 \wedge C_5 \wedge C_7$ true.

Theorem 7. *If we have an asymmetric configuration $\mathscr{C}$ in phase 5 at some time t, then after finite number of steps phase 5 completes with $C_2 \wedge C_3 \wedge C_4 \wedge C_5 \wedge C_7 = $ true.*
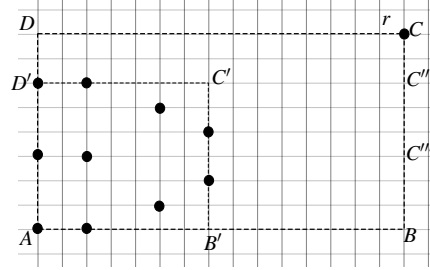
Fig. 6: Illustration of case 1 of phase 5.

4.6 Phase 6

If we have $\neg C_1 \wedge C_2 \wedge C_3 \wedge C_4 \wedge C_7 = \text{true}$, then the algorithm is in phase 6. In this case, the head will move towards h_{target}.
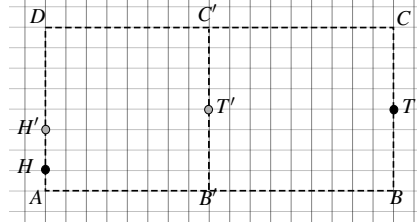

Fig. 7: Illustration of phase 6.

Consider an asymmetric configuration $\mathscr{C}$ which is in phase 6. Let $ABCD$ be the smallest enclosing rectangle, with λ_{AD} being the (strictly) largest string. Let H and T be the position of the head and the tail respectively. H and T are clearly on AD and BC respectively. Plot the points of $\mathscr{C}_{target}$ on the grid with respect to the current coordinate system (X-axis $= \overrightarrow{AB}$ and Y-axis $= \overrightarrow{AD}$). The smallest enclosing rectangle of these points is $AB'C'D$ (See Fig. 7). Let H' and T' be the points h_{target} and t_{target}. Therefore, if the head moves from H to H' and the tail moves from T to T', then the given pattern is formed. H' and T' are clearly on AD and $B'C'$ respectively, with T and T' being on the same horizontal line.

The aim of this phase is to move the head from H to H'. Let $\mathscr{C}_h$ be the configuration obtained from $\mathscr{C}$, if the the head moves from H to H'. Since the input is given in canonical coordinates, λ_{AD} is the lexicographically largest string in $\mathscr{C}_{target}$. This implies that λ_{AD} is lexicographically largest in $\mathscr{C}_h$. Recall that, λ_{AD} may not be strictly largest, as $\mathscr{C}_{target}$ may have some symmetries. In particular, if $\mathscr{C}_{target}$ has a reflectional symmetry with respect to a horizontal axis, then $\mathscr{C}_h$ also has the same. Then λ_{AD} and λ_{DA} are both lexicographically largest in $\mathscr{C}_h$. In any case, as the head moves from H to H', λ_{AD} remains lexicographically strictly largest until it reaches H'.

Theorem 8. *Let $\mathscr{C}$ be an asymmetric configuration in phase 6. Then after finite number of moves by the head, phase 6 completes with $C_1 \wedge C_2 \wedge C_3 \wedge C_4 =$ true, and hence, $\neg C_0 \wedge C_1 \wedge C_2 =$ true. The final configuration has a horizontal reflectional symmetry, if $\mathscr{C}_{target}$ also has the same.*

4.7 Phase 7

Finally, the algorithm is in phase 7 if we have $\neg C_0 \wedge C_1 \wedge C_2 =$ true. Suppose that the tail is at T'. The tail will move horizontally towards $T = t_{target}$. Note that a configuration $\mathscr{C}$ in phase 7 can have a reflectional symmetry with respect to a horizontal axis. Let λ_{AD} be a largest string in the configuration. Since λ_{AD} is also a largest string in the final configuration, it is easy to see that, when the tail is moving from T' to T, λ_{AD} remains the largest string (may be jointly with λ_{DA}) until it reaches T.

Theorem 9. *If we have a configuration $\mathscr{C}$ at some time t, with $\neg C_0 \wedge C_1 \wedge C_2 =$ true, then after finite number of moves by the tail, C_0 becomes true.*

It is not difficult to verify (See Appendix) that any configuration with $C_0 =$ false, belongs to one of the seven phases that we have discussed. From the results we have proved, it follows that starting from any asymmetric configuration, our algorithm can form any given pattern in finite time (See Appendix for a phase transition diagram of our proposed algorithm). Hence, we can conclude the following theorem.

Theorem 10. ARBITRARY PATTERN FORMATION *is solvable in ASYNC from any asymmetric initial configuration.*

5 Concluding Remarks

We have proved that any arbitrary pattern is formable by a set of asynchronous robots if the initial configuration is asymmetric. The immediate course of future research would be to characterize the patterns formable from symmetric configurations. It can be proved that if a configuration $\mathscr{C}$ admits symmetry φ such that no robot lies on the axis of reflection or the center of rotation, then any configuration formable from $\mathscr{C}$ necessarily has the same symmetry φ. This is however not true, if the axis of reflection or the center of rotation contains a robot r. The symmetry may be broken by asking the robot r to move. However, this is not straightforward especially in a crowded situation. It would be also interesting to consider randomized algorithms. Another direction of future research would be to extend our work for patterns allowing multiplicities.
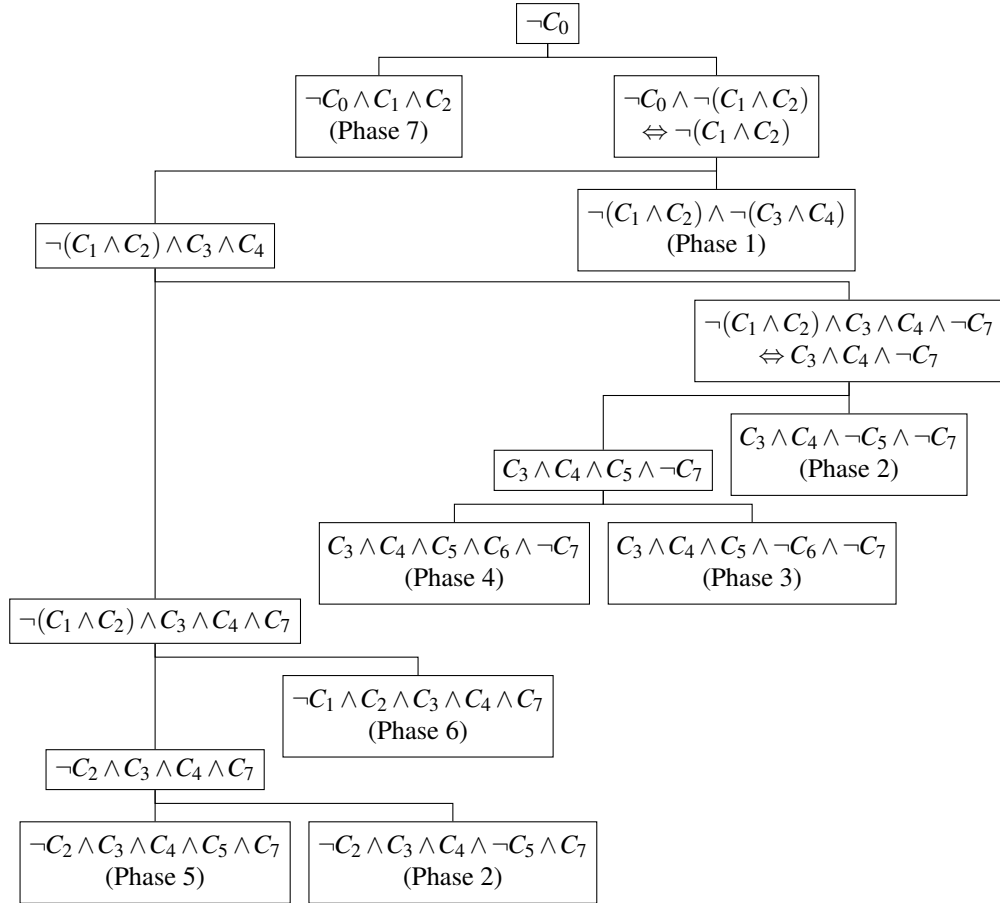
References

1. Adhikary, R., Bose, K., Kundu, M.K., Sau, B.: Mutual visibility by asynchronous robots on infinite grid. In: 14th International Symposium on Algorithms and Experiments for Wireless Networks (ALGOSENSORS 2018), 23-24 August 2018, Helsinki, Finland ((Forthcoming))
2. Bramas, Q., Tixeuil, S.: Brief announcement: Probabilistic asynchronous arbitrary pattern formation. In: Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, Chicago, IL, USA, July 25-28, 2016. pp. 443–445 (2016), doi: 10.1145/2933057.2933074
3. Cicerone, S., Di Stefano, G., Navarra, A.: Asynchronous arbitrary pattern formation: the effects of a rigorous approach. Distributed Computing pp. 1–42 (2018), doi: 10.1007/s00446-018-0325-7
4. Cicerone, S., Di Stefano, G., Navarra, A.: Embedded pattern formation by asynchronous robots without chirality. Distributed Computing pp. 1–25 (2018), doi: 10.1007/s00446-018-0333-7
5. Das, S., Flocchini, P., Santoro, N., Yamashita, M.: Forming sequences of geometric patterns with oblivious mobile robots. Distributed Computing **28**(2), 131–145 (2015), doi: 10.1007/s00446-014-0220-9
6. Dieudonné, Y., Petit, F., Villain, V.: Leader election problem versus pattern formation problem. CoRR **abs/0902.2851** (2009), http://arxiv.org/abs/0902.2851
7. Dieudonné, Y., Petit, F., Villain, V.: Leader election problem versus pattern formation problem. In: Distributed Computing, 24th International Symposium, DISC 2010, Cambridge, MA, USA, September 13-15, 2010. Proceedings. pp. 267–281 (2010), doi: 10.1007/978-3-642-15763-9_26
8. Flocchini, P., Prencipe, G., Santoro, N., Widmayer, P.: 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
9. Fujinaga, N., Yamauchi, Y., Ono, H., Kijima, S., Yamashita, M.: Pattern formation by oblivious asynchronous mobile robots. SIAM J. Comput. **44**(3), 740–785 (2015), doi: 10.1137/140958682
10. Lukovszki, T., Meyer auf der Heide, F.: Fast collisionless pattern formation by anonymous, position-aware robots. In: Principles of Distributed Systems - 18th International Conference, OPODIS 2014, Cortina d'Ampezzo, Italy, December 16-19, 2014. Proceedings. pp. 248–262 (2014), doi: 10.1007/978-3-319-14472-6_17
11. Stefano, G.D., Navarra, A.: Gathering of oblivious robots on infinite grids with minimum traveled distance. Inf. Comput. **254**, 377–391 (2017), doi: 10.1016/j.ic.2016.09.004
12. Suzuki, I., Yamashita, M.: Distributed anonymous mobile robots. In: SIROCCO'96, The 3rd International Colloquium on Structural Information & Communication Complexity, Siena, Italy, June 6-8, 1996. pp. 313–330 (1996)
13. Suzuki, I., Yamashita, M.: Distributed anonymous mobile robots: Formation of geometric patterns. SIAM J. Comput. **28**(4), 1347–1363 (1999), doi: 10.1137/S009753979628292X
14. Yamauchi, Y., Yamashita, M.: Pattern formation by mobile robots with limited visibility. In: Structural Information and Communication Complexity - 20th International Colloquium, SIROCCO 2013, Ischia, Italy, July 1-3, 2013, Revised Selected Papers. pp. 201–212 (2013), doi: 10.1007/978-3-319-03578-9_17
15. Yamauchi, Y., Yamashita, M.: Randomized pattern formation algorithm for asynchronous oblivious mobile robots. In: Distributed Computing - 28th International Symposium, DISC 2014, Austin, TX, USA, October 12-15, 2014. Proceedings. pp. 137–151 (2014), doi: 10.1007/978-3-662-45174-8_10

Appendix

A Different Phases of the Main Algorithm

From the following tree, it can be easily seen that any configuration with $C_0 = $ false belongs to one of the seven phases described in the algorithm. It is also evident that the phases are mutually disjoint.

B   Phase Transition Diagram of the Main Algorithm

A general scheme of transition between different phases of the algorithm is shown in the following diagram. Observe that the only cycle in the graph is the one involving phase 1 and phase 3. It has been shown in the proof of Theorem 4 that this does not create a livelock. Clearly, starting from any phase, the algorithm terminates with $C_0 =$ true.