Space and Move-Optimal Arbitrary Pattern Formation on Infinite Rectangular Grid by Oblivious Robot Swarm

Avisek Sharma
Department of Mathematics, Jadavpur University, India

Satakshi Ghosh
Department of Mathematics, Jadavpur University, India

Pritam Goswami
Department of Mathematics, Jadavpur University, India

Buddhadeb Sau
Department of Mathematics, Jadavpur University, India

Abstract

Arbitrary Pattern Formation (APF) is a fundamental coordination problem in swarm robotics. It requires a set of autonomous robots (mobile computing units) to form an arbitrary pattern (given as input) starting from any initial pattern. This problem has been extensively investigated in continuous and discrete scenarios, with this study focusing on the discrete variant. A set of robots is placed on the nodes of an infinite rectangular grid graph embedded in the euclidean plane. The movements of each robot is restricted to one of the four neighboring grid nodes from its current position. The robots are autonomous, anonymous, identical, and homogeneous, and operate Look-Compute-Move cycles. In this work, we adopt the classical $\mathcal{OBLOT}$ robot model, meaning the robots have no persistent memory or explicit communication methods, yet they possess full and unobstructed visibility. This work proposes an algorithm that solves the APF problem in a fully asynchronous scheduler assuming the initial configuration is asymmetric. The considered performance measures of the algorithm are space and number of moves required for the robots. The algorithm is asymptotically move-optimal. Here, we provide a definition of space complexity that takes the visibility issue into consideration. We observe an obvious lower bound $\mathcal{D}$ of the space complexity and show that the proposed algorithm has the space complexity $\mathcal{D}+4$. On comparing with previous related works, we show that this is the first proposed algorithm considering $\mathcal{OBLOT}$ robot model that is asymptotically move-optimal and has the least space complexity which is almost optimal.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Distributed algorithms, Oblivious robots, Optimal algorithms, Swarm robotics, Space optimization, and Rectangular grid

Digital Object Identifier 10.4230/LIPIcs.SAND.2024.20

Related Version *Full Version*: `https://doi.org/10.48550/arXiv.2309.11190` [14]

Funding The first and the third authors are supported by the University Grants Commission (UGC), India. The second author is supported by the West Bengal State Government Fellowship Scheme. The fourth author is supported by the Science and Engineering Research Board (SERB), India

1 Introduction

Swarm robotics involves a group of simple computing units referred to as *robots* that operate autonomously without having any centralized control. Moreover, the robots are generally anonymous (no unique identifier), homogeneous (all robots execute the same algorithm), and identical (physically indistinguishable). Generally on activation, a robot first takes a snapshot of its surroundings. This phase is called the Look phase. Then based on the snapshot an inbuilt algorithm determines a destination point. This phase is called the COMPUTE phase. Finally, in the MOVE phase it moves towards the computed destination. These three phases together are called a LOOK-COMPUTE-MOVE (LCM) cycle of a robot.

Through collaborative efforts, these robot swarms can accomplish different tasks such as gathering at a specific point, configuring into predetermined patterns, navigating networks, etc. Presently, the field of robotics research is witnessing significant enthusiasm for swarm robots. The inherent decentralized characteristics of these algorithms provide swarm robots with a notable advantage, as distributed algorithms are both easily scalable and more resilient in the face of errors. Furthermore, swarm robots boast a multitude of real-world applications, including but not limited to tasks like area coverage, patrolling, network maintenance, etc.

In order to accomplish specific tasks, robots require some computational capabilities, which can be determined by various factors such as memory, communication, etc. With respect to memory and communication, the literature identifies two primary robot models. The first one is called the classical $\mathcal{OBLOT}$ model. In this model, the robots are devoid of persistent memory and communication abilities. Another robot model is the $\mathcal{LUMI}$ model where the robots are equipped with a finite number of lights that can take a finite number of different colors. These colors serve as persistent memory (as a robot can see its own color) and communication architecture (as the colors of lights are visible to all other robots). The responsibility for activating robots rests with an entity referred to as the *Scheduler*. Within the existing literature, three primary types of schedulers emerge: *Fully-Synchronous* ($\mathcal{FSYNC}$), *Semi-Synchronous* ($\mathcal{SSYNC}$), and *Asynchronous* ($\mathcal{ASYNC}$). In the case of fully synchronous and semi-synchronous schedulers, time is partitioned into rounds of uniform length. The duration of the LOOK, COMPUTE, and MOVE phases for all activated robots are identical. Under a fully-synchronous scheduler, all robots become active at the onset of each round, but in a semi-synchronous setup, not all robots may activate simultaneously in a given round. In an asynchronous scheduler, round divisions are absent. At any given moment, a robot can be either idle or engaged in any of the LOOK, COMPUTE, or MOVE phases. The duration of these phases and the spans of robot idleness are finite but unbounded.

The primary focus of this study is to solve the Arbitrary Pattern Formation (APF) problem on an infinite rectangular grid while minimizing spatial utilization. The APF problem involves a group of robots situated within an environment, aiming to create a designated pattern. This pattern is conveyed to each robot as a set of points within a coordinate system as an input. This problem has been extensively studied in the euclidean plane ([2, 3, 4, 6, 7, 8, 16, 17]) and also on a continuous circle [13]. Bose et al. [1] first proposed this problem on a rectangular grid. The rectangular grid is a natural discretization of the plane. To the best of our knowledge, on the discrete domain, this problem has been studied in [1, 5, 9, 10, 11, 12, 15]. In this paper, the focus is placed on an environment characterized by an infinite rectangular grid. In the upcoming subsection, we delve into the reasons behind the introduction of spatial constraint in the context of this problem.

1.1 Motivation

In the majority of previous studies, the implementation of this problem on a grid necessitates a substantial allocation of space (space of a configuration formed by a set of robots is the dimension of the smallest enclosing square of the configuration), even when both the initial and target configurations have minimal spatial requirements. This promptly gives rise to a lot of problems. To begin with, in the scenario where the grid is of bounded dimensions, it is possible that certain patterns cannot be formed, even if robots are initially located within the bounded grid and the target pattern could potentially fit within the grid. This limitation arises due to the existence of intermediate configurations that demand a spatial extent that cannot be accommodated within the confined grid. Moreover, when the spatial demand for an APF algorithm on a grid increases, the count of patterns that can be formed within a bounded grid becomes noticeably fewer compared to the count of patterns formable on the same grid with a lower space requirement. To be more specific, patterns that are "big enough" can not be formed if the space requirement is "big" on a bounded grid. So, the requirement of large space compromises better utilization of the space.

Moreover, even if complete visibility is entertained for theoretical considerations, this assumption does not hold practical validity within an unbounded environment. In the context of a bounded region, it can be applied with the premise that the environment is finite, and the entire environment falls within the visibility range of each robot. However, introducing the concept of an infinite grid disrupts this assumption. In situations where the grid lacks bounds, it is possible that due to substantial spatial requirements, certain robots might stray beyond the visibility range of others. To the best of our knowledge, there remains an absence of work that addresses the APF challenge within the constraints of limited visibility, an asynchronous scheduler, and the absence of any global coordinate agreement. Thus in this paper, the problem of APF on a grid with minimal spatial requirement has been considered.

1.2 Related Work

In the discrete setting, the problem is first studied in [1]. Here, the authors solved the problem deterministically on an infinite rectangular grid with $\mathcal{OBLOT}$ robots in an asynchronous scheduler. Later in [5], the authors studied the problem on a regular tessellation graph. In [1], authors count the total required moves asymptotically and also give an asymptotic lower bound for the move complexity, i.e., total number of moves required to solve the problem. In [5], authors did not count the total number of moves required for their proposed algorithm. In [9], the authors provided two deterministic algorithms for solving the problem in an asynchronous scheduler. The first algorithm of [9] solves the APF problem for the $\mathcal{OBLOT}$ model. The move complexity of this algorithm matches the asymptotic lower bound given in [1]. Thus, this algorithm is asymptotically move-optimal. The second algorithm of [9] solves the problem for the $\mathcal{LUMI}$ model, and this algorithm is asymptotically move-optimal. Further authors showed that the algorithm is time-optimal, i.e., the number of epochs (a time interval in which each robot activates at least once) to complete the algorithm is asymptotically optimal. In [11], the authors provided a deterministic algorithm for solving the problem with opaque (non-transparent) point robots in the $\mathcal{LUMI}$ model with an asynchronous scheduler assuming one-axis agreement. In [10], the authors proposed two randomized algorithms for solving the APF problem in an asynchronous scheduler. The second algorithm works for the $\mathcal{OBLOT}$ model. This algorithm is asymptotically move-optimal and time-optimal. The randomization in this algorithm is only used to break any present symmetry in the initial configuration. If the initial configuration is asymmetric then the algorithm is deterministic. The first algorithm works for opaque point robots with the $\mathcal{LUMI}$ model. This algorithm is also asymptotically move-optimal and time-optimal. In [12], the authors solve the problem with opaque fat robots (robots having nontrivial dimension) with the $\mathcal{LUMI}$ model in an asynchronous scheduler assuming one-axis agreement. In [15], the authors provide an asymptotically move-optimal algorithm solving this problem with robots in the $\mathcal{LUMI}$ model. The work also considered a special requirement and showed that the algorithm is space-optimal. In the next section, we formally state the space complexity of an algorithm and discuss the space complexity of the mentioned works.

1.3 Space Complexity of APF Algorithms in Rectangular Grid

In [15], the authors considered the total space required to execute an algorithm. In Definition 1, we define the space complexity of an algorithm executed by a set of robots on a rectangular grid. Before that let's define the dimension of a rectangle, vertices of which are on some grid nodes, as $m \times n$ if the rectangle has m horizontal grid lines and n vertical grid lines.

▶ **Definition 1.** *The space complexity of an algorithm executed by a set of robots on a rectangular grid is the minimum dimension of the squares (whose sides are parallel with the grid lines) such that no robot steps out of the square throughout the execution of the algorithm.*

Let the smallest enclosing rectangle (SER), the sides of which are parallel to grid lines, of the initial configuration and pattern configuration formed by the robots, respectively, have dimensions $m \times n$ ($m \geq n$) and $m' \times n'$ ($m' \geq n'$). Let $\mathcal{D} = \max\{m, n, m', n'\}$. Then the minimum space complexity for an algorithm to solve the APF problem is $\mathcal{D}$. Definition 1 assigns a real number to the space complexity that makes it easy to compare different APF algorithms. But consider an APF algorithm that takes a space enclosed by an axis aligned rectangle of dimension $p \times q$. if $M = \max\{m, m'\}$ and $N = \max\{n, n'\}$, then the APF algorithm is better (as far as space is concerned) if p is closer to M and q is closer to N.

Space Complexity of the Previous APF Algorithms

(*$\mathcal{OBLOT}$ model APF algorithms*) The algorithm proposed in [1] has space-complexity at least $2\mathcal{D}$ in the worst case as one of the leaders, named tail moves far away from the rest of the configuration. The first algorithm proposed in [9] is for the $\mathcal{OBLOT}$ model. It requires the robots to form a compact line. The space complexity of these algorithms is $\mathcal{D}^2$ in the worst case. The second randomized algorithm in [10] is for the $\mathcal{OBLOT}$ model. In this algorithm, the leader robot moves upwards far away from the rest of the configuration. Thus, it has a space complexity of at least $30\mathcal{D}$ in the worst case.

(*$\mathcal{LUMI}$ model APF algorithms*) The second algorithm proposed in [9] is for the $\mathcal{LUMI}$ model. This algorithm requires a step-looking configuration where each robot occupies a unique vertical line. Therefore, the space complexity of the algorithm can be $\mathcal{D}^2$ in the worst case. This algorithm needs each robot to have a light with three distinct colors. The first randomized algorithm in [10] for $\mathcal{LUMI}$ model has space-complexity at least $\mathcal{D} + 2$. The authors also did not count the number of lights and colors required for the robots. With a closer look, we observe that this algorithm uses at least 31 distinct colors. Further, deterministic APF algorithms proposed in [11, 12] solved it for obstructed visibility. These works also need the robots to form a compact line, hence the space complexity of these algorithms is $\mathcal{D}^2$ in the worst case. The proposed algorithm in [15] has space-complexity $\mathcal{D} + 1$ and it requires three distinct colors.

We say that the first algorithm proposed in [10] and algorithm proposed in [15] are *almost* space-optimal, as the space-complexity is of the form $\mathcal{D} + c$, $\mathcal{D}$ is a lower bound of the space-complexity and c is a constant independent of $\mathcal{D}$. If we consider the rectangle to measure the space, then a rectangle of dimension $M \times N$ is minimally required to solve the APF problem. The first algorithm in [10] and the algorithm in [15] takes space enclosed by rectangle of dimension $(M + 2) \times (N + 2)$ and $(M + 1) \times N$ respectively. We can consider these algorithms as so far the best APF algorithms as far as space complexity is concerned. For the rest of the algorithms one dimension of the rectangle that encloses the required space shoots up twice (algorithm in [1]) or 30 times (2^{nd} algorithm in [10]) or squares (algorithm in [9, 11, 12]). For the rectangle version, if an APF algorithm takes a space of enclosing rectangle of dimension $(M + c_1) \times (N + c_2)$, where c_1 and c_2 are constants independent of M and N, then the algorithm is said to be almost optimal. The challenge of this work is to reconfigure the (oblivious and silent) robots in an optimal space avoiding the occurrence of symmetric configurations and collision among robots while keeping the number of movements asymptotically optimal.

Our Contribution

First a deterministic algorithm for solving APF in an infinite discrete line is presented. Then exploiting that algorithm this manuscript presents a deterministic algorithm for solving APF in an infinite rectangular grid which is almost space-optimal as well as asymptotically move-optimal. Precisely, the space complexity for the algorithm is $\mathcal{D} + 4$ and this algorithm takes a space enclosing the rectangle of dimension $(M + 4) \times (N + 1)$. The move-complexity of the algorithm is $O(k\mathcal{D})$[1], where k is the number of robots. The robot model is the classical $\mathcal{OBLOT}$ model and the scheduler is fully asynchronous. To the best of our knowledge so far, this is the first deterministic algorithm solving APF problem in the $\mathcal{OBLOT}$ robot model that has the least space-complexity and optimal move-complexity (See Table 1 for comparison with the previous works). The architecture of the description of the algorithm and correctness proof are motivated from [1].

Table 1 Comparison table.

Work	Model	Visibility	Deterministic/ Randomised	Space complexity
[1]	$\mathcal{OBLOT}$	Unobstructed	Deterministic	$\geq 2\mathcal{D}$
1^{st} algorithm in [9]	$\mathcal{OBLOT}$	Unobstructed	Deterministic	$\mathcal{D}^2$
2^{nd} algorithm in [10]	$\mathcal{OBLOT}$	Unobstructed	Randomised[2]	$\geq 30\mathcal{D}$
2^{nd} algorithm in [9]	$\mathcal{LUMI}$	Unobstructed	Deterministic	$\mathcal{D}^2$
1^{st} algorithm in [10]	$\mathcal{LUMI}$	Obstructed	Randomised	$\geq \mathcal{D}+2$
[11]	$\mathcal{LUMI}$	Obstructed	Deterministic	$\mathcal{D}^2$
[12]	$\mathcal{LUMI}$	Obstructed (fat robot)	Deterministic	$\mathcal{D}^2$
[15]	$\mathcal{LUMI}$	Unobstructed	Deterministic	$\mathcal{D} + 1$
Algorithm in this work	$\mathcal{OBLOT}$	Unobstructed	Deterministic	$\mathcal{D} + 4$

2 Model and Problem Statement

Robot

The robots are assumed to be identical, anonymous, autonomous, and homogeneous. Robots are oblivious, i.e., they do not have any persistent memory to remember previous configurations or past actions. Robots do not have any explicit means of communication with other robots. The robots are modeled as points on an infinite rectangular grid graph embedded on

[1] In [10], the authors provides this tight lower bound
[2] The randomisation is only used to break any symmetry present in the initial configuration

a plane. Initially, robots are positioned on distinct grid nodes. A robot chooses the local coordinate system such that the axes are parallel to the grid lines and the origin is its current position. Robots do not agree on a global coordinate system. The robots do not have a global sense of clockwise direction. A robot can only rest on a grid node. Movements of the robots are restricted to the grid lines, and through a movement, a robot can choose to move to one of its four adjacent grid nodes.

Look-Compute-Move Cycle

A robot has two states: sleep/idle state and active state. On activation, a robot operates in Look-Compute-Move (LCM) cycles, which consist of three phases. In the Look phase, a robot takes a snapshot of its surroundings and gets the position of all the robots. We assume that the robots have full, unobstructed visibility. In the Compute phase, the robots run an inbuilt algorithm that takes the information obtained in the Look phase and obtains a position. The position can be its own or any of its adjacent grid nodes. In the Move phase, the robot either stays still or moves to the adjacent grid node as determined in the Compute phase.

Scheduler

The robots work asynchronously. There is no common notion of time for robots. Each robot independently gets activated and executes its LCM cycle. The time length of LCM cycles, Compute phases, and Move phases of robots may be different. Even the length of two LCM cycles for one robot may be different. The gap between two consecutive LCM cycles, or the time length of an LCM cycle for a robot, is finite but can be unpredictably long. We consider the activation time and the time taken to complete an LCM cycle to be determined by an adversary. In a fair adversarial scheduler, a robot gets activated infinitely often.

Grid Terrain and Configurations

Let $\mathcal{G}$ be an infinite rectangular grid graph embedded on $\mathbb{R}^2$. The $\mathcal{G}$ can be formally defined as a geometric graph embedded on a plane as $\mathcal{P} \times \mathcal{P}$, which is the cartesian product of two infinite (from both ends) path graphs $\mathcal{P}$. Suppose a set of $k > 2$ robots is placed on $\mathcal{G}$. Let f be a function from the set of vertices of $\mathcal{G}$ to $\mathbb{N} \cup \{0\}$, where $f(v)$ is the number of robots on the vertex v of $\mathcal{G}$. Then the pair $(\mathcal{G}, f)$ is said to be a *configuration* of robots on $\mathcal{G}$. For the initial configuration $(\mathcal{G}, f)$, we assume $f(v) \leq 1$ for all v.

Symmetries

Let $(\mathcal{G}, f)$ be a configuration. A *symmetry* of $(\mathcal{G}, f)$ is an automorphism ϕ of the graph $\mathcal{G}$ such that $f(v) = f(\phi(v))$ for each node v of $\mathcal{G}$. A symmetry ϕ of $(\mathcal{G}, f)$ is called *trivial* if ϕ is an identity map. If there is no non-trivial symmetry of $(\mathcal{G}, f)$, then the configuration $(\mathcal{G}, f)$ is called an *asymmetric* configuration and otherwise a *symmetric* configuration. Note that any automorphism of $\mathcal{G} = \mathcal{P} \times \mathcal{P}$ can be generated by three types of automorphisms, which are translations, rotations, and reflections. Since there are only a finite number of robots, it can be shown that $(\mathcal{G}, f)$ cannot have any translation symmetry. Reflections can be defined by an axis of reflection that can be horizontal, vertical, or diagonal. The angle of rotation can be of 90° or 180°, and the center of rotation can be a grid node, the midpoint of an edge, or the center of a unit square. We assume the initial configuration to be asymmetric. The necessity of this assumption is discussed after the problem statement.

Problem Statement

Suppose a swarm of robots is placed in an infinite rectangle grid such that no two robots are on the same grid node and the configuration formed by the robots is asymmetric. The Arbitrary Pattern Formation (APF) problem asks to design a distributed deterministic algorithm following which the robots autonomously can form any arbitrary but specific (target) pattern, which is provided to the robots as an input, without scaling it. The target pattern is given to the robots as a set of vertices in the grid with respect to a cartesian coordinate system. We assume that the number of vertices in the target pattern is the same as the number of robots present in the configuration. The pattern is considered to be formed if a configuration is formed and that is the same with target pattern up to translations, rotations, and reflections. The algorithm should be *collision-free*, i.e., no two robots should occupy the same node at any time, and two robots must not cross each other through the same edge.

Admissible Initial Configurations

We assume that in the initial configuration there is no multiplicity point, i.e., no grid node that is occupied by multiple robots. This assumption is necessary because all robots run the same deterministic algorithm, and two robots located at the same point have the same view. Thus, it is deterministically impossible to separate them afterward. Next, suppose the initial configuration has a reflectional symmetry with no robot on the axis of symmetry or a rotational symmetry with no robot on the point of rotation. Then it can be shown that no deterministic algorithm can form an asymmetric target configuration from this initial configuration. However, if the initial configuration has reflectional symmetry with some robots on the axis of symmetry or rotational symmetry with a robot at the point of rotation, then symmetry may be broken by a specific move of such robots. But making such a move may not be very easy as the robots' moves are restricted to their adjacent grid nodes only. In this work, we assume the initial configuration to be asymmetric.

3 Space-optimal Arbitrary Pattern Formation on a Grid Line

In this section, we solve this problem on a discrete straight line. Suppose we have an infinite path graph $\mathcal{P} = \{(i, i+1) \mid i \in \mathbb{Z}\}$ embedded on a straight line. Suppose k robots are placed on $\mathcal{P}$ at distinct nodes. A configuration is defined similarly as done in the previous section by considering $\mathcal{G} = \mathcal{P}$. The target pattern is given as a set of k distinct positive integers.

Leader Election and Global Coordinate Setup

We assume the initial configuration of robots does not have reflectional symmetry. First, we set up a global coordinate system that can be agreed upon by all the robots. Suppose $\mathcal{C}$ is a configuration having no reflectional symmetry. For a configuration, we define the smallest enclosing line segment (SEL) to be the smallest line segment in length that contains all the robots in the configuration. Let $\mathcal{L} = AB$ be the SEL of the configuration $\mathcal{C}$. Consider two binary strings of length $|AB|$ (the length of a line segment is the number of grid points on the line segment) called λ_A and λ_B with respect to the endpoints of $\mathcal{L}$. Let $\lambda_A = \{a_i\}_{i=1}^{|AB|}$ such that $a_i = 1$ if and only if the node on the AB line segment having distance $i-1$ from A is occupied by a robot. Similarly, we define λ_B. Since $\mathcal{C}$ has no reflectional symmetry, λ_A and λ_B are different. Therefore one of them is lexicographically smaller than the other.

Suppose λ_A is lexicographically smaller than λ_B. Then A is considered as the origin and $\overrightarrow{AB}$ is considered as the positive (right) direction. Also, the robot located at A is said to be *head* and the robot located at B is said to be *tail*. We denote $\mathcal{C} \setminus \{\text{tail}\}$ as $\mathcal{C}'$.

Target Embedding

Next we embed the pattern in the following way. Considering the integers given in the target pattern on the number line proceed similarly as done above for $\mathcal{C}$. Let $\mathcal{C}_{target}$ be the target configuration and $A'B'$ be the SEL of $\mathcal{C}_{target}$. Consider two binary strings $\lambda_{A'}$ and $\lambda_{B'}$. If both the strings are equal then the target pattern has a reflectional symmetry. In this case, embed the pattern such that all the target positions are on the right side of the origin except the left most one which is on the origin. If the strings are different then we suppose $\lambda_{A'}$ is the lexicographically smaller one. In this case, embed the pattern such that $A = A'$ and all the target positions are on the right side of the origin. After embedding, the farthest target position from the origin is said to be the *tail-target* and denoted as t_{target}. We define, $\mathcal{C}'_{target} = \mathcal{C}_{target} \setminus \{t_{target}\}$.

Proposed APF algorithm a Line

Next, we describe our proposed algorithm ApfLine. If in a snapshot of a robot, another robot is seen on an edge then the robot discards the snapshot and goes to sleep. Therefore, for simplicity, we assume that any snapshot taken by a robot contains a still configuration $\mathcal{C}$. The head never moves in the algorithm. Firstly, if $\mathcal{C}' = \mathcal{C}'_{target}$ then the tail moves to t_{target}. Otherwise, if t_{target} is at the right of the tail, then the tail moves right and the other robots remain static. If $\mathcal{C}' \neq \mathcal{C}'_{target}$, and the tail is at the t_{target} or to the right of the t_{target}, then inner robots move to make $\mathcal{C}' = \mathcal{C}'_{target}$. Let r_i be the i^{th} robot from the left and t_i be the i^{th} target position from the left. We try to design the algorithm such that r_i moves to t_i. The r_1 robot is the head and it is already on t_1. If t_i is towards the left of r_i and the left adjacent grid node is empty, then an inner robot r_i moves towards the left. If for each inner robot r_j which is not currently on t_j, t_j is at the right of the r_j, then an inner robot r_i moves right if t_i is at the right of the r_i and the right adjacent grid node is empty (The pseudo-code of the algorithm is given in Algorithm 1).

Algorithm 1 ApfLine (for a generic robot r).

if $\mathcal{C}' = \mathcal{C}'_{target}$ **then**
    tail moves towards t_{target};
else
    if t_{target} *is at the right of the tail* **then**
        tail moves towards right;
    else
        if $r = r_i$ *is an inner robot* **then**
            if t_i *is at the left of* r_i **then**
                if *left adjacent grid node is empty* **then**
                    r moves towards left;
            else if *for each inner robot* r_j *which is not currently on* t_j, t_j *is at the right of the* r_j **then**
                if t_i *is at the right of* r_i **then**
                    if *right adjacent grid node is empty* **then**
                        r moves towards right;

▶ **Theorem 2.** *From any asymmetric initial configuration, the algorithm* ApfLine *can form any target pattern on an infinite grid line within finite time under an asynchronous scheduler.*

Proof. See the full version [14]. ◀

4 The Proposed Apf Algorithm on a Rectangular Grid

4.1 Agreement of a Global Coordinate System and Target Embedding

Let $\mathcal{C}$ be an asymmetric configuration. Consider the smallest enclosing rectangle (SER) containing all the robots where the sides of the rectangle are parallel to the grid lines. Let $\mathcal{R} = ABCD$ be the SER of the configuration, a $m \times n$ rectangle with $|AD| = m \geq n = |AB|$. The length of the sides of $\mathcal{R}$ is considered to be the number of grid points on that side. If all the robots are on a grid line, then R is just a line segment. In this case, $\mathcal{R}$ is considered a $m \times 1$ "rectangle" with $A = B$, $D = C$, and $AB = CD = 1$.

For a side, say AB, of $\mathcal{R}$ we define a binary string, denoted as λ_{AB}, as follows. Let $(A = A_1, A_2, \ldots, A_m = D)$ be the sequence of grid points on the AD line segment and $(B = B_1, B_2, \ldots, B_m = C)$ be the sequence of grid points on the BC line segment. Scan the line segment AB from A to B. Then scan the line segments A_iB_i one by one in the increasing order of i. The direction of scanning the line segment A_iB_i is set as follows: Scan it from B_i to A_i if i is even and scan it from A_i to B_i if i is odd. While scanning, for each grid point put 0 or 1 according to whether it is empty or occupied, respectively (See λ_{AB} in Fig. 1).

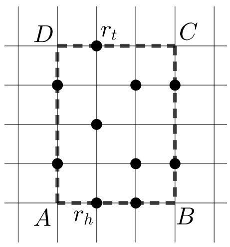

Figure 1 $ABCD$ is the SER of the configuration. $\lambda_{AB} = 01101101010011010100$ is the largest lexicographic string, and r_h and r_t are respectively the head and tail robots of the configuration.

If $m > n > 1$, then for each corner point A, B, C, and D, consider the binary strings λ_{AB}, λ_{BA}, λ_{CD} and λ_{DC}, respectively. If $m = n > 1$, then for each corner point, we have to associate two binary strings with respect to the two sides adjacent to the corner point. Then we have eight binary strings λ_{AB}, λ_{BA}, λ_{AD}, λ_{DA}, λ_{BC}, λ_{CB}, λ_{DC} and λ_{CD}. If any two strings of them are equal then it can be shown that $\mathcal{C}$ has a (reflectional or rotational) symmetry. Since $\mathcal{C}$ is asymmetric, we can find a unique lexicographically largest string (See Fig. 1). Let λ_{AB} be the lexicographically largest string, and then A is considered the *leading corner* of the configuration. The leading corner is taken as the origin, and $\overrightarrow{AB}$ is as the x-axis, and $\overrightarrow{AD}$ is as the y-axis.

If $\mathcal{R}$ is an $m \times 1$ rectangle, then λ_{AB} and λ_{BA} are the same string. Then we have two strings to compare. Since the configuration is asymmetric, these two strings must be distinct. Then we shall have a leading corner, say $A = B$. For this case, A is considered as the origin, and $\overrightarrow{AD}$ as the y-axis. There will be no agreement of the x-axis in this case but since all the robots are on the y-axis, so x-coordinate of the positions of the robots are 0 at this time.

If $\mathcal{C}$ is asymmetric then a unique string can be elected and hence, all robots can agree on a global coordinate system. By "up" ("down") and "right" ("left"), we shall refer to the positive ("negative") directions of the x-axis and y-axis of the coordinate system, respectively. The robot responsible for the first 1 in this string is considered the *head* robot of $\mathcal{C}$ and the robot responsible for the last 1 is considered the *tail* of $\mathcal{C}$. The robot other than the head and tail is termed the *inner robot*. We define, $\mathcal{C}' = \mathcal{C} \setminus \{\text{tail}\}$ and $\mathcal{C}'' = \mathcal{C} \setminus \{\text{head, tail}\}$.

Target Pattern Embedding

Here we discuss how robots are supposed to embed the target pattern when they agree on a global coordinate system. The target configuration C_{target} is given with respect to some arbitrary coordinate system. Let the $\mathcal{R}' = A'B'C'D'$ be the SER of the target pattern, an $m' \times n'$ rectangle with $|A'D'| \geq |A'B'| > 1$. We associate binary strings similarly for $\mathcal{R}'$ as done for $\mathcal{R}$. Let $\lambda_{A'B'}$ be the lexicographically largest (but may not be unique because the C_{target} can be symmetric) among all other strings for $\mathcal{R}'$. The first target position on this string $\lambda_{A'B'}$ is said to be *head-target* and denoted as h_{target} and the last target position is said to be *tail-target* and denoted as t_{target}. The rest of the target positions are called *inner target* positions. Then the target pattern is to be embedded such that A' is the origin, $\overrightarrow{A'B'}$ direction is along the positive x-axis, and $\overrightarrow{A'D'}$ direction is along the positive y-axis. Next, let us consider the case when $|A'B'| = 1$, that is when the SER of the target pattern is a line $A'D'$. Let $\lambda_{A'D'}$ be the lexicographically largest string between $\lambda_{A'D'}$ and $\lambda_{D'A'}$. Then the target is embedded in such a way that A' is at the origin and $\overrightarrow{A'D'}$ direction is along the positive y-axis. The positive x-axis direction can be decided randomly by the robot which first moves out of that line making the SER a rectangle. We define, $\mathcal{C}'_{target} = \mathcal{C}_{target} \setminus \{t_{target}\}$ and $\mathcal{C}''_{target} = \mathcal{C}_{target} \setminus \{h_{target}, t_{target}\}$.

4.2 Outline of the Proposed Algorithm

The algorithm is logically divided into seven phases[3]. A robot infers which phase it is in from the configuration visible at that time. It does so by checking which conditions in Table 2 are fulfilled. We assume that in a visible configuration, no robot is seen on an edge. We maintain such assumption by an additional condition that, if a robot sees a configuration where a robot is on an edge then discard the snapshot and go to sleep.

A Preview of the Algorithm

- Firstly the tail robot moves upwards to reach a horizontal line such that neither the horizontal line nor other horizontal lines above it contain any robot or target position (Phase I).
- Next the head robot moves left to reach the origin (Phase II).
- Then the tail robot moves a few steps upwards to remove the chance of occurrence of symmetry during the later inner robot movements (phase I).
- Then the tail robot moves rightwards to reach a vertical line such that neither the vertical line nor any vertical line to the right of it contains any robot or target positions (Phase III).

[3] The phases are assigned numerical names, yet the sequence of these numerals doesn't precisely correspond to the sequence of their execution during algorithm execution.

- After that a spanning line is considered (Figure 2) and inner robots carefully move along this line (Function `Rearrange`) to take their respective target position avoiding collision or forming any symmetric configuration (Phase IV).
- After that the tail moves horizontally to reach the vertical line that contains t_{target} (Phase V).
- Then the head robot moves horizontally to reach h_{target} (Phase VI).
- After that the tail moves vertically to reach t_{target} (Phase VII).

Table 2 Set of conditions on an asymmetric configuration $\mathcal{C}$ having SER $ABCD$ such that the origin is at A.

C_0	$\mathcal{C} = \mathcal{C}_{target}$
C_1	$\mathcal{C}' = \mathcal{C}'_{target}$
C_2	$\mathcal{C}'' = \mathcal{C}''_{target}$
C_3	x-coordinate of the tail = x-coordinate of t_{target}
C_4	There is neither any robot except the tail nor any target positions on or above H_t, where H_t is the horizontal line containing the tail
C_5	y-coordinate of the tail is odd
C_6	SER of $\mathcal{C}$ is not a square
C_7	There is neither any robot except the tail nor any target positions on or at the right of V_t, where V_t is the vertical line containing the tail
C_8	The head is at origin
C_9	If the tail and the head are relocated respectively at C and A, then the new configuration remains asymmetric
C_{10}	$\mathcal{C}'$ has a symmetry with respect to a vertical line

4.3 Detail Discussion of the Phases

Phase I

A robot infers itself in Phase I if $\neg(C_4 \wedge C_5 \wedge C_6) \wedge \neg(C_1 \wedge C_3)$ is true. In this phase, the tail moves upward and all other robots remain static. The aim of this phase is to make $C_4 \wedge C_5 \wedge C_6$ true.

Phase II

A robot infers itself in Phase II if $(C_4 \wedge C_5 \wedge C_6 \wedge \neg C_8) \wedge ((C_2 \wedge \neg C_3) \vee \neg C_2)$ is true. In this phase, the head moves towards the left, and other robots remain static. This phase aims to make C_8 true.

Phase III

A robot infers itself in Phase III if $C_4 \wedge C_5 \wedge C_6 \wedge C_8 \wedge \neg C_2 \wedge \neg C_7$ is true. The aim of this phase is to make C_7 true. In this phase, there are two cases to consider. The robots will check whether C_{10} is true or not. If C_{10} is false, then robots check whether C_9 is true or not. If C_9 is not true then the tail moves upward. Otherwise, the tail moves right or upwards in accordance with $m > n + 1$ or $m = n + 1$ (dimension of the current SER is $m \times n$ with $m \geq n$). If C_{10} is true, then the tail moves left or upwards in accordance with $m > n + 1$ or $m = n + 1$. Other robots remain static in both cases.

Phase IV

A robot infers itself in Phase IV if $C_4 \wedge C_5 \wedge C_6 \wedge C_7 \wedge C_8 \wedge \neg C_2$ is true. In this phase, the inner robots execute function `Rearrange` to make C_2 true.

Function `Rearrange`

In this function inner robots move to take their respective target positions. Let $\mathcal{C}$ be the current configuration. Let $ABCD$ be the SER of $\mathcal{C}$. According to the assumption exactly two nonadjacent vertices are occupied by robots in rectangle $ABCD$. Specifically, these two robots are the head and the tail of the configuration. Let the head and tail be located at A and C respectively. Consider the path $\mathcal{P}$ starting from A to C as illustrated in bold edges in Fig. 2. Inner robots adopt algorithm ApfLine considering this path as the line. Here, we define a robot r' at the `left` (`right`) side of r if r' is closer to the head (tail) than r in $\mathcal{P}$. Let us order the target positions. Denote h_{target} as t_1, then the next closest target position from the head in $\mathcal{P}$ as t_2. Similarly, denote the i^{th} closest target positions in $\mathcal{P}$ from the head as t_i. Note that, t_k is the t_{target}. Similarly order all the robots, $\{r_i\}_{i=1}^{k}$, where r_1 is the head and r_i $(i > 1)$ is the i^{th} closest robot from the head on $\mathcal{P}$.

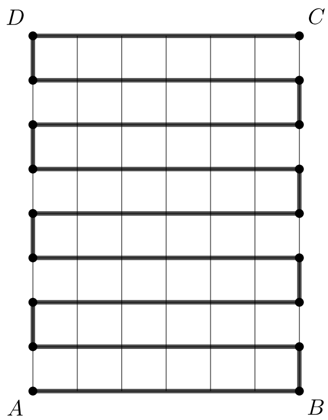

Figure 2 Path joining the nodes A and C mentioned in bold edges.

If t_i is at the `left` of r_i and there are no other robots in the sub-path of $\mathcal{P}$ starting from the position of r_i to t_i, then r_i moves to t_i. The movement strategy is described as follows. If r_i and t_i are at the same vertical (or, horizontal) line then r_i moves through the vertical (or, horizontal) line joining r_i and t_i. Suppose, r_i and t_i are not at the same vertical line or horizontal line. If the downward adjacent vertex of r_i is at the `right` of t_i then r_i moves downwards. If the downward adjacent vertex is at the `left` of t_i, then r_i moves to its `left` adjacent node on $\mathcal{P}$.

If there is no robot r_j such that t_j is at the `left` of r_j, then movements of an inner robot towards `right` start. If t_i is at the `right` of r_i, and there are no other robots in the sub-path of $\mathcal{P}$ starting from the position of r_i to t_i, then r_i moves to t_i. The movement strategy is described as follows. If r_i and t_i are at the same vertical (or, horizontal) line then r_i moves through the vertical (or, horizontal) line joining r_i and t_i. Suppose, r_i and t_i are not at the same vertical line or horizontal line. If the upward adjacent vertex of r_i is at the `left` of t_i then r_i moves upwards. If the upward adjacent vertex is at the `right` of t_i, then r_i moves to its `right` adjacent on node $\mathcal{P}$ (pseudo code of the function `Rearrange` is given Algorithm 2).

Algorithm 2 Function `Rearrange` for a robot $r = r_i$.

if *t_i is at the `left` of r_i* **then**
  if *there are no other robot in the sub-path of $\mathcal{P}$ starting from position of r_i to t_i* **then**
    if *r_i and t_i are at the same vertical (or, horizontal) line* **then**
      r_i moves towards t_i through the vertical (or, horizontal) line joining r_i and t_i;
    else
      if *the downward adjacent vertex of r_i is at the `right` of t_i* **then**
        r_i moves downwards;
      else
        r_i moves to its `left` adjacent node on $\mathcal{P}$;
else if *t_i is at the `right` of r_i* **then**
  if *there is no inner robot r_j such that t_j is at the `left` of r_j* **then**
    if *there are no other robot in the sub-path of $\mathcal{P}$ starting from position of r_i to t_i* **then**
      if *r_i and t_i are at the same vertical (or, horizontal) line* **then**
        r_i moves towards t_i through the vertical (or, horizontal) line joining r_i and t_i;
      else
        if *the upwards adjacent vertex of r_i is at the `left` of t_i* **then**
          r_i moves upwards;
        else
          r_i moves to its `right` adjacent node on $\mathcal{P}$;

Phase V

A robot infers itself in Phase V if $C_2 \wedge C_4 \wedge C_5 \wedge C_6 \wedge C_8 \wedge \neg C_3$ is true. In this phase, the tail moves horizontally to make C_3 true. Let H_t be the horizontal line containing the tail and T' be the point on the H_t that has the same x-coordinate with t_{target}. If C_{10} is not true then the tail moves horizontally towards T'. Next let C_{10} be true. Let $ABCD$ be the SER of the current configuration $\mathcal{C}$ and $AB'C'D'$ be the SER of $\mathcal{C}'$. Let C'' be the point where line $B'C'$ intersects with H_t. Let E be the point on the H_t (See Figure 3). Let the tail robot be at T. If both T and T' are at the right side of C'' or in on the line segment DE, then the tail moves towards T'. Otherwise, the tail moves leftward.

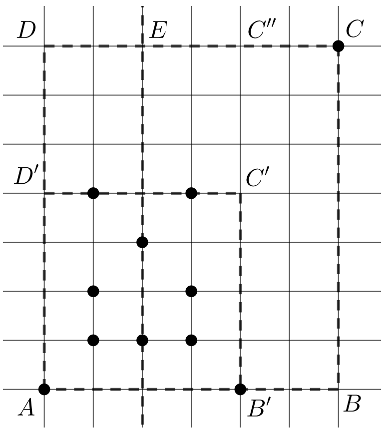

Figure 3 An image related to Phase V.

Phase VI

A robot infers itself in Phase VI if $\neg C_1 \wedge C_2 \wedge C_3 \wedge C_4 \wedge C_5 \wedge C_6$ is true. In this phase, the head moves horizontally to reach h_{target}. After the completion of this phase, $\neg C_0 \wedge C_1 \wedge C_3$ becomes true.

Phase VII

A robot infers itself in Phase VII if $\neg C_0 \wedge C_1 \wedge C_3$ is true. In this phase, the tail moves vertically to reach t_{target}.

4.4 Correctness and Performance of the Proposed Algorithm

In this section, we prove the correctness of the proposed algorithm. First, we show that any initial asymmetric configuration for which C_0 is not true falls in one of the seven phases (See Figure 4). Then we show that from any asymmetric initial configuration, the algorithm allows the configuration to satisfy $C_0 =$ true after passing through several phases (See Figure 5). The correctness proof details are omitted from this paper due to space constraint. See the full version of the paper in [14].

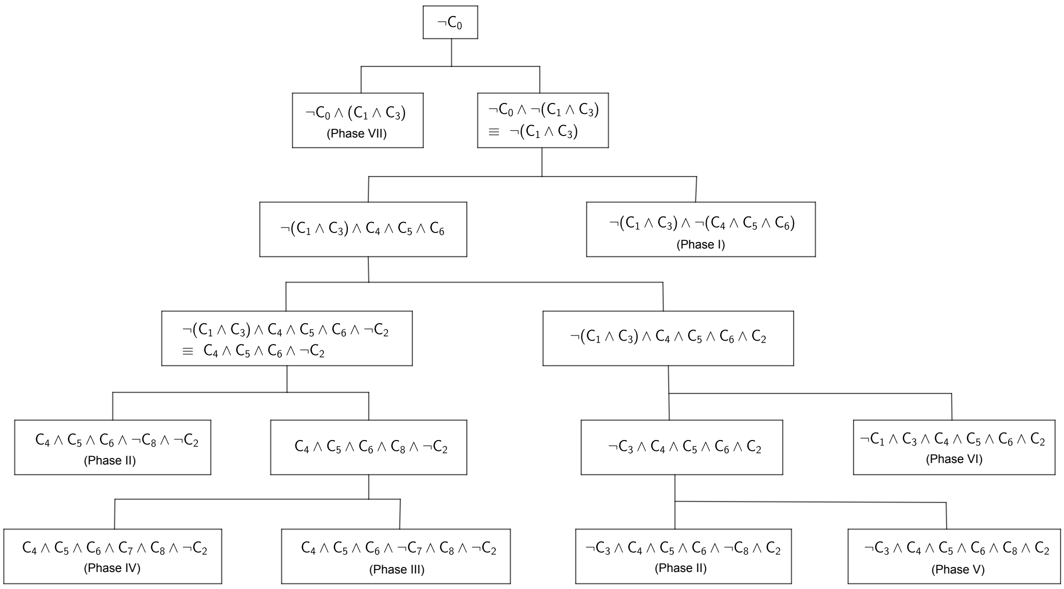

Figure 4 For any configuration with $C_0 =$ false belongs to one of the seven phases.

▶ **Theorem 3.** *The proposed algorithm can form any pattern consisting of k points by a set of k oblivious asynchronous robots if the initial configuration formed by the robots is an asymmetric configuration and has no multiplicity point.*

Recall the Definition 1 of the space complexity of an algorithm executed by a set of robots on an infinite rectangular grid. In Theorem 4, we calculate the space complexity of the proposed algorithm. The move complexity is recorded in the Theorem 5. Proofs of these theorems are available in full version of the paper [14].

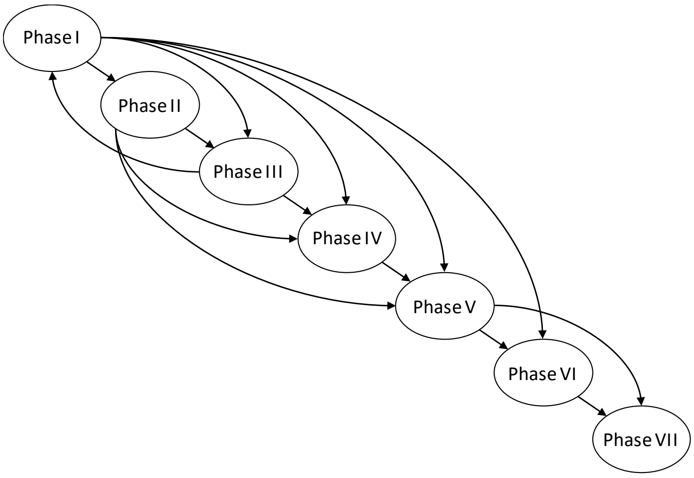

Figure 5 Phase transition digraph.

▶ **Theorem 4.** *Let $\mathcal{D} = \max\{m, n, m', n'\}$ where $m \times n$ $(m \geq n)$ is the dimension of the SER of the initial configuration and $m' \times n'$ $(m' \geq n')$ is the dimension of the SER of the target configuration. Then the space complexity of the proposed algorithm is at most $\mathcal{D} + 4$. More precisely, if $M = \max\{m, m'\}$ and $N = \max\{n, n'\}$, then the proposed algorithm takes the space enclosed by a rectangle of dimension $(M + 4) \times (N + 1)$.*

▶ **Theorem 5.** *The proposed algorithm requires each robot to make $O(\mathcal{D})$ movements, hence the move-complexity of the proposed algorithm is $O(k\mathcal{D})$.*

5 Conclusion

This work first provides an algorithm that solves the APF problem in an infinite line by a robot swarm. Then adopting the method, it provides another algorithm that solves the APF in an infinite rectangular grid by a robot swarm. The robots are autonomous, anonymous, identical, and homogeneous. The robot model used here is the classical $\mathcal{OBLOT}$ model. The robots work under a fully asynchronous scheduler. The proposed algorithm is almost space-optimal (Theorem 4) and asymptotically move-optimal (Theorem 5).

A few limitations of this work are the following. Here we assume that the initial configuration is asymmetric. Finding complete characterization of the initial configurations from which APF can be solved deterministically is an interesting future direction. Next, the version of the APF problem under consideration does not permit multiple points in the target configuration. More precisely, the number of target positions in the target pattern is equal to the number of robots within the system. Solving a more generalized version of the problem that allows target patterns with target positions less than the total number of robots, is a possible future direction. Next, the proposed algorithm is almost space optimal, so finding out the exact lower bound when starting from an asymmetric initial configuration is an interesting direction. Also, this does not consider time-optimality, so considering all the three parameters space, move and time at the same time can be an interesting future work.

References

1 Kaustav Bose, Ranendu Adhikary, Manash Kumar Kundu, and Buddhadeb Sau. Arbitrary pattern formation on infinite grid by asynchronous oblivious robots. *Theoretical Computer Science*, 815:213–227, 2020. doi:10.1016/j.tcs.2020.02.016.

2 Kaustav Bose, Archak Das, and Buddhadeb Sau. Pattern formation by robots with inaccurate movements. In Quentin Bramas, Vincent Gramoli, and Alessia Milani, editors, *25th International Conference on Principles of Distributed Systems, OPODIS 2021, December 13-15, 2021, Strasbourg, France*, volume 217 of *LIPIcs*, pages 10:1–10:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.OPODIS.2021.10.

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

4 Quentin Bramas and Sébastien Tixeuil. Arbitrary pattern formation with four robots. In Taisuke Izumi and Petr Kuznetsov, editors, *Stabilization, Safety, and Security of Distributed Systems - 20th International Symposium, SSS 2018, Tokyo, Japan, November 4-7, 2018, Proceedings*, volume 11201 of *Lecture Notes in Computer Science*, pages 333–348. Springer, 2018. doi:10.1007/978-3-030-03232-6_22.

5 Serafino Cicerone, Alessia Di Fonso, Gabriele Di Stefano, and Alfredo Navarra. Arbitrary pattern formation on infinite regular tessellation graphs. *Theor. Comput. Sci.*, 942:1–20, 2023. doi:10.1016/j.tcs.2022.11.021.

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

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

8 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.

9 Satakshi Ghosh, Pritam Goswami, Avisek Sharma, and Buddhadeb Sau. Move optimal and time optimal arbitrary pattern formations by asynchronous robots on infinite grid. *International Journal of Parallel, Emergent and Distributed Systems*, 0(0):1–23, 2022. doi:10.1080/17445760.2022.2124411.

10 Rory Hector, Gokarna Sharma, Ramachandran Vaidyanathan, and Jerry L. Trahan. Optimal arbitrary pattern formation on a grid by asynchronous autonomous robots. In *2022 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2022, Lyon, France, May 30 - June 3, 2022*, pages 1151–1161. IEEE, 2022. doi:10.1109/IPDPS53621.2022.00115.

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

12 Manash Kumar Kundu, Pritam Goswami, Satakshi Ghosh, and Buddhadeb Sau. Arbitrary pattern formation by opaque fat robots on infinite grid. *International Journal of Parallel, Emergent and Distributed Systems*, 37(5):542–570, 2022. doi:10.1080/17445760.2022.2088750.

13 Brati Mondal, Pritam Goswami, Avisek Sharma, and Buddhadeb Sau. Arbitrary pattern formation on a continuous circle by oblivious robot swarm. In *Proceedings of the 25th International Conference on Distributed Computing and Networking, ICDCN 2024, Chennai, India, January 4-7, 2024*, pages 94–103. ACM, 2024. doi:10.1145/3631461.3631545.

14 Avisek Sharma, Satakshi Ghosh, Pritam Goswami, and Buddhadeb Sau. Space and move-optimal arbitrary pattern formation on infinite rectangular grid by oblivious robot swarm. *CoRR*, abs/2309.11190, 2023. doi:10.48550/arXiv.2309.11190.

15 Avisek Sharma, Satakshi Ghosh, Pritam Goswami, and Buddhadeb Sau. Space and move-optimal arbitrary pattern formation on a rectangular grid by robot swarms. In *Proceedings of* *the 25th International Conference on Distributed Computing and Networking, ICDCN 2024, Chennai, India, January 4-7, 2024*, pages 65–73. ACM, 2024. `doi:10.1145/3631461.3631542`.

16 Ichiro Suzuki and Masafumi Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing*, 28(4):1347–1363, 1999. `doi:10.1137/S009753979628292X`.

17 Masafumi Yamashita and Ichiro Suzuki. Characterizing geometric patterns formable by oblivious anonymous mobile robots. *Theoretical Computer Science*, 411(26):2433–2453, 2010. `doi:10.1016/j.tcs.2010.01.037`.