Move optimal and time optimal arbitrary pattern formations by asynchronous robots on infinite grid

Satakshi Ghosh, Pritam Goswami, Avisek Sharma & Buddhadeb Sau

To cite this article: Satakshi Ghosh, Pritam Goswami, Avisek Sharma & Buddhadeb Sau (2023) Move optimal and time optimal arbitrary pattern formations by asynchronous robots on infinite grid, International Journal of Parallel, Emergent and Distributed Systems, 38:1, 35-57, DOI: 10.1080/17445760.2022.2124411

To link to this article: https://doi.org/10.1080/17445760.2022.2124411

Published online: 20 Sep 2022.

Submit your article to this journal

Article views: 31

View related articles

View Crossmark data

INTERNATIONAL JOURNAL OF PARALLEL, EMERGENT AND DISTRIBUTED SYSTEMS
2023, VOL. 38, NO. 1, 35–57
https://doi.org/10.1080/17445760.2022.2124411

Move optimal and time optimal arbitrary pattern formations by asynchronous robots on infinite grid

Satakshi Ghosh, Pritam Goswami, Avisek Sharma and Buddhadeb Sau

Department of Mathematics, Jadavpur University, Kolkata, India

ABSTRACT
The Arbitrary Pattern Formation (Apf) is widely studied in distributed computing for swarm robots. This paper deals with the Apf problem in an infinite grid under an asynchronous scheduler. In [Bose K, Adhikary R, Kundu MK, et al. Arbitrary pattern formation on infinite grid by asynchronous oblivious robots. Theor Comput Sci. 2020;815:213–227], the authors proposed an algorithm for Apf problem in $\mathcal{OBLOT}$ model under an asynchronous scheduler, but the proposed algorithm was neither time optimal nor move optimal. This work provides two algorithms that solve Apf problem in an asynchronous scheduler. The first algorithm is move optimal considering $\mathcal{OBLOT}$ model and the second algorithm is move and time optimal considering the $\mathcal{LUMI}$ model, where each robot has one light having three distinct colours.

ARTICLE HISTORY
Received 27 May 2022
Accepted 1 September 2022

KEYWORDS
Distributed computing; autonomous robots; arbitrary pattern formation; robots with lights; asynchronous; look–compute–move cycle; grid

1. Introduction

In distributed systems, the robot swarm coordination problems are being studied for the last two decades. The main aim of the distributed system is to use a swarm of inexpensive robots to do any particular work rather than using a very expensive robot. Arbitrary Pattern Formation (Apf) is a fundamental coordination problem for autonomous robot swarms. The goal of this problem is to design a distributed algorithm that guides the robots to form any specific but arbitrary pattern given to the robots as an input. In this context, the main research difficulties are which patterns can be formed and how they can be formed. In the Euclidean plane, robots can move in any direction for a very small amount of distance, but it is not always possible for robots with weak capabilities to move accurately. So it is interesting to consider this type of problem in grid terrain, where robot movement is restricted in between grid points. In practical applications, the interest has shifted to using a large number of simple robots which are easy to design and deploy and have minimal capabilities to make the system cost-effective.

In the theoretical framework, depending on the capabilities there are generally four types of robot models. These models are $\mathcal{OBLOT}$, $\mathcal{FSTA}$, $\mathcal{FCOM}$ and $\mathcal{LUMI}$. In each of these models robots are assumed to be autonomous (i.e the robots do not have any central control), identical (i.e the robots are physically indistinguishable), and homogeneous (i.e each robot runs on the same algorithm). In the $\mathcal{OBLOT}$ model, the robots are silent (i.e there is no means of communication between the robots) and oblivious (i.e the robots do not have any persistent memory to remember their previous state), in $\mathcal{FSTA}$ model, the robots are silent but not oblivious, in $\mathcal{FCOM}$ model, the robots are oblivious but

CONTACT Satakshi Ghosh satakshighosh.math.rs@jadavpuruniversity.in; Pritam Goswami pritamgoswami.math.rs@jadavpuruniversity.in; Avisek Sharma aviseks.math.rs@jadavpuruniversity.in; Buddhadeb Sau buddhadeb.sau@jadavpuruniversity.in

not silent and in $\mathcal{LUMI}$ model, robots are neither silent nor oblivious. The robots do not have access to any global coordinate system. The robots after getting activated operate in a Look–Compute–Move (Lcm) cycle. In Look phase, a robot takes input from its surroundings and then with that input runs the algorithm in Compute phase to get a destination point as an output. The robot then goes to that destination point by moving in the Move phase. The activation of the robots is controlled by a scheduler. There are mainly three types of schedulers considered in the literature. In a fully synchronous scheduler, time is divided into global rounds. In a fully synchronous (FSync) scheduler, each robot is activated in all rounds and executes Lcm cycle simultaneously. In a semi-synchronous scheduler (SSync), all robots may not get activated in each round but the robots that are activated in the same round execute the Lcm cycle simultaneously. Lastly in the asynchronous scheduler (ASync), there is no common notion of time, a robot can be activated at any time. There is no concept of rounds. So there is no assumption regarding synchronisation.

Leader election is an important task for the pattern formation problem, where a unique robot is elected as a leader. In [1], an arbitrary pattern formation algorithm is given in an infinite grid under asynchronous scheduler considering $\mathcal{OBLOT}$ model. This paper aims to solve the arbitrary pattern formation problem in an infinite grid by a swarm of robots with the optimal number of moves and within optimal time under a fully asynchronous scheduler. First, this work proposes an algorithm that solves the Apf problem in an infinite grid with the optimal number of robot moves in $\mathcal{OBLOT}$ model. Furthermore, we propose another algorithm for solving Apf problem on an infinite grid considering in $\mathcal{LUMI}$ model for robots which is both move and time optimal. Our work shows that the algorithm proposed in [1] is not move optimal asymptotically. And this work also shows that Apf problem can be solved faster than the algorithm proposed in [1] by introducing communicable memory.

2. Related works and our contribution

2.1. Related work

The Arbitrary pattern formation problem has been studied in various settings. In the Euclidean plane, this problem was first studied by Suzuki and Yamashita [2]. They provided a complete characterisation of the class of pattern formable in FSync and SSync for anonymous robots with unbounded memory. Later in [3], they characterised the families of patterns formable by oblivious robots in FSync and SSync. Then Flochhini [4] studied the cases of solvability of this problem under various assumptions. They showed that without a common coordinate system Apf problem is not solvable, but when there are both axes-agreement, the problem can be solved. Furthermore, with one axis agreement, any odd number of robots can form an arbitrary pattern, but an even number of robots cannot in the worst case. In [5], the authors have established a relationship between Leader Election and Arbitrary Pattern Formation of robots under an asynchronous scheduler. Later they also showed that the arbitrary pattern formation is possible to solve when $n \geq 4$ with chirality (resp. $n \geq 5$ without chirality) if and only if leader election is solvable. In [6], the authors consider the arbitrary pattern formation problem with four robots in the asynchronous model with or without chirality. In [7], the authors have solved the Apf problem with inaccurate movement where the formed pattern is very similar to the target pattern, but not exactly the same. A randomised pattern formation problem was studied in [8]. In [9], the authors have shown some configurations where embedded pattern formation is solvable without chirality and some configuration where embedded pattern formation are deterministically unsolvable.

For grid network, the arbitrary pattern formation was first studied by [1] in $\mathcal{OBLOT}$ model with full visibility. Later in [10], the authors studied the Apf on a regular tessellation graph.

Another interesting direction of solving this problem is when visibility is limited. Yamauchi in their paper [11] first showed that oblivious robots under FSync model with limited visibility can not solve Apf. Therefore, they considered non-oblivious robots with unlimited memory. For these robots, they

Table 1. Comparison table.

Paper	Robot model	Visibility model	No. of colours	Move complexity	Time complexity
[1]	$\mathcal{OBLOT}$	Unobstructed	–	$O(\mathcal{D}'^3)$	$\Omega(\mathcal{D}'^2)$[Res.1]
[18]	$\mathcal{LUMI}$	Obstructed	9	–	–
APFOptMove	$\mathcal{OBLOT}$	Unobstructed	–	$O(\mathcal{D}'^2)$[Th.5.12] (**optimal**)	$O(\mathcal{D}'^2)$[Th.5.13]
FastAPF	$\mathcal{LUMI}$	Unobstructed	3	$O(\mathcal{D}'^2)$[Th.6.5] (**optimal**)	$O(\mathcal{D}')$[Th.6.4] (**optimal**)

presented algorithms that work in FSync with non-rigid movements and in SSync with rigid movements. After that in [12], the authors have solved this problem in an infinite grid under 2 hop visibility. The problem was studied in a synchronous setting for robots with constant-size memory, where the robots agree on a common coordinate system.

A special case of a formation problem is mutual visibility problem [13] and a gathering problem [14]. In mutual visibility, robots are opaque so the main task is to form a configuration where no three robots are co-linear. Recently, Apf problem was solved in a euclidean plane in [15] with opaque robots and in [16] with fat robots considering the luminous model. In an infinite grid, the arbitrary pattern formation problem was studied in [17] with opaque robots and in [18] with fat robots. The work in [19] solves Apf in the obstructed visibility model without any agreement in the coordinate system, where they showed that the run time to solve Apf is bounded above by the time required to elect a leader. Another special case of formation problem, Uniform circle formation is investigated in [20, 21]. Das et al. solved the problem of forming a sequence of patterns in a given order [22]. Further, they extended the sequence of pattern formation problems for luminous robots in [23]. There are many works [24, 25] where the pattern can be formed by robots with multiplicities. Pattern formation in the presence of faulty robots is an important topic of research. In [26], they studied the formation of patterns allowing only crash fault robots. In the next subsection, we shall discuss the scope of our work comparing it with relevant works.

2.2. Our contribution

In this work, our goal is to solve Apf problem under the full visibility model optimally in terms of energy and time. Energy is measured by the number of moves made by the robots and time can be measured by the total number of epochs required to solve the problem. First, we define the input size of the problem. Let k be the number of robots in the input configuration and $\mathcal{D}$ be the size of the smallest square that can contain both the initial configuration and the target pattern/configuration. Let $\mathcal{D}' = \max\{\mathcal{D}, k\}$.

In [1], the authors provided an algorithm solving Apf problem in $\mathcal{OBLOT}$ model. But the solution is not optimal in terms of energy because the algorithm proposed in [1] needs all total $O(k\mathcal{D}^2) = O(\mathcal{D}'^3)$ moves. Also in [1], the authors showed that any algorithm solving Apf requires $\Omega(\mathcal{D}'^2)$ total moves. In this work, we propose an algorithm that solves the Apf problem in $\mathcal{OBLOT}$ model which requires $O(\mathcal{D}'^2)$ total robot moves, which is asymptotically optimal. Then the algorithm proposed in [1] requires $\Omega(\mathcal{D}'^2)$ epoch time in the worst case (Result 1). We show in this work that any algorithm solving Apf requires $\Omega(\mathcal{D}')$ epoch time. In this work, we propose another algorithm that solves Apf in $\mathcal{LUMI}$ model. This algorithm requires $O(\mathcal{D}')$ epoch time which is faster than the algorithm proposed in [1]. This also establishes that our algorithm is asymptotically time optimal in $\mathcal{LUMI}$ model. Although we cannot tell that the algorithm proposed in [1] is not time optimal because it is done in $\mathcal{OBLOT}$ which is a weaker model than $\mathcal{LUMI}$. Further, we also show that the second algorithm is move optimal as well.

Although all the above-mentioned works in this subsection are done under the full visibility model, in [18], the authors solve Apf under obstructed view considering fat robots. However, authors did not do any complexity analysis. Authors in [18] used 9 colours to solve the problem where our algorithm uses only 3 colours. The above discussion is briefly presented in Table 1.

2.3. Organization of the paper

Section 3 describes the relevant robotics model of this work. Section 4 states and describes APF problem. Next Section 5 and Section 6 provide two algorithms and their algorithm descriptions and correctness proofs. Section 5 provides the move optimal algorithm in $\mathcal{OBLOT}$ model and Section 6 provides the time and move optimal algorithm in $\mathcal{LUMI}$ model. Finally, Section 7 concludes this work.

3. Robot model

3.1. Classical oblivious robots

In the first problem, the $\mathcal{OBLOT}$ model is considered for the robots. In this model, robots are anonymous, identical, and oblivious, i.e. they have no memory of their past rounds. They can not communicate with each other. All robots are initially in distinct positions on the grid. The robots can see the entire grid and all other robots' positions which means they have global visibility. This implies the robots are transparent and hence the visibility of a robot can not be obstructed by other robots. Robots have no access to any common global coordinate system. They have no common notion of chirality or direction. A robot has its local view and it can calculate the positions of other robots with respect to its local coordinate system with the origin at its own position. There is no agreement on the grid about which line is x- or y-axis and also about the positive or negative direction of the axes. As the robots can see the entire grid, they will set the axes of their local coordinate systems along the grid lines.

3.2. Robots with lights

In the second problem the $\mathcal{LUMI}$ model has been considered. In this model, the robots are anonymous and identical and they have constant memory (finite number of lights). Each robot has a light that can assume one colour at a time from a constant number of different colours. All the other assumptions are the same as the classical oblivious robots model.

3.3. Look–Compute–Move cycles

An active robot operates according to the Look–Compute–Move cycle. In each cycle, a robot takes a snapshot of the positions of the other robots according to its own local coordinate system (LOOK); based on this snapshot, it executes a deterministic algorithm to determine whether to stay put or to move to an adjacent grid point (COMPUTE); and based on the algorithm the robot either remain stationary or makes a move to an adjacent grid point (MOVE). When the robots are oblivious they have no memory of past configurations and previous actions. After completing each LOOK–COMPUTE–MOVE cycle, the contents in each robot's local memory are deleted. When each robot is equipped with an externally visible light, which can assume a $O(1)$ number of predefined colours, the robots communicate with each other using these lights. The lights are not deleted at the end of a cycle. In second algorithm, we use one light which takes OFF, HEAD and LINE colours.

3.4. Scheduler

We assume that robots are controlled by a fully asynchronous adversarial scheduler (ASYNC). The robots are activated independently and each robot executes its cycles independently. This implies the amount of time spent in LOOK, COMPUTE, MOVE and inactive states is finite but unbounded, unpredictable and not same for different robots. The robots have no common notion of time.

3.5. Movement

In discrete domains, the movements of robots are assumed to be instantaneous. This implies that the robots are always seen on grid points, not on edges. However, in our work, we do not need this assumption. In the first proposed move optimal algorithm, we assume the movements are to be instantaneous for simplicity. However, this algorithm also works without this assumption. In that case, the robots are asked to wait and do nothing if they see a robot on a grid edge. In the second proposed time optimal algorithm no such assumption is required. That is, if a robot sees any robot on an edge, it still does its job as directed by the algorithm. The movement of the robots is restricted from one grid point to one of its four neighbouring grid points.

3.6. Measuring run time

Generally, time is measured in rounds in fully synchronous settings. But as robots can stay inactive for an indeterminate time in semi-synchronous and asynchronous models, epochs are considered instead of rounds. During an epoch, it is assumed that all robots are activated at least once. Here in the second algorithm, we calculate the run time with respect to epochs.

Next, we describe the Arbitrary Pattern Formation problem in an infinite grid in the next section.

4. Problem description

4.1. Problem statement

Suppose a swarm of robots is placed in an infinite 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 algorithm following which the robots autonomously can form any arbitrary but specific pattern, which is provided to the robots as an input, without scaling it or colliding with another robot.

Let's discuss some facts about the problem. The input target pattern can be any arbitrary pattern. The required distributed algorithm has to be independent of the input target pattern, that is, the same algorithm should work for any target input pattern. The whole target pattern is given to all the robots but no robot knows its target position beforehand. The target pattern can be provided to the robots with respect to some coordinate system but the robots initially do not agree on any coordinate system. Robots are not allowed to scale the pattern configuration but are allowed to transform or rotate it. From the motivation from [27], we assume that the initial configuration of robots is asymmetric. But the target pattern can be any pattern (can possibly be symmetric). The algorithm solving APF must take care that no two robots collide.

In each of the next two sections, we provide an algorithm that solves the Arbitrary Pattern Formation problem. The first one works in $\mathcal{OBLOT}$ model and the latter one works in $\mathcal{LUMI}$ model.

5. Optimal move APF algorithm (APFOptMove)

5.1. Agreement on global coordinate system

Here we consider an infinite grid G as a cartesian product $P \times P$, where P is an infinite (from both sides) path graph. The infinite grid G is embedded in the Cartesian Plane R^2. We know that the solvability of Arbitrary pattern formation depends on the symmetries of the initial configuration of the robots. Here we are assuming that the initial configuration is asymmetric. The robots do not have an access to any global coordinate system even though each robot can form a local coordinate system aligning the axes along the grid lines. To form the target pattern the robots need to reach an agreement on a global coordinate system. This subsection provides details of the procedure that allows the robots to reach an agreement on a global coordinate system.

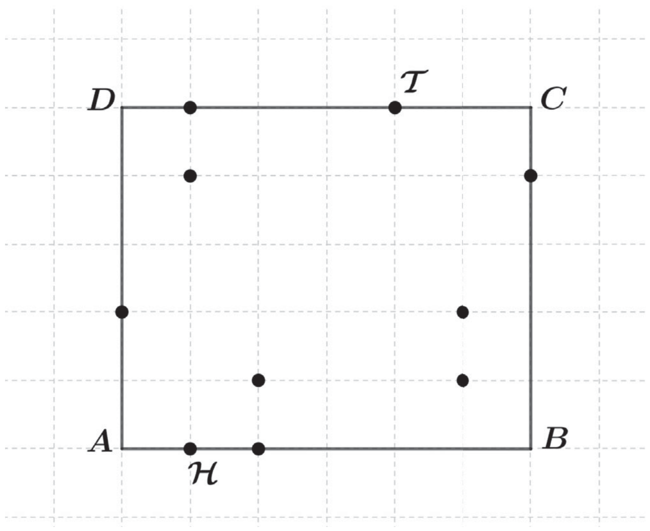

Figure 1. In this configuration, AB is the largest string. Here the binary string of AB is 011000000100 1010000100000000001000010100100. $\mathcal{H}$ and $\mathcal{T}$ are head and tail, respectively.

For a given configuration ($\mathcal{C}$) formed by the robots, let the smallest enclosing rectangle, denoted by $s.rect(\mathcal{C})$, is the smallest grid aligned rectangle which contains all the robots. Suppose the $s.rect$ of the initial configuration $\mathcal{C}_I$ is a rectangle $\mathcal{R} = ABCD$ of size $m \times n$, such that $m > n > 1$. Let $|AB| = n$. Then consider the binary string $\{s_i\}$ associated with a corner A, λ_{AB} as follows. Scan the grid from A along the side AB to B and sequentially all grid lines of $s.rect(\mathcal{C}_{\mathcal{I}})$ parallel to AB in the same direction. And $s_i = 0$, if the position is unoccupied and $s_i = 1$ otherwise (Figure 1). Similarly construct the other binary strings λ_{BA}, λ_{CD} and λ_{DC}. Since the initial configuration is asymmetric we can find a unique lexicographically largest string. If λ_{AB} is the lexicographically largest string, then A is called as the leading corner of $\mathcal{R}$.

Next, suppose $\mathcal{R}$ is an $m \times m$ square, then consider the eight binary strings $\lambda_{AB}, \lambda_{BA}, \lambda_{CD}, \lambda_{DC}, \lambda_{BC}, \lambda_{CB}, \lambda_{AD}, \lambda_{DA}$. Again since the initial configuration is asymmetric, we can find a unique lexicographically largest string among them. Hence we can find a leading corner here as well.

Next, let $\mathcal{C}_I$ be a line AB, we will have two strings λ_{AB} and λ_{BA}. Since $\mathcal{C}_I$ is asymmetric then λ_{AB} and λ_{BA} must be distinct. If λ_{AB} is lexicographically larger than λ_{BA}, then we choose A as the leading corner.

Now for either case, if λ_{AB} is the lexicographically largest string then the leading corner A is considered as the origin, and the x-axis is taken along the AB line. If $\mathcal{C}_I$ is not a line then the y-axis is taken along the AD line. If λ_{AB} is a line then the y-coordinate of all the positions of robots is going to be zero and in this case, the y-axis will be determined later.

For any given asymmetric configuration $\mathcal{C}$ if λ_{AB} is the largest associated binary string to $\mathcal{C}$ then the robot causing the first non-zero entry in λ_{AB} is called *head* let $\mathcal{H}$ and the robot causing last non-zero entry in λ_{AB} is called as *tail* let $\mathcal{T}$. We denote the ith robot of the λ_{AB} string as r_{i-1}. A robot other than head and tail is called ***Inner robot***. Further we denote $\mathcal{C}' = \mathcal{C} \setminus \{tail\}$ and $\mathcal{C}'' = \mathcal{C} \setminus \{tail, head\}$ and $\mathcal{C}''' = \mathcal{C} \setminus \{Head\}$,

Let $\mathcal{C}_T$ be the target configuration and $s.rect(\mathcal{C}_T) = \mathcal{R}_T$. Let $\mathcal{R}_T$ is a rectangle of size $M \times N$ with $M \geq N$. We can calculate the binary strings associated with corners in the same manner as the previous one. $\mathcal{C}_T$ is expressed in the coordinate system with respect to the origin, where the origin will be the leading corner. Let the $A'B'C'D'$ be the smallest rectangle enclosing the target pattern with $A'B' \leq B'C'$. Let $\lambda_{A'B'}$ be the largest (may not be unique) among all other strings. Then the target pattern is to be formed such that $A = A'$, $A'B'$ direction is along the positive x-axis and $A'D'$ direction is along the positive y -axis. If target pattern has symmetry then we have to choose any one among the larger string and fixed the coordinate system. So as previously said $head_{target}$ will be the first one and $tail_{target}$ will be

Table 2. Boolean functions.

S_0	$\mathcal{C} = \mathcal{C}_T$
S_1	$\mathcal{C}' = \mathcal{C}'_T$
S_2	x-coordinate of the tail in $\mathcal{C} = x$ Coordinate of t_{target} in $\mathcal{C}_T$
S_3	$m \geq \max\{N, n\} + 2$
S_4	$m \geq 2.\max\{M, V\}$ where v is the length of the vertical side of the smallest enclosing rectangle of $\mathcal{C}'$
S_5	The head in $\mathcal{C}$ is at the origin.
S_6	$n \geq \max\{N+1, H+1, k\}$ where H is the length of the horizontal side of the smallest enclosing rectangle of $\mathcal{C}'$
S_7	$\mathcal{C}'' = \mathcal{C}''_T$
S_8	$\mathcal{C}'$ has a non-trivial reflectional symmetry with respect to a vertical line.
S_9	$\mathcal{C}''' = \mathcal{C}'''_T$.
S_{10}	Line formation on x-axis without tail and one inner robot.

the last one in the *s.rect* of $\mathcal{C}_\mathcal{T}$. Also we define $\mathcal{C}_\mathcal{T}' = \mathcal{C}_\mathcal{T} \setminus \{tail_{target}\}, \mathcal{C}_\mathcal{T}'' = \mathcal{C}_\mathcal{T} \setminus \{head_{target}, tail_{target}\}$, $\mathcal{C}_\mathcal{T}''' = \mathcal{C}_\mathcal{T} \setminus \{head_{target}\}$. We denote the $head_{target}$ position as t_0 and $tail_{target}$ position as t_{k-1}. Let H_i be the horizontal line having the height i from x-axis. Let for each i there are $p(i)$ target positions on H_i. We denote the target positions of H_0 as $t_0, t_1 \ldots\ldots t_{p(0)-1}$ from left to right. For H_1 we denote the target positions as $t_{p(0)}$ to $t_{p(0)+p(1)-1}$ from right to left. For H_2 we denote the target positions as $t_{p(0)+p(1)}$ to $t_{p(0)+p(1)+p(2)-1}$ from right to left. Similarly, we can denote all other target positions on $H_i, i > 0$ except $tail_{target}$. Next, we give some relevant definitions.

Definition 5.1 (Inner robot): A robot that is not head or tail in a configuration is called an inner robot.

Definition 5.2 (Compact Line): A line is called compact if there is no empty grid point between two robots.

5.2. A brief outline of APFOptMove algorithm

In this algorithm, our goal is to make Apf with move optimal. For this, robots initially form a line and then form the arbitrary pattern that is given as input. We can divide our algorithm into eight phases. Since the initial configuration is asymmetric, the robot can agree on a global coordinate system. So robots can recognise where the pattern can be formed. Here robots have to maintain the coordinate system during their movements. When a robot wakes up, it can be in any phase among the eight phases. So, as the robots are oblivious they can check by their condition in which phase it is currently in. The conditions are expressed in the Boolean function listed in Table 2. As the movement of a robot is restricted in the discrete domain, here we have to maintain the movement without collision throughout the algorithm. As maintaining the asymmetry is another main challenge of our algorithm, in the first three phases the head will be put at the origin and the tail will expand to the smallest enclosing rectangle for another robot can move but the head and tail remain unchanged and asymmetry is also maintained. In phase four robots form a line on the x-axis without the tail and one inner robot of $\mathcal{C}'$. In phase five, all robots move to the fixed target position sequentially without a head and tail. In the last three phases, the head and tail will reach their fixed target positions. During these phases movement of robots are difficult as here the coordinate system may change. But here, we showed that asymmetry and global coordinate will be maintained in all phases. In this way, arbitrary pattern formation can be done.

5.3. Detailed description of the eight phases

5.3.1. Phase 1

A robot is in phase 1 then tail will move upwards and all other robots will remain static. When in phase 1 $\neg\{S_1 \wedge S_2\} \wedge \neg\{S_3 \wedge S_4\}$ is true.

Theorem 5.3: *If we have an asymmetric configuration* $\mathcal{C}$ *at some time t*

- *After one move upward, the new configuration is still asymmetric and the coordinate system remains unchanged.*
- *after one move by the tail upwards $\neg\{S_1 \wedge S_2\} = true$*

Proof: Let *ABCD* be the initial smallest enclosing rectangle at any time t, and let the binary string associated with *AB* be the largest with $|AB| = n$ and $|AD| = m, m \geq n$. So initially, the tail is on the side *CD*. But after the tail moves upward, the smallest enclosing rectangle changes. Let the new rectangle is $ABC'D'$. Here tail $\mathcal{T}$ is now on the side $C'D'$. Now let $\mathcal{T}$ be the only robot initially on side *CD*, then it is obvious that $\lambda_{AB}^{new} > \lambda_{BA}^{new}$. But if there are multiple robots on *CD* then let *t* be the *p*th and *q*th term of λ_{AB}^{old} and λ_{BA}^{old}. Then,

Case-1: When $p = q$ then *t* is the middle robot of CD, here *p*th term is the last 1 occurs in λ_{AB}^{old} so if we calculate the binary string of first $(p - 1)$ term of *AB* and *BA* in the new *s.rect* then also we get $\lambda_{AB}^{new} > \lambda_{BA}^{new}$.

Case-2: When $p > q$ then if we calculate the binary strings of λ_{AB}^{old} and λ_{BA}^{old} then $\mathcal{T}$ will appear earlier in *BA*, than *AB*. Now if we calculate the binary strings of the first *q* term of AB and BA then $\lambda_{BA}^{old}|_q > \lambda_{BA}^{new}|_q$. Also $\lambda_{AB}^{old}|_q > \lambda_{AB}^{new}|_q$. But $\lambda_{AB}^{old} > \lambda_{BA}^{old}$. So we have $\lambda_{AB}^{old}|_q > \lambda_{BA}^{old}|_q$. Finally, we can say $\lambda_{AB}^{new}|_q > \lambda_{BA}^{new}|_q$, so $\lambda_{AB}^{new} > \lambda_{BA}^{new}$.

Case-3: When $p < q$ in that case when $\mathcal{T}$ robot moves upward then in the new *s.rect* we can calculate that in this case also $\lambda_{AB}^{new} > \lambda_{BA}^{new}$.

So in all the cases $\lambda_{AB}^{new} > \lambda_{BA}^{new}$. Now we show that the binary string of *AB* is larger than $C'D'$. As $\mathcal{T}$ is the tail so we know that the binary string of *AB* is larger than *CD*, but when the tail robot moves one step upward in that case as there is no robot other than the tail in $C'D'$ so if we calculate binary string it will be $\lambda_{AB}^{new} > \lambda_{C'D'}^{new}$.

In this case, $ABC'D'$ is a non-square grid, so four binary strings to consider here. By calculating we can say that *AB* is the largest binary string in this new smallest enclosing rectangle. So the new configuration is still asymmetric. So the coordinate system is unchanged. As the tail moves upward so the *x*-coordinate remains unchanged, so $\neg\{S_1 \wedge S_2\}$ remains the same after one move by the tail robot. ■

5.3.2. Phase 2

In this phase head $\mathcal{H}$ will move left towards the origin. When the algorithm is in phase 2 then either $S_3 \wedge S_4 \wedge \neg S_5 \wedge \neg S_7$ or $\neg S_2 \wedge S_3 \wedge S_4 \wedge \neg S_5 \wedge S_7$ is true.

Theorem 5.4: *If we have an asymmetric configuration $\mathcal{C}$ at time t in phase 2 then*

(1) *after one move by the head robot to the left, the new configuration is still asymmetric and the coordination system is not changed.*
(2) *after a finite number of moves by the head to the left S_5 is true and phase 2 terminates.*

Proof: As *ABCD* is the smallest enclosing rectangle of all robots, let λ_{AB} be the lexicographically largest string, after one move of the head to the left, now also λ_{AB} is the largest string. Let at the *i*th term the first 1 occurs in λ_{AB}, then in the other strings all the $(i - 1)$ terms are 0. But when the head moves one distance to the left then the new string form is now the largest. So the new configuration is asymmetric and the global coordinate will not change. So the statement (1) is true, similarly by the finite number of moves head move to the origin then S_5 true. ■

5.3.3. Phase 3

The goal of this phase is to make S_6 true. In this phase, the robots will check either S_8 is true or false. When the algorithm is in phase 3 then $S_3 \wedge S_4 \wedge S_5 \wedge \neg S_6 \wedge \neg S_7 =$ true. When S_8 is false, then the tail will move rightwards and the rest will remain static. But when S_8 is true, the $\mathcal{C}'$ has a non-trivial reflectional symmetry with respect to a vertical line *V*.

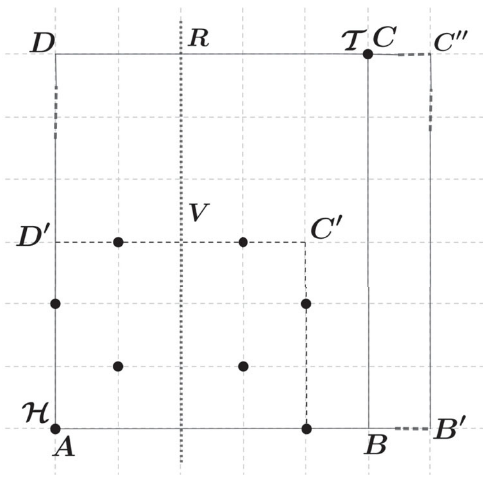

Figure 2. Case-1: $\mathcal{C}'$ has a vertical symmetry and tail will move rightwards.

Let the smallest enclosing rectangle is $\mathcal{R} = ABCD$ where $|AB| = n$ and $|AD| = m, m > n$. Let λ_{AB} be the lexicographically largest string. In this case, tail will move right and after finite number of moves, we have $S_3 \wedge S_4 \wedge S_5 \wedge S_6 \wedge \neg S_7 =$ true.

Theorem 5.5: *If we have an asymmetric configuration $\mathcal{C}$ at some time t then*

(1) *after one move rightward the new configuration is still asymmetric and the global coordinate system remains unchanged.*
(2) *after one move $S_4 \wedge S_5 \wedge \neg S_7 =$ true.*
(3) *after finite number of moves by the tail to the rightwards $S_3 \wedge S_4 \wedge S_5 \wedge S_6 \wedge \neg S_7 =$ true.*

Proof: Let the smallest enclosing circle at time t is $ABCD$, where λ_{AB} is the largest string. After one move by the tail rightward there may arise two cases.

Case-1: Suppose now tail robot is at C, then by one move of tail the new *s.rect* is $APQD$, where $|AP| = (n+1)$. Now it is easy to check that as $m \geq n+2$ so $m > n+1$. So we get that $AD > AP$. This implies that the new configuration is still not square, so we have to consider here only four binary strings, and as earlier λ_{AP} will be a larger string. So we can conclude that the configuration is still asymmetric and the coordinate system is not changed by one move of the tail. It is easy to check that S_4 and S_5 are true here but not S_7. After the movement of the tail, S_3 may become false, so we are in phase 1 then. Then the tail moves upwards and one upwards move still has $S_3 \wedge S_4 \wedge S_5 \wedge S_6 \wedge \neg S_7$ is true.

Case-2: Let after one move by the tail, the smallest enclosing circle remain unchanged. As in this phase, the head is in origin and the tail has moved until S_4 true, in the binary string of CD or DC is smaller than AB. Also in this phase, S_8 is not true. So we must have AB larger string than BA, so finally, we get $\lambda_{AB}^{new} > \lambda_{BA}^{new}$. Note that S_8 is either true or false in phase 3 by a finite number of moves of the tail the configuration remains asymmetric.

If S_8 is true then there may happen two cases:

Case-1: there is a vertical symmetry in $\mathcal{C}'$ but if we consider $ABCD$ then the tail is rightward of vertical symmetry line V. Then tail will move rightwards and after a finite number of moves, S_6 is true (Figure 2).

Case-2: When the tail is in the left portion with respect to the vertical symmetric line V. Then tail will not move rightwards (Figure 3). It will move to the left one more step than D, then the coordinate system will be changed. B will be then the origin and x-axis $= BA$ and y-axis $= BC$. Then the case will be the same as case-1.

∎

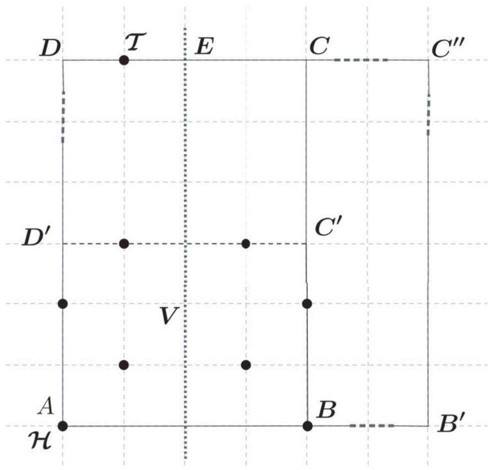

Figure 3. Case-2: $\mathcal{C}'$ has a vertical symmetry and tail will move leftwards.

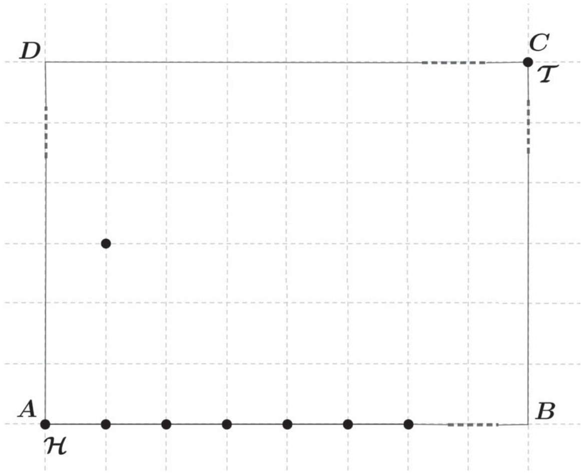

Figure 4. Line formation of robots on x-axis without tail and last inner robot.

5.3.4. Phase 4

In this phase, the configuration satisfies $S_3 \wedge S_4 \wedge S_5 \wedge S_6 \wedge \neg S_7 \wedge \neg S_{10} =$ true. Other than the tail and one inner robot, all other inner robots form a line on the x-axis. In phase four, the head is in origin, and all the robots on the x-axis first make the line compact, i.e. there is no empty grid point between two robots. After the x-axis becomes compact, when a robot r_i is on H_i and there are no robots in between H_i and the x-axis and the right part of r_i is empty in its horizontal line, then the robot moves to the x-axis. This procedure is done one by one by robots. In between this movement, no collision will occur. Finally when a robot sees that except for itself and tail all other inner robots are on the x-axis then it will not move to the x-axis (Figure 4). So all the inner robots other than the tail and one inner robot form a line on the x-axis.

Theorem 5.6: *If we have an asymmetric configuration $\mathcal{C}$ such that $S_3 \wedge S_4 \wedge S_5 \wedge S_6 \wedge \neg S_7 \neg S_{10} =$ true then in phase 4 after finite number of moves by the robots S_{10} becomes true and phase 4 terminates.*

Proof: In previous phases when the tail robot expands the smallest enclosing rectangle and the head robot moves to the origin, then in this phase robots that are on the x-axis make the line compact. In

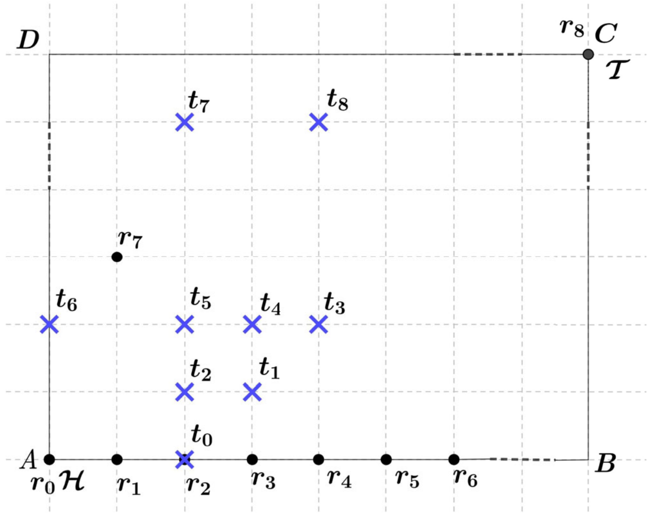

Figure 5. Target formation without head and tail.

this move, a robot will move to its left grid point if it is empty, so the robot's movement is in the left direction. So collision will not occur. A robot r_i which is on a horizontal line (let H_k) first checks that all the down lines robots are on the x-axis or not, if yes then when a robot sees that the right side of its horizontal line has also no robots, it will move to the x-axis. In this way, robots move down to the x-axis one by one. So the movement of the robot is sequential here. A robot will not move until it sees that it is the rightmost robot in its horizontal line and there is no more robot in the down horizontal lines other than the x-axis. As the *s.rect* is expanded by the tail robot in the previous phases, a robot always gets a path in the grid and moves to the x-axis. So collision will not occur in this movement. Finally without the tail and one inner robot, after a time all other robots will form a line on the x-axis. Hence S_{10} is true. ■

5.3.5. Phase 5

In this phase, inner robots will move to the fixed target one by one. When one inner robot sees that all other robots without itself and the tail are on the x-axis then it moves to t_{k-2}. We call this inner robot as *Last inner robot*. When a robot on the x-axis sees that the *Last inner robot* is at its target position then ith robot from the left on the x-axis will reach to t_i target position when it sees that from t_{i+1} to t_{k-2} positions are occupied (Figure 5).

Theorem 5.7: *If we have an asymmetric configuration initially at some time t then*

(1) *By movement of inner robots, the new configuration is still asymmetric and the coordinate system remains the same.*
(2) *After any move of the inner robots in this phase, $C'' = C''_T$ and phase 5 terminate and S_7 become true.*

Proof: As the head is in origin and the tail robot expands the *s.rect* of the initial configuration, so by the movement of inner robots, the coordinate system will not change and the configuration remains asymmetric. Here our main concern is collision. In this phase, an inner robot r_i moves to target position t_i when t_{i+1} to t_{k-2} positions are occupied by robots. As of last inner robot moves to t_{k-2} at the start of this phase. Let us denote the robots on the x-axis from left to right as $r_0, r_1, \ldots, r_{k-3}$. Then firstly r_{k-3} reaches t_{k-3}, then r_{k-4} reaches t_{k-4} and so on. Finally, r_1 moves to t_1. As the movement in this phase is sequential that is no other robot moves until one moving robot reaches its target position. Also since the target positions are ordered in such a way that every inner robot will find a unblock

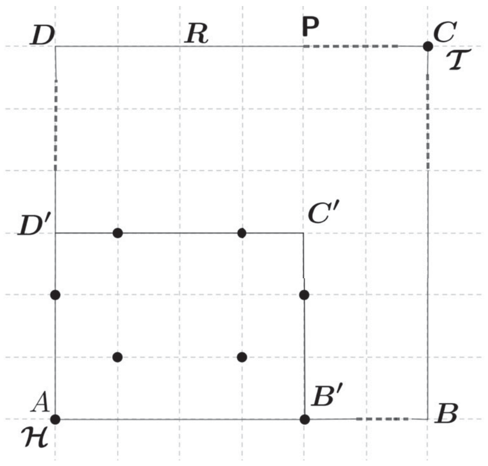

Figure 6. When S_8 is true in phase 6.

path to reach its target position. So here no collision will occur. Eventually, each inner robot reaches its target position. So finally S_7 is true. ■

5.3.6. Phase 6

The tail will move to the left until the x-coordinate matches with the $tail_{target}$. In this phase, the tail will move to make S_2 true.

Theorem 5.8: *If we have an asymmetric configuration $\mathcal{C}$ at some time t then after a finite number of moves phase 6 terminates and S_2 becomes true.*

Proof: In phase 6, if we have an asymmetric configuration, then depending on whether S_8 is true or not there are two cases. Let the smallest enclosing rectangle of $\mathcal{C}$ be $ABCD$ where A is the origin and $AB = x$-axis and $AD = y$-axis. Let without tail the smallest enclosing rectangle is $AB'C'D'$. In this phase without tail, all the robots are now on $AB'C'D'$.

Case-1: Let S_8 is not true i.e. there is no symmetry in $\mathcal{C}'$. Then the tail robot will move on the CD side, no matter where $tail_{target}$ is AB will be the lexicographically largest string. Finally, tail will move until S_2 becomes true.

Case-2: Let S_8 is true (Figure 6). Since there is symmetry in $\mathcal{C}'$, here the head robot is in A. Let P be the point of intersection between $B'C'$ and CD. In this case, when the tail robot moves left in the line CD, then it will move up to that point whose x-coordinate is the same as $tail_{target}$. In this move by the tail when it will reach a point let R which is in between P and D then a vertical symmetry will be created. The tail robot's destination can not be $[P,R]$ because then B' will be the head. So when the tail crosses R there will be a vertical symmetry. In this case also, S_1 holds. When the tail robot moves left in CD then in other cases coordinate system remains invariant and S_2 holds.

So in both cases, when phase 6 terminates then $S_2 \wedge S_3 \wedge S_4 \wedge S_5 \wedge S_7$ is true. ■

5.3.7. Phase 7

The aim of this phase is that Head will move to $head_{target}$ (Figure 7). Consider a configuration that is asymmetric and in phase 7, then with respect to the global coordinate system as fixed, let $ABCD$ is the smallest enclosing rectangle and λ_{AB} be the lexicographically largest string. Clearly head is on the side AB and the tail is on CD. If we mark all the target points on the grid then let the smallest rectangle be $AB'C'D$. Let $head_{target}$ be the final position of the head, then by finite move head will move to $head_{target}$.

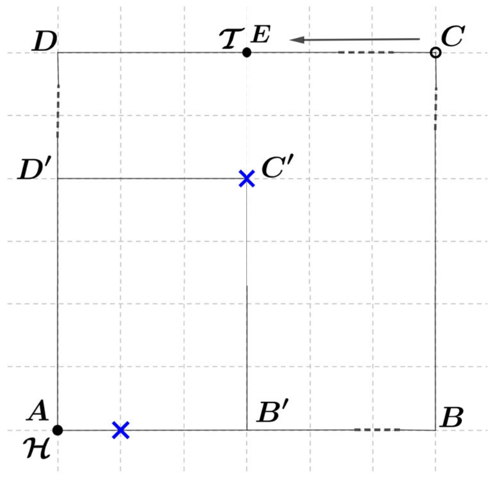

Figure 7. In phase 7, head moves to it's fixed target and in phase 8, tail moves to its target point.

Theorem 5.9: *Let $\mathcal{C}$ be the asymmetric configuration at time t in phase 7 then by a finite number of moves by the head to the right phase 7 terminate when $\neg S_0 \wedge S_1 \wedge S_2 \wedge S_9 = true$.*

Proof: Let *ABCD* be the smallest enclosing circle of an asymmetric configuration $\mathcal{C}$ of phase 7, Here *AB* is the lexicographically largest string. Now we have to plot the smallest enclosing rectangle of the target pattern with respect to our current coordinate system. This phase aims to move the head robot from the origin *A* to its fixed target position, which will be in the right direction of *A* on *AB*. As there may be vertical symmetry in target configuration when the head moves to its target then also vertical symmetry will happen. So in all the cases, *AB* will be a lexicographically larger string when the head robot moves to its target position. So when the head reaches its final position phase 7 terminates and $\neg S_0 \wedge S_1 \wedge S_2 \wedge S_9$ is true. ■

5.3.8. Phase 8

Tail moves downwards up to $tail_{target}$. In this phase, the position of the tail will be upward of $tail_{target}$. So when in phase 8 tail will move downwards to the point $tail_{target}$ (Figure 7). So when in phase 8 then $\neg S_0 \wedge S_1 \wedge S_2 =$ true, but after tail move then S_0 is true.

Theorem 5.10: *If we have a configuration $\mathcal{C}$ at some time t then after a finite number of moves by the tail S_0 becomes true.*

Proof: In this phase by a finite number of moves by the tail robot, the arbitrary pattern given as input will form. After phase 6 may be the configuration has symmetry or not.

(1) Let the configuration is asymmetric. Then the tail robot is now on the *CD* side, where *ABCD* is the smallest enclosing rectangle. Let *AB′C′D′* be the *s.rect* of the target configuration. Then $tail_{target}$ will be in the downwards vertical line of the tail. So when the tail robot moves down to its target position AB will be always a lexicographically larger string.
(2) Let the configuration is asymmetric but the position of $tail_{target}$ is on the upper side or in its horizontal line or downside. This is only possible when the initial configuration is the same as the target without the tail's position. In this case, the tail will move to its position. No symmetry will occur during this move of the tail.
(3) Let the configuration is symmetric. As the initial configuration is asymmetric the symmetry may arise in phase 7, so when there is a vertical symmetry then let *ABCD* be the *s.rect* and *AB* and *BA* be

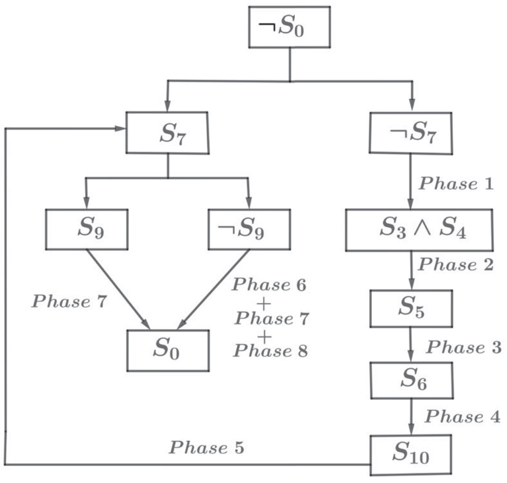

Figure 8. Flow Chart of the Algorithm APFOPTMOVE.

the larger string, as S_4 is true here so the target position for tail will be the downside of its recent position and after it moves and reaches to its $tail_{target}$ then S_0 is true. ■

In Figure 8, we presented the algorithm AAPFOPTMOVE by a flow chart. Starting from any asymmetric configuration where S_0 is not true, we can observe that the path ends, where S_0 is true, passing through a finite number of phases. So we can conclude that.

Theorem 5.11: *The APFOPTMOVE solves the Arbitrary pattern formation problem within finite time in $\mathcal{OBLOT}$ model.*

5.4. Move complexity of the algorithm

In [1], the author proved that any algorithm solving the arbitrary pattern formation problem in an infinite grid requires $\Omega(k\mathcal{D})$ moves, where $\mathcal{D}$ is the side of the smallest square that can contain both initial and target configuration, that is, $\mathcal{D} = \max\{AB, CD, A'B', C'D'\}$ and, k is the number of robots. Let $\mathcal{D}' = \max\{k, \mathcal{D}\}$. Then the mentioned result in [1] becomes that any algorithm solving the arbitrary pattern formation problem in an infinite grid requires $\Omega(\mathcal{D}'^2)$ moves. We show that our algorithm requires $O(\mathcal{D}')$ moves for each robot. Which gives the total number of required movements is $O(k\mathcal{D}') = O(\mathcal{D}'^2)$, which is asymptotically optimal. In Phase 1, Phase 2, and Phase 3 of our algorithm, only one robot moves. So a robot participating in these phases uses $O(\mathcal{D}')$ move. Then in phase 4 a robot comes down on the x-axis to form a compact line and in phase 5 a robot reaches its target position from the x-axis, so here also a robot in total has to move at most $4\mathcal{D}$ steps. In phase 6, phase 7, and phase 8 only one robot moves, and that robot needs to make $O(\mathcal{D}')$ moves. So we can conclude that any phase of our algorithm requires $O(\mathcal{D}')$ moves for a robot, which is asymptotically optimal. Since starting from any asymmetric configuration, our algorithm terminates via a finite number of phases among 8 phases, so we can finally conclude the following theorem.

Theorem 5.12: *Arbitrary pattern formation is solvable by optimal $O(\mathcal{D}'^2)$ moves in an asynchronous scheduler by oblivious robots from any asymmetric configuration.*

Now we observe that the time complexity of the APFOPTMOVE is $O(\mathcal{D}'^2)$. We observed in the proof of Theorem 5.12 that each robot makes $O(\mathcal{D}')$ moves. Hence, in the worst case for each robot $O(\mathcal{D}')$, epochs are sufficient. Since the movements of robots are sequential in this algorithm, the algorithm must terminate within $O(k\mathcal{D}') = O(\mathcal{D}'^2)$ epochs. We record this result in the following Theorem.

Theorem 5.13: *The* APFOPTMOVE *algorithm solves the* APF *problem within* $O(\mathcal{D}'^2)$ *epoch time under asynchronous scheduler.*

In the next section, we propose a faster algorithm solving the APF problem in $\mathcal{LUMI}$ model under an asynchronous scheduler.

6. Optimal time algorithm FastAPF for APF

This section proposes an algorithm, named FASTAPF for APF which is time optimal and move optimal as well. Before going to the algorithm, for convenience, let's go through some definitions, notions supporting the algorithm. Let *ABCD* be the unique smallest rectangle enclosing a given initial configuration, where $AB \geq BC$. If *ABCD* is not square then consider the set of strings *S* to be $\{\lambda_{AB}, \lambda_{BA}, \lambda_{CD}, \lambda_{DC}\}$. If *ABCD* is square then consider the set of strings *S* to be $\{\lambda_{AB}, \lambda_{BA}, \lambda_{CD}, \lambda_{DC}, \lambda_{BC}, \lambda_{CB}, \lambda_{AD}, \lambda_{DA}\}$.

6.1. Coordinate system setup

If the given configuration is asymmetric then *S* contains a lexicographically strictly largest string. Let λ_{AB} be the largest among other strings in *S*. Then robots consider *A* as the origin and *AB* direction as the positive *x*-axis and *AD* direction as the positive *y* -axis. In such a case, the first robot in λ_{AB} string is said to be *head*.

Each robot has a light. This light can take two colours, namely, HEAD and LINE which are readable as well as communicable. The light can indicate another state when the light is off. We denote this one as OFF colour.

Next suppose in a given configuration there is a robot, call it *head*, with HEAD colour ON on the boundary of *ABCD*. There are two cases, firstly if the head is not situated in any corner and another when the head is at a corner of the *ABCD*, say *A*. For the first case, we assume that the head is on the side *AB*, such that $AB \geq CD$. For such a case, consider *A* as origin, *AB* direction as the positive *x* -axis, and *AD* direction as the positive *y* -axis. For the second case, consider the head robot is situated at a corner, say *A*. In such a case, consider *A* as the origin. If $AB > BC$ then consider *AB* direction as positive *x*-axis and *AD* direction as positive *y* -axis. If $AB = BC$, then we assume the configuration is asymmetric and hence there is a lexicographically strictly largest string, say λ_{AB} is *S*. In such a case, consider the *AB* direction as a positive *x*-axis and *AD* direction as a positive *y* -axis.

Definition 6.1 (Tail robot): *Case-I:* (When HEAD colour is not ON) In this case, we assume that the configuration is asymmetric. The last robot in the lexicographically strictly largest string in *S* is said to be Tail.

Case-II: (When HEAD colour is ON) In this case, the tail is the rightmost robot of the topmost horizontal line.

Let the $A'B'C'D'$ be the smallest rectangle enclosing the target pattern with $A'B' \geq B'C'$. Let $\lambda_{A'B'}$ be the largest (may not be unique) among all other strings in *S* for the target pattern. We denote the *i*th target position in $\lambda_{A'B'}$ string as t_i. Then the target pattern is to be formed such that $A = A'$, $A'B'$ direction is along the positive *x*-axis and $A'D'$ direction is along the positive *y* -axis.

Let $s = \max\{AD, A'D'\}$. Lets name the lines parallel and above to *x*-axis by $H_1, H_1, \ldots, H_s$. Name the vertical lines from left to right as $V_1, V_2, \ldots,$ where V_1 is the *y*-axis. Let at any time $C(t)$ the configuration be $C(t)$. Let in the target pattern the number of robots in H_i line be n_i. Let in $C(t)$ the total number of robots below the line H_i be b_i. Let in $C(t)$ the total number of robots above the line H_i be a_i. Let

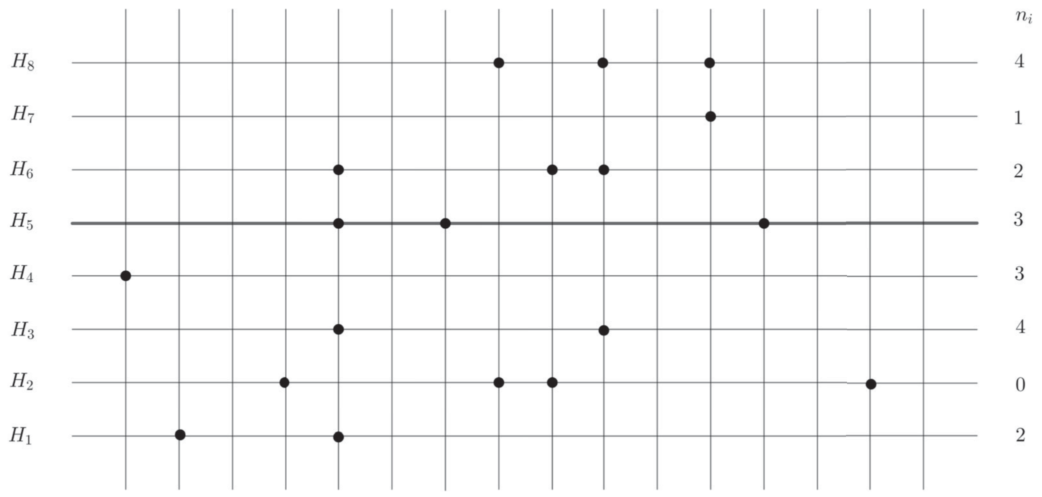

Figure 9. H_5 is a saturated line.

Table 3. A list of invariant parameters for a robot.

1	Smallest enclosing rectangle *ABCD* with $AB \geq BC$
2	If the configuration is asymmetric, lexicographically largest string λ_{AB}
3	Head robot
4	Tail robot

$b'_i = \sum_{j<i} n_i$ and $a'_i = \sum_{j>i} n_i$. A horizontal line H_i is said to be *saturated* if $a_i = a'_i$ and $b_i = b'_i$ In Figure 9 H_5 is a saturated line.

The next subsection describes all the intermediate procedures of `FastAPF`.

6.2. Elements of the algorithm

In order to optimise the time, our main motive is to make the algorithm so that it allows parallel movement of robot as much as possible avoiding collision. The algorithm is divided into six phases. Initially, since the configuration is asymmetric, a robot that activates at first can find out the things listed in Table 3.

Throughout the algorithm, the head robot remains head using HEAD colour as the flag. And further, the coordinate system remains unaltered throughout the algorithm that has been taken care of. Once the HEAD colour is ON and hence the head robot is fixed. Any robot other than the head or tail is said to be *inner* robot. Next, we define terminologies for different configurations.

(1) $\mathcal{C}_{init}$ = Initial configuration
(2) $\mathcal{C}_{target}$ = Target configuration
(3) $\mathcal{C}$ = A possible configuration from initial configuration
(4) $\mathcal{C}' = \mathcal{C} \setminus \{Head\}$
(5) $\mathcal{C}'' = \mathcal{C} \setminus \{Tail\}$
(6) $\mathcal{C}''' = \mathcal{C} \setminus \{Head, Tail\}$
(7) $\mathcal{C}'_{target} = \mathcal{C}_{target} \setminus \{Head\}$
(8) $\mathcal{C}''_{target} = \mathcal{C}_{target} \setminus \{Tail\}$
(9) $\mathcal{C}'''_{target} = \mathcal{C}_{target} \setminus \{Head, Tail\}$

Next, we define different conditions on configuration.

(1) $C_0 : \mathcal{C} = \mathcal{C}_{target}$
(2) $C_1 : \mathcal{C}' = \mathcal{C}'_{target}$
(3) $C_2 : \mathcal{C}'' = \mathcal{C}''_{target}$
(4) $C_3 : \mathcal{C}''' = \mathcal{C}'''_{target}$
(5) C_4 : HEAD colour is ON.
(6) C_5 : Head colour is at corner.
(7) C_6 : All inner robots are with LINE colour ON except those who are on a saturated line.
(8) C_7 : Tail is at a point with x-coordinate $\max\{AB+1, A'B'+1, k\}$ and y-coordinate $\max\{BC, B'C'\}$.

(1) *Procedure-I: Input*: $\neg C_3 \vee (C_3 \wedge \neg C_1 \wedge \neg C_2)$

In the first procedure, Head identifies itself and turns its HEAD colour on. Then it goes to origin if it is not.

Output: $C_4 \wedge C_5$ is true.

Discussion This procedure gets executed when either there is an inner robot not at its target position or all inner robots are at their target but neither head nor tail is at its target position. In such a case, the head robot first puts ON its HEAD colour and then moves leftwards until it reaches its origin. Note that if the input configuration is asymmetric then the configuration throughout the procedure remains asymmetric. And also the things listed in Table 3 remain unchanged.

(2) *Procedure-II: Input*: $C_1 \wedge \neg C_2$

In this procedure, the head moves to its target position and turns off its HEAD colour if it is ON.

Output: C_0 is true.

Discussion This procedure gets executed when every robot except the head is at their respective target position. In this phase, the head occupies its target position and turns off its HEAD colour if it is ON. Now the head target must be on the x-axis, so either head needs to move right or left to reach its destination in this procedure. If the head needs to move left then clearly the things listed in Table 3 remain unchanged. Also in the other case, since the head target is the first target position of the lexicographically largest string of the target pattern, the listed things in Table 3 remain unchanged.

(3) *Procedure-III: Input*: $(C_2 \wedge \neg C_1) \vee (C_4 \wedge C_5 \wedge C_3)$

Case-I: If the y-coordinate of the tail target is the same as the y-coordinate of the tail, then the tail reaches at target by horizontal movements.

Case-II: If the y-coordinate of the tail target is not the same as the y-coordinate of the tail, then we consider the following cases.

Case-IIA: If the y-coordinate of the tail target is greater than the y-coordinate of the tail, then move upwards until Case-I is achieved.

Case-IIB: If the y-coordinate of the tail target is less than the y-coordinate of the tail, move horizontally so that the x-coordinate of the tail target becomes the same as the x-coordinate of the tail. Then the tail moves downwards until condition C_0 is true.

Output: $C_0 \vee C_1$ is true.

Discussion This procedure gets executed when either the pattern except the tail is formed ($C_2 \wedge \neg C_1$) or all inner robots are at their target position and the head is at the origin with head colour ON ($C_4 \wedge C_5 \wedge C_3$). For both cases, careful movement is assigned to the tail robot. For all the cases it can be checked that the things listed in Table 3 remain unchanged. If the input configuration was $C_2 \wedge \neg C_1$, after execution of this procedure the target pattern is formed (C_0). And if the input pattern was $C_4 \wedge C_5 \wedge C_3$ then after execution of this phase the resulting configuration is such that the target pattern is formed except head robot.

(4) *Procedure-IV: Input*: $C_4 \wedge C_5 \wedge \neg C_3 \wedge \neg C_7$

In this procedure, the tail moves right until its x-coordinate becomes $\max\{AB+1, A'B'+1, k\}$. Then the tail moves upward until its y-coordinate becomes $\max\{BC, B'C'\}$.

Output: C_7 is true.

Discussion This procedure gets executed when at least one inner robot is not at its target position and the head robot is at its origin. In this procedure, the robot moves rightwards in order to ensure that there is enough big smallest rectangle of the current configuration for procedure-V and procedure-VI to execute and also to ensure that $AB > BC$ so that the coordinate system does not change.

(5) *Procedure-V: Input*: $C_4 \wedge C_5 \wedge \neg C_3 \wedge C_7 \wedge \neg C_6$

In this procedure, if an inner robot is not on a saturated horizontal line then it checks whether its LINE colour is ON or not. If the LINE colour is not ON, then it counts the number of robots below it. Let's say the number is b. Then the robot counts the number, say, l of robots on the left side to it in its horizontal line. Let $v = b + l + 1$. Then the robot tries to move to the vth vertical line in its horizontal line. If vth vertical line is in its left (right) and its left (right) grid point is empty then it moves to left (right). When a robot reaches the vth vertical line, the robot turns on its LINE colour.

Output: C_6 is true (Figure 10).

Discussion These procedures are executed when the head robot is at the origin with its HEAD colour ON and the tail robot's coordinate is $(\max\{AB + 1, A'B' + 1, k\}, \max\{BC, B'C'\})$ and there is at least one inner robot which is not on the saturated line and with no line colour ON. In this procedure, such robot reaches at vth vertical line and turns ON its LINE colour. At the end of this procedure in the configuration, the head and tail remain at their starting position and each inner robot is either with LINE colour ON or on a saturated line. In these procedures since no robot jumps through horizontal lines, so no collision of robots takes place.

(7) *Procedure-VI: Input*: $C_4 \wedge C_5 \wedge \neg C_3 \wedge C_7 \wedge C_6$

In this procedure, there are two exhaustive cases.

Case-I: This is the case when the robot sees that its own horizontal line is not saturated. In such a case, the robot counts its vertical line number. Let the robot be at v_ith vertical line. Then it moves to the horizontal line which contains the v_ith target position by vertical movements until it reaches there (Figure 11).

Case-II: This is the case when the robot sees that its own horizontal line is saturated. In this case firstly if its LINE colour is ON, then it turns it OFF. Otherwise, if the robot is the kth robot on its horizontal line from the left, then it tries to reach the kth target position on that horizontal line. If kth target position in its horizontal line is in its left (right) and its left (right) grid point is empty then it moves to left (right).

Output: C_3 is true.

Discussion In the input configuration of this procedure an inner robot is either on a saturated line or not on a saturated line but with its LINE colour ON. In this procedure, no two inner robots with LINE colour ON are on the same vertical line. In this phase, two types of movements are happening simultaneously. Firstly robots with line colour ON but not on a saturated line do vertical movements and rests do horizontal movements. Eventually, all robots will reach their destined horizontal line and all horizontal lines will become saturated. Next, we need to show no two robots collide in this procedure. We can guarantee that if no robot with LINE colour reaches a saturated line. Now from the definition of a saturated line, one can note that every robot below a saturated line has its target horizontal line below the saturated line. And also every robot above a saturated line has its target horizontal line above the saturated line. Hence no robot making vertical movement will reach a saturated line where horizontal movement is possibly happening. Hence this procedure is collision-free.

The next subsection formally presents the algorithm `FastAPF` in flow chart form.

6.3. Algorithm FastAPF

The main algorithm FASTAPF is presented below in the flow chart in Figure 12.

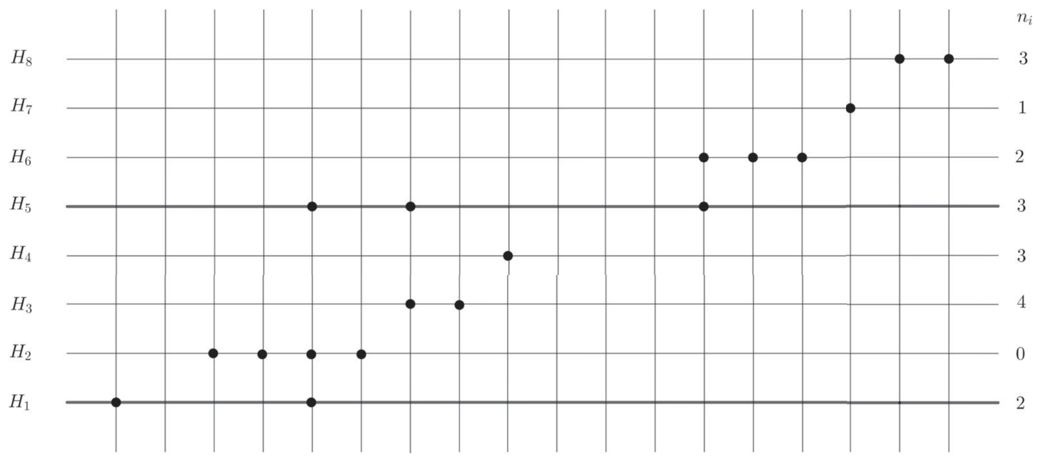

Figure 10. When C_6 is true.

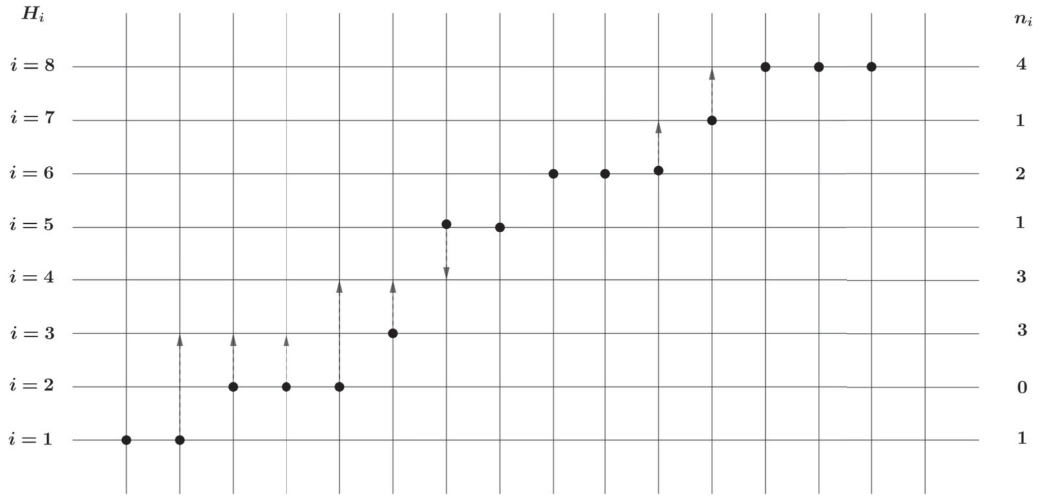

Figure 11. Procedure-VI illustration.

Starting from any possible configuration where C_0 is not true, in the flow chart, we can observe that the path ends to the C_0 configuration passing through some procedures. Hence we conclude the following.

Theorem 6.2: *The* FastAPF *algorithm solves the* Apf *problem in* $\mathcal{LUMI}$ *model.*

6.4. Time complexity and movement analysis of FastAPF algorithm

We show that each of six procedures included in FastAPF algorithm takes $O(\mathcal{D}')$ epoch. The flow chart in Figure 12 shows that starting from any asymmetric initial configuration the algorithm terminates via executing a finite number of these procedures. This will prove our claim that, the algorithm FastAPF solves Apf problem in $O(\mathcal{D}')$ epochs.

In Procedure-I, Procedure-II, Procedure-III, and Procedure-IV only one robot is making its move. And in each of these procedures, a robot does $O(\mathcal{D})$ moves, so all the phases can take $\mathcal{D}$ epochs in the worst

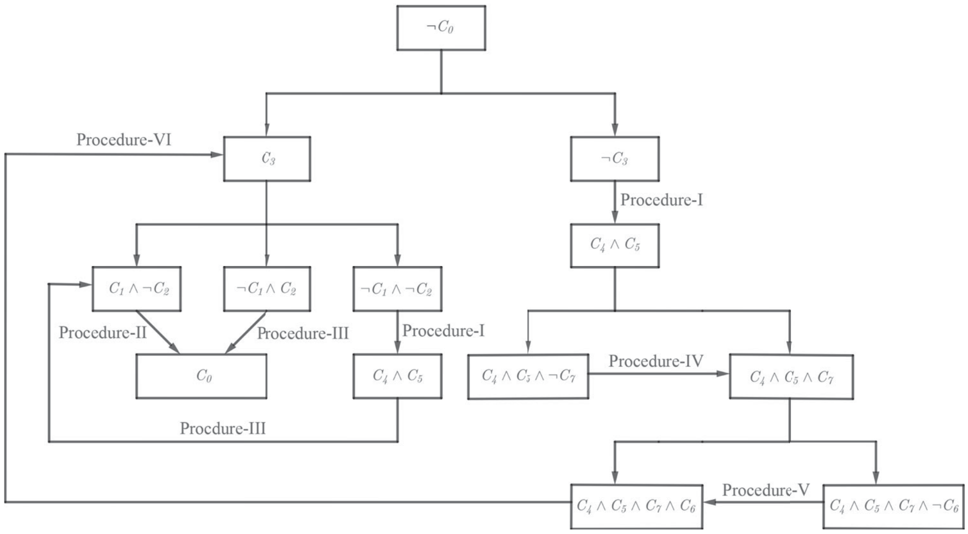

Figure 12. Algorithm FASTAPF flow chart.

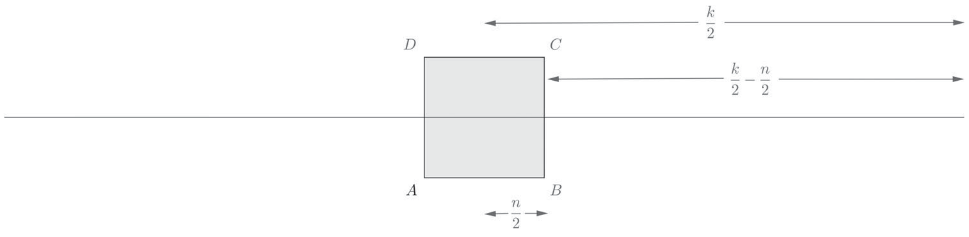

Figure 13. An image related to Theorem 6.4.

case to complete. In Procedure-V in each horizontal line, robots are making horizontal parallel movements. On a horizontal line H_i, for all robots to reach the destined vertical line maximum it will take $2\mathcal{D}'$ epochs. Hence Procedure-V takes at most $2\mathcal{D}'$ epochs to complete. In procedure-VI, a robot with LINE colour makes vertical movements. All vertical movements in this procedure happen simultaneously. So all the vertical movements are done in $\max\{BC, B'C'\}$ epochs. Then in a saturated line horizontal movements happen in order to reach the target positions. Similarly, this also takes $2\mathcal{D}'$ epochs in the worst case. Hence this procedure also takes $O(\mathcal{D}')$ epochs. Hence we conclude this discussion with the following theorem.

Theorem 6.3: *The FASTAPF algorithm solves the APF problem in $\mathcal{LUMI}$ model in $O(\mathcal{D}')$ epochs.*

Next, we show that the algorithm proposed in [1] takes $O(\mathcal{D}'^2)$ epochs to complete. Which will prove that the FASTAPF algorithm is faster. In phase 4 of the proposed algorithm in [1], a robot might need to make $\Omega(\mathcal{D}^2)$ movements. Therefore, this algorithm requires $\Omega(\mathcal{D}'^2)$ epoch time to complete. We state this in the following result.

Result 1: Algorithm proposed in [1] requires $\Omega(\mathcal{D}'^2)$ epoch time to solve Arbitrary Pattern Formation problem.

Now we show that our algorithm is asymptotically optimal in time. Let's consider an initial configuration of robots has the smallest enclosing rectangle a square of length n and each grid point of the square has a robot. Thus there are $k = n^2$ robots in total. Let the target pattern be a compact line. Then we can see that the farthest point on the line (wherever it may be placed) from the initial configuration is at least $\frac{k-n}{2} = \frac{k}{2} - \frac{\sqrt{k}}{2}$ (See Figure 13). Hence there is a robot that at least needs to move $\frac{k}{2} - \frac{\sqrt{k}}{2}$ steps. Hence at least $\frac{k}{2} - \frac{\sqrt{k}}{2}$ epochs are necessary. In this case, note that $\mathcal{D} = k$. Hence we can state the following.

Theorem 6.4: *Any algorithm solving* Apf *problem requires $\Omega(\mathcal{D}')$ epochs.*

The above result shows that the algorithm FastAPF is time optimal asymptotically. Next, we show that algorithm FastAPF is also move optimal.

Theorem 6.5: *Algorithm FastAPF all total requires at most $O(\mathcal{D}'^2)$ robot movements.*

Proof: Any robot participating in Procedure-I, Procedure-II, Procedure-III, or Procedure-IV makes at most $2\mathcal{D}'$ moves. Next any robot participating in Procedure-V only makes a horizontal move of length at most $\mathcal{D}'$. Next any robot participating in Procedure-VI makes at most a vertical move (if the robot is not on a saturated horizontal line) of maximum length $\mathcal{D}'$ followed by a horizontal move of maximum length $\mathcal{D}'$. Hence for each procedure, a robot participating in that procedure makes $O(\mathcal{D}')$ moves in that procedure. Since starting from any asymmetric configuration the algorithm FastAPF terminates via passing through a finite number of procedures. Hence for each robot, it needs to make $O(\mathcal{D}')$ moves. Thus, the total number of required robot moves in algorithm FastAPF is $O(k\mathcal{D}') = O(\mathcal{D}'^2)$. ■

7. Conclusion

This paper studies algorithms for Arbitrary Pattern Formation (Apf) problem on an infinite grid by a robot swarm starting from any asymmetric initial configuration in classical $\mathcal{OBLOT}$ and $\mathcal{LUMI}$ robotic model. This work gives an algorithm for the Apf problem in $\mathcal{OBLOT}$ model which is asymptotically move optimal. Then this work proposed another algorithm for the Apf problem in $\mathcal{LUMI}$ model which is asymptotically time optimal and move optimal as well. If $\mathcal{D}'$ is the maximum of the number of robots and the side of the smallest enclosing square enclosing both target and initial configuration, then the first algorithm uses total $O(\mathcal{D}'^2)$ movements and the second algorithm takes total $O(\mathcal{D}')$ epoch time and total $O(\mathcal{D}'^2)$ moves to solve the Apf problem. Unfortunately, the move optimal algorithm in $\mathcal{OBLOT}$ model is not time optimal, so studying for an algorithm in $\mathcal{OBLOT}$, which is both move optimal and time optimal, could be a possible direction from this work. One can consider another version of Apf problem where the initial or the target configuration of robots can have multiplicity points.

Acknowledgement

We want to thank our colleague Dr. Kaustav Bose for his valuable advice in this work. We also want to thank the anonymous reviewers for their valuable comments and suggestions which helped us to improve the quality of this paper.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by West Bengal State Government Fellowship Scheme University Grants Commission [P-1/RS/23/2020].

ORCID

Satakshi Ghosh http://orcid.org/0000-0003-1747-4037
Pritam Goswami http://orcid.org/0000-0002-0546-3894
Avisek Sharma http://orcid.org/0000-0001-8940-392X
Buddhadeb Sau http://orcid.org/0000-0001-7008-6135

References

[1] Bose K, Adhikary R, Kundu MK, et al. Arbitrary pattern formation on infinite grid by asynchronous oblivious robots. Theor Comput Sci. 2020;815:213–227. DOI:10.1016/j.tcs.2020.02.016.
[2] Suzuki I, Yamashita M. Distributed anonymous mobile robots – formation and agreement problems. In: Proceedings of the 3rd International Colloquium on Structural Information and Communication Complexity (SIROCCO '96); 1996. p. 1347–1363.
[3] Yamashita M, Suzuki I. Characterizing geometric patterns formable by oblivious anonymous mobile robots. Theor Comput Sci. 2010;411(26):2433–2453. Available from: https://www.sciencedirect.com/science/article/pii/S0304397510000745.
[4] Flocchini P, Prencipe G, Santoro N, et al. Arbitrary pattern formation by asynchronous, anonymous, oblivious robots. Theor Comput Sci. 2008;407(1–3):412–447. DOI:10.1016/j.tcs.2008.07.026.
[5] Dieudonné Y, Petit F, Villain V. Leader election problem versus pattern formation problem. In: Lynch NA, Shvartsman AA, editors. Proceedings of the 24th International Symposium on Distributed Computing, DISC 2010, September 13–15; Cambridge (MA): Springer; 2010. p. 267–281. (Lecture Notes in Computer Science; 6343). DOI:10.1007/978-3-642-15763-9_26.
[6] Bramas Q, Tixeuil S. Arbitrary pattern formation with four robots. In: Izumi T, Kuznetsov P, editors. Proceedings of 20th International Symposium on Stabilization, Safety, and Security of Distributed Systems, (SSS 2018), November 4–7; Tokyo, Japan: Springer; 2018. p. 333–348. (Lecture Notes in Computer Science; Vol. 11201). DOI:10.1007/978-3-030-03232-6_22.
[7] Bose K, Das A, Sau B. Pattern formation by robots with inaccurate movements. In: Bramas Q, Gramoli V, Milani A, editors. 25th International Conference on Principles of Distributed Systems, (OPODIS 2021), December 13–15, 2021, Strasbourg, France: Schloss Dagstuhl – Leibniz-Zentrum für Informatik; 2021. p. 1–20. (LIPIcs; 217). DOI:10.4230/LIPIcs.OPODIS.2021.10.
[8] Bramas Q, Tixeuil S. Probabilistic asynchronous arbitrary pattern formation (short paper). In: Bonakdarpour B, Petit F, editors. Proceedings of 18th International Symposium, Stabilization, Safety, and Security of Distributed Systems, (SSS 2016), November 7–10; Lyon, France; 2016. p. 88–93. (Lecture Notes in Computer Science; 10083). DOI:10.1007/978-3-319-49259-9_7.
[9] Cicerone S, Stefano GD, Navarra A. Embedded pattern formation by asynchronous robots without chirality. Distributed Comput. 2019;32(4):291–315. DOI:10.1007/s00446-018-0333-7.
[10] Cicerone S, Fonso AD, Stefano GD, et al. Arbitrary pattern formation on infinite regular tessellation graphs. CoRR. 2020;abs/2010.14152. Available from: https://arxiv.org/abs/2010.14152.
[11] Yamauchi Y, Yamashita M. Pattern formation by mobile robots with limited visibility. In: Moscibroda T, Rescigno AA, editors. 20th International Colloquium on Structural Information and Communication Complexity, SIROCCO 2013, July 1–3, Revised Selected Papers; Ischia, Italy: Springer; 2013. p. 201–212. (Lecture Notes in Computer Science; 8179). DOI:10.1007/978-3-319-03578-9_17.
[12] Lukovszki T. Fast collisionless pattern formation by anonymous, position-aware robots. In: Aguilera MK, Querzoni L, Shapiro M, editors. Proceedings of 18th International Conference Principles of Distributed Systems, OPODIS 2014, December 16–19, Cortina d'Ampezzo, Italy: Springer; 2014. p. 248–262. (Lecture Notes in Computer Science; 8878). DOI:10.1007/978-3-319-14472-6_17.
[13] Adhikary R, Bose K, Kundu MK, et al. Mutual visibility by asynchronous robots on infinite grid. In: Gilbert S, Hughes D, Krishnamachari B, editors. Algorithms for Sensor Systems – 14th International Symposium on Algorithms and Experiments for Wireless Sensor Networks, ALGOSENSORS 2018, August 23–24, Revised Selected Papers; Helsinki, Finland: Springer; 2018. p. 83–101. (Lecture Notes in Computer Science; 11410). DOI:10.1007/978-3-030-14094-6_6.
[14] Stefano GD, Navarra A. Gathering of oblivious robots on infinite grids with minimum traveled distance. Inf Comput. 2017;254:377–391. DOI:10.1016/j.ic.2016.09.004.
[15] Bose K, Kundu MK, Adhikary R, et al. Arbitrary pattern formation by asynchronous opaque robots with lights. Theor Comput Sci. 2021;849:138–158. DOI:10.1016/j.tcs.2020.10.015.
[16] Bose K, Adhikary R, Kundu MK, et al. Arbitrary pattern formation by opaque fat robots with lights. CoRR. 2019;abs/1910.02706. Available from: http://arxiv.org/abs/1910.02706.
[17] Kundu MK, Goswami P, Ghosh S, et al. Arbitrary pattern formation by asynchronous opaque robots on infinite grid. CoRR. 2022;abs/2205.03053. DOI:10.48550/arXiv.2205.03053.

[18] Kundu MK, Goswami P, Ghosh S, et al. Arbitrary pattern formation by opaque fat robots on infinite grid. International Journal of Parallel, Emergent and Distributed Systems. 2022 Jun:1–29. DOI:10.1080%2F17445760.2022.2088750.

[19] Vaidyanathan R, Sharma G, Trahan J. On fast pattern formation by autonomous robots. Inform Comput. 2021;285:104699. Available from: https://www.sciencedirect.com/science/article/pii/S0890540121000146.

[20] Feletti C, Mereghetti C, Palano B. Uniform circle formation for swarms of opaque robots with lights. In: Izumi T, Kuznetsov P, editors. Proceedings of the 20th International Symposium, Stabilization, Safety, and Security of Distributed Systems, SSS 2018, November 4–7; Tokyo, Japan: Springer; 2018. p. 317–332. (Lecture Notes in Computer Science; 11201). DOI:10.1007/978-3-030-03232-6_21.

[21] Adhikary R, Kundu MK, Sau B. Circle formation by asynchronous opaque robots on infinite grid. Comput Sci. 2021;22(1). DOI:10.7494/csci.2021.22.1.3840.

[22] Das S, Flocchini P, Santoro N, et al. Forming sequences of geometric patterns with oblivious mobile robots. Distributed Comput. 2015;28(2):131–145. DOI:10.1007/s00446-014-0220-9.

[23] Das S, Flocchini P, Prencipe G, et al. Forming sequences of patterns with luminous robots. IEEE Access. 2020;8:90577–90597. DOI:10.1109/ACCESS.2020.2994052.

[24] Fujinaga N, Yamauchi Y, Ono H, et al. Pattern formation by oblivious asynchronous mobile robots. SIAM J Comput. 2015;44(3):740–785. DOI:10.1137/140958682.

[25] Cicerone S, Stefano GD, Navarra A. Asynchronous arbitrary pattern formation: the effects of a rigorous approach. Distributed Comput. 2019;32(2):91–132. DOI:10.1007/s00446-018-0325-7.

[26] Pattanayak D, Foerster K, Mandal PS, et al. Conic formation in presence of faulty robots. In: Pinotti CM, Navarra A, Bagchi A, editors. 16th International Symposium on Algorithms and Experiments for Wireless Sensor Networks, Algorithms for Sensor Systems, ALGOSENSORS 2020, September 9–10, Revised Selected Papers; Pisa, Italy: Springer; 2020. p. 170–185. (Lecture Notes in Computer Science; 12503): DOI:10.1007/978-3-030-62401-9_12.

[27] Suzuki I, Yamashita M. Distributed anonymous mobile robots: formation of geometric patterns. SIAM J Comput. 1999;28(4):1347–1363. DOI:10.1137/S009753979628292X.