Distributed Uniform Partitioning of a Region using Opaque ASYNC Luminous Mobile Robots

Subhajit Pramanick, Saswata Jana, Adri Bhattacharya, and Partha Sarathi Mandal

Department of Mathematics
Indian Institute of Technology Guwahati, India
psm@iitg.ac.in

Abstract. We are given N autonomous mobile robots inside a bounded region. The robots are opaque which means that three collinear robots are unable to see each other as one of the robots acts as an obstruction for the other two. They operate in classical *Look-Compute-Move* (LCM) activation cycles. Moreover, the robots are oblivious except for a persistent light (which is why they are called *Luminous robots*) that can determine a color from a fixed color set. Obliviousness does not allow the robots to remember any information from past activation cycles. The Uniform Partitioning problem requires the robots to partition the whole region into sub-regions of equal area, each of which contains exactly one robot. Due to application-oriented motivation, we, in this paper consider the region to be well-known geometric shapes such as rectangle, square and circle. We investigate the problem in *asynchronous* setting where there is no notion of common time and any robot gets activated at any time with a fair assumption that every robot needs to get activated infinitely often. To the best of our knowledge, this is the first attempt to study the Uniform Partitioning problem using oblivious opaque robots working under asynchronous settings. We propose three algorithms considering three different regions: rectangle, square and circle. Robots partition the region in a distributed way and reach their respective positions in the partitions. The algorithms proposed for rectangular and square regions run in $O(N)$ epochs whereas the algorithm for circular regions runs in $O(N^2)$ epochs, where an epoch is the smallest unit of time in which all robots are activated at least once and execute their LCM cycles.

Keywords: Distributed algorithms, Multi-agent systems, Mobile robots, Uniform Partitioning, Luminous Robots

1 Introduction

1.1 Motivation

Distributed algorithms for many real-world problems grabbed a lot of attention from researchers for many years now. A swarm of mobile robots is a very important tool in designing such distributed algorithms. The collaborative actions of

these robots achieve the end goal of the problems. One such problem is *Uniform Partitioning* of a bounded region, which is a very common problem in our daily life. For example, if a number of people are asked to paint a wall, the natural strategy is to divide the region into equal parts and assign each person to a distinct part of the wall for painting. Keeping the same motivation in mind, autonomous mobile robots can be more useful in much more critical situations like cleaning spillage of liquid radioactive waste in a laboratory. In such hazardous situations, robots are the safest options for us. Uniform partitioning is equally applicable to another scenario where a city needs to be well-covered with networks. A group of autonomous drones, each of which is equipped with necessary instruments, can be deployed to serve the purpose, where drones should position themselves in such a way that each of them covers a part of the whole city of the same area as others.

Classically, these robots are autonomous (no external control), homogeneous (execute the same algorithm), anonymous (having no unique identifier) and disoriented (do agree on any global coordinate system or orientation). This robots are modelled as points on the plane. They are equipped with vision which enables them to gather information from the surroundings. These robots operate in *Look-Compute-Move* cycles, which we define later in the paper. Two robots might not be able to see each other due to the presence of other robots between them (it is called *obstructed visibility model*). These robots are called *opaque* robots. There is no means of communication for the robots except an externally visible persistent light on them, which can determine color from a prefixed color set. Since there is a fixed number of colors, this type of communication is considered as a weak form of communication between the robots. Other than this light, robots do not have any persistent memory to store past information (this type of robot is called *oblivious luminous robots*).

We, in this paper, initiate the study of distributed uniform partitioning of a bounded region using opaque luminous mobile robots. The activation schedule of the robots is *asynchronous* (ASYNC) where any robots can be activated at any time and there is no notion of a global time. The primary motive is to use a swarm of mobile robots with assumptions as less as possible. The problem aims to arrange the robots in such a way that the region gets divided into equal partitions and each partition contains exactly one robot. We retain our focus on this objective only, whereas the problem could even be extended to a version where the robots have sufficient memory or some extra ability to store the coordinates of the partitions. The problem gets challenging due to the oblivious nature of the robots. The obstructed visibility of the robots adds to the challenge even more, because it is not always possible to count the number of robots present in the region. We also assume that the robots do not have any knowledge about the total number of robots. Although mutual visibility algorithms can be used to make any three robots non-collinear and robots can count the total number of robots, the obliviousness of the robots disables them to keep the information beyond the current LCM cycle. From the application point of view, we assume that the robots can detect the boundary of the region. We propose algorithms

for the robots considering the region to be a standard geometric shape such as a rectangle, square or circle. Moreover, the algorithms are collision-free which means that two robots can never collocate at the same point.

1.2 Contributions

Our contributions, in this paper, are listed below.

- To the best of our knowledge, this is the first work towards the problem of distributed uniform partitioning of a bounded region $\Re$ using a swarm of N ASYNC oblivious opaque mobile robots having no global axes agreement.
- We propose an algorithm when $\Re$ is a rectangle. The algorithm runs in $O(N)$ epochs and uses 4 colors.
- We propose an algorithm when $\Re$ is a square, which runs in $O(N)$ epochs and uses 7 colors.
- We propose an algorithm when $\Re$ is a circle, which runs in $O(N^2)$ epochs and uses 9 colors.

1.3 Related works

Distributed algorithms using mobile robots have been widely studied for many years. Some of the popular topics in this area are gathering, pattern formation, mutual visibility, etc. These problems are studied using different robot models in the literature that provide us with an extensive idea about how mobile robots work. In various literature [7,10,2], the pattern formation problem is studied with mobile robots with different capabilities. Mutual visibility is another popular area where mobile robots are used. To achieve mutual visibility, the movement of the robots needs to be designed carefully enough to avoid collision between robots. Several papers [6,9,12] highlight this important aspect of mobile robots which is an essential part of designing the movement of the robot because collision might lead to a multiplicity points, whereas multiplicity detection is a costlier task. Saha et al. [11] inspected surveillance of uneven surfaces using drones which falls into the category of coverage problem. Their target is to give a compact coverage of the area so that the diameter of the drone network gets minimized. As mentioned earlier, painting a bounded region is an application of the uniform partitioning problem. Das and Mukhapadhyaya [3] proposed an asynchronous algorithm for distributed painting in a rectangular region with a robot swarm where robots have agreement on a global line. The algorithm divides the region into uniform horizontal strips, but one of the advantages in this paper is that the robots are transparent which enables them to see all other robots in the region in every activation. Later in [4], they studied the distributed painting with robots having limited visibility and a global coordinate system. Robots are not completely oblivious and transparent so a robot can see all the robots within its visibility range. Das et al. [5] extended the distributed painting when the rectangular region has opaque obstacles in it. They consider the robots

to be transparent and work under semi-synchronous settings having total agreement in the direction and orientation of their local coordinate systems. Pavone et al. [8] proposed an algorithm for equitable partitioning of an environment using a team of mobile agents working synchronously. Their algorithm uses the Voronoi-based partitioning approach, but our proposed algorithms can partition a region uniformly using mobile robots working under asynchronous setting in a much simpler way. Acevedo et al. [1] gave an algorithm for partitioning a known region using aerial robots, but the partitioning is not uniform.

2 Model

Robots: A set of N mobile robots $\{r_1, r_2, \cdots r_N\}$ is deployed at distinct points within a bounded region. This region can be thought of as a subset of the Euclidean plane. Each robot is considered as a point on the plane. The robots are *autonomous*, *anonymous*, *homogeneous* and *disoriented*. We consider the robots to be *opaque* because of which a robot r_i is unable to see the robot r_k if there is a robot r_j lying on the line segment joining r_i and r_k. However, robots can see the boundary of the region from any point within the region. A robot is visible to itself, but might not be able to see all the robots present inside the region, due to obstructed visibility. Moreover, robots have no knowledge about N, the total number of robots. Each robot has its local coordinate system where the current position of the robot is considered as the origin of its coordinate system.

Each robot has a persistent light on it which is externally visible to all other visible robots. This light can determine color from a prefixed set of colors. These colors enable the robots to have a weak form of communication among themselves. $r_i.color$ represents the current color of the robot r_i at any time. Other than this persistent light, robots do not have any other memory to remember any information from the past. We misuse the notation r to denote the current position of the robot r. Two robots exhibit collision if they are collocated at a point at the same time.

Region: We are given with a bounded region, denoted by $\Re$. The *interior* of the region $\Re$, denoted by $Int(\Re)$ is defined to be the part of $\Re$ without the boundary. In this paper, we consider the region to be a standard geometric region such as a rectangle, square or circle. A robot lying on $Int(\Re)$, is called an *interior robot*. A robot is a *boundary robot* when it lies on the boundary of $\Re$, but not on the corners. When $\Re$ has corners, robots lying on them are called *corner robots*. The boundary of $\Re$ is identifiable by the robots from any point in the region which enables them to identify whether the region is a rectangle, square or a circle. For any two points A and B in the region, $\overline{AB}$ denotes the line segment joining A and B. $\overleftrightarrow{AB}$ is the line passing through the two points. The length of a line segment $\overline{AB}$ is represented by $len(\overline{AB})$. We denote the distance between two points A and B by $d(A, B)$. A similar notation $d(p, L)$ is used to represent the shortest distance between a point p and a line L.

Activation Cycle: We consider the mobile robots operating in the *Look-Compute-Move* (LCM) model, in which the actions of the robots are divided into three phases.

- *Look:* The robot takes a snapshot of its surroundings, i.e., the vertices within the visibility range and the colors of the robots occupying them.
- *Compute:* The robot runs the algorithm using the snapshot as the input and determines a target vertex or chooses to remain in place. It also changes its current color, if necessary.
- *Move:* The robot moves to the target vertex if needed.

Scheduler and Run-time: Robots are activated under a fair *asynchronous* (ASYNC) scheduler. There is no common notion of time in the ASYNC setting. Any number of robots can be activated at any time, with the fairness assumption that each robot is activated infinitely often. Time is measured in terms of *epochs* which is the smallest time interval in which every robot gets activated and executes its LCM cycle at least once.

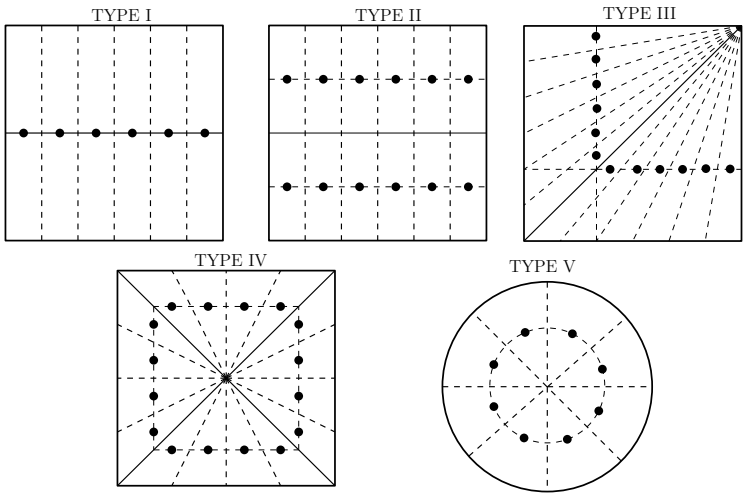

Fig. 1: Illustrating different types of partitioning

Problem Definition (Distributed Uniform Partitioning): N oblivious, opaque, luminous point robots are deployed at arbitrary distinct points on a bounded region $\Re$. The robots neither have any global agreement on the coordinate axis nor any knowledge of N. Each of them operates in *Look-Compute-Move* activation cycles and has a persistent light attached to it that can assume a color from a predefined color set. The objective is to divide the region $\Re$ into N uniform

partitions using N mobile robots such that each partition contains exactly one robot.

Partition Types: Depending on the positions and distribution of the robots over the region $\Re$, robots decide the type of partitioning in a distributed manner. All the partitions should be the same area. We identify four types of partitioning for the rectangular and square regions, as shown in Fig. 1. When the region $\Re$ is a rectangle, robots follow either Type I or Type II partitioning, whereas, in the case of a square region, robots terminate in one of the four types. In the case of a circular region, Type V partitioning is followed.

Table 1: The list of colors with their Specification

Region	**Color**	**Specification**
Rectangle	`OFF`	Initial color
	`MONITOR`	Used when the robot is moving to apex point to count the number of robots
	`MOVE`	Used when the robot is moving to the final position for Type I and II partitions
	`FINISH`	Used by robots at termination for Type I and II partitions
Square	`OFF`	Initial color
	`MONITOR`	Used when the robot moves to the apex point to count the number of robots in case of robots are only on one or two opposite sides of $\Re$
	`MONITOR1`	Used when the robot is moving to the apex point to count the number of robots in case of robots are not on one or two opposite sides of $\Re$
	`MOVE`	Used when the robot is moving to final position for Type I and II partitions
	`FINISH`	Used by robots at termination for Type I and II partitions
	`FINISH1`	Used by robots at termination for Type III partition
	`FINISH2`	Used by robots at termination for Type IV partition
Circle	`OFF`	Initial color of the robots
	`HEAD`	Used for the head of an eligible cluster
	`TAIL`	Used for the tail of an eligible clusters
	`MID`	Used for the middle robots of an eligible cluster
	`MOVE-H`	Used for the movement of the head towards the other cluster when the head moves half of the excess distance
	`MOVE-F`	Used for the movement of the head towards the other cluster when the head moves the excess distance completely
	`HALF`	Used after `MOVE-H`
	`FULL`	Used after `MOVE-F`
	`FINISH`	Used by robots at termination

Organization: Rest of the paper is organized as follows. Section 3 discusses the algorithm for the rectangular region. Section 4 explores the algorithm when the

region is a square. Section 5 mentions the algorithm for the circular region. In Section 6, we discuss about the semi-synchronous setting. Finally, we conclude in Section 7.

3 Algorithm when $\Re$ is a Rectangle

In this section, we consider the region $\Re$ to be a rectangle. Initially, robots are deployed over distinct points on the rectangle $\Re$ with color `OFF`. Robots will follow either Type I or Type II partitioning, which is ensured by our algorithm. The algorithm, in this case, uses 4 colors which are described in the Table 1. Our strategy is to bring the robots to the boundary of the rectangle first. Then, each robot will occupy a point of a partition of the rectangle. We define some notations that will be used to explain the algorithm. We must note that the variables are defined with respect to the local coordinate system of a robot r. A point visible to two different robots can be perceived differently, depending on their coordinate system.

Notations: For any robot r, we define the following notations.

- S_r is the nearest longest side of the rectangle $\Re$ to r.
- S_r^{opp} denotes the side opposite to S_r in the rectangle.
- L_r is the line perpendicular to S_r that passes through r.
- p_r is the point of intersection of the two lines L_r and S_r.
- The two endpoints of a side S of $\Re$ is denoted by e_S^1 and e_S^2.
- For two opposite sides S and S' of $\Re$ and a constant k, kS is the line segment of length $len(S)$ parallel to the side S that satisfies $d(kS, S) = kd(S, S')$ and intersects $\Re$ at two points.

3.1 Description of the Algorithm

After activation with $r.color$ = `OFF`, the target of r is to reach one of the two longest sides of $\Re$ and waits till all other interior robots, all the visible corner robots and the robots on the two shortest sides with color `OFF` reach on one of the two longest sides of $\Re$. Before moving to one of the longest sides of $\Re$, r needs to choose a side of $\Re$ as its target. Let us denote this target side by S_r. If r is already situated on one of the two longest sides of $\Re$, then it selects that side as S_r. If r is a corner robot, r chooses the longest side incident with it as S_r. If r is an interior robot or a boundary robot on one of the two shortest sides, it chooses the nearest longest side as S_r. In case of r being equidistant from both the longest sides, r chooses any one of them as S_r.

Moving on Longest Sides: When r gets activated with $r.color$ = `OFF`, r finds the target side S_r in the current LCM cycle and calculates the point of intersection p_r of S_r and L_r. Now, r needs to consider its current position in the region provided the point p_r is visible to r. If the point p_r is not visible to r, it maintains the status quo. When p_r is visible, we can differentiate the following four cases based on the different positions of r and propose different strategies

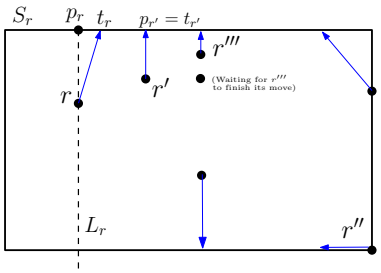

Fig. 2: r moves to t_r on the nearest longest side S_r

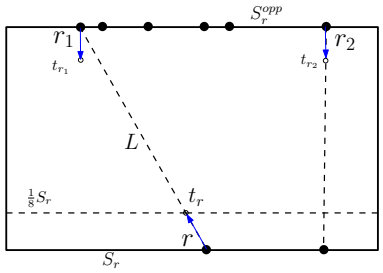

Fig. 3: The movement of r to its apex point

for r. (i) If r is an interior robot with p_r empty, r simply moves to p_r with its current color `OFF`. (ii) If r is a boundary robot on one of the two shortest sides of $\Re$ and p_r (which, in this case, is one of the corners of $\Re$) has a robot on it, r waits with no change in its current color till p_r gets empty. (iii) r is a boundary robot on one of the two shortest sides of $\Re$ and p_r is empty. (iv) r is either an interior robot with another robot on p_r or a corner robot. In the last two cases (iii) and (iv), r needs to find a target point t_r on S_r in such a way that it does not encounter any collision. In this process, r finds a set $\mathcal{V}_r$ that consists of all visible robots not lying on L_r. There can be two sub-cases.

- $\mathcal{V}_r$ **is empty:** In this case, r selects a target point t_r on S_r such that $d(p_r, t_r) = \frac{1}{2}\max\{d(e^1_{S_r}, p_r), d(e^2_{S_r}, p_r)\}$.
- $\mathcal{V}_r$ **is non-empty:** As shown in Fig. 2, r in this case, chooses the target point t_r that satisfies $d(p_r, t_r) = \frac{1}{4}\min_{r' \in \mathcal{V}_r}\{d(r', L_r)\}$.

Finally, r moves to the point t_r without changing the current color `OFF`. If r is already positioned on one of the longest sides of $\Re$, it does not change its current position or color until all the `OFF`-colored robots not lying on the longest sides reach on their nearest longest side.

Now, we need the following definitions to categorize the robots.

Definition 1. *(Terminal Robot): Let L be a line segment with two endpoints e^1_L and e^2_L. A non-corner robot r lying on L is called a terminal robot on L if at least one of line segments $\overline{re^1_L}$ and $\overline{re^2_L}$ is not occupied by any other robot.*

Definition 2. *(Monitor Robot): Let S_r be the longest side of $\Re$ where r is situated in the current LCM cycle. r is called a monitor robot if it satisfies all the following conditions. (i) r is a terminal robot on S_r. (ii) $Int(\Re)$ has no robot. (iii) There is no corner robot visible to r. (iv) There is no robot on both the shortest sides of $\Re$.*

After all the visible robots lying on either S_r or S_r^{opp}, the robot r finds whether it is a monitor robot or not. The strategy is to move the monitor robots at a particular distance from the longest sides of $\Re$ so that they can count the

number of total robots present in the region and decide the type of partition of $\Re$. This is important because neither the robots possess any memory to store the value of N nor they are transparent to see each other. If r does not qualify to be a monitor robot on S_r, it does nothing. When r is a monitor robot on S_r, it needs to move to a point t_r inside $\Re$, referred as *apex point*, from where it can count the number of robots which is necessary to decide the next action of r. The robot r moves to its apex point with color `MONITOR`. At this stage, our target is to gather all the robots on one of the longest sides of $\Re$, if the number of robots lying on the two longest sides is not the same. In this case, our objective is to relocate all the robots from the side with a smaller number to the side with a larger number of robots.

On the other hand, if the number of robots on both of the longest sides are same, we adopt a different strategy to partition the region. Here, the monitor robots first place themselves to their respective final positions in $\Re$ and change their color to `FINISH`. Other robots decide their positions based on these `FINISH`-colored robots. We can identify two cases for monitor robots on S_r.

Case 1 (S_r^{opp} has robots on it): r first moves to the apex point t_r. The calculation of the apex point depends on the positions of the robots on S_r^{opp} so as to confirm that r does not become an obstruction between the other two robots after its movement and all the robots on S_r lie on one half plane delimited by L_r after the movement. Let r_1 and r_2 be the two terminal robots on S_r^{opp}, as shown in Fig. 3. r selects the line $\overleftrightarrow{rr_1}$ as L if the one of the half-plane delimited by the line $\overleftrightarrow{rr_1}$ does not contain any robot. Otherwise, it selects $L = \overleftrightarrow{rr_2}$. r then chooses the point t_r on L such that $d(t_r, S_r) = \frac{1}{8}d(S_r, S_r^{opp})$, if the intersecting point of L_r and $\frac{1}{8}S_r$ lies on the same half plane delimited by L where other robots lies. Otherwise, t_r is the intersecting point of L_r and $\frac{1}{8}S_r$. Finally r moves to t_r after changing its color to `MONITOR` from the color `OFF`.

When r gets activated with the color `MONITOR`, it first identifies the side S_r where it was situated in its previous LCM cycle. To identify S_r, r considers the nearest longest side of $\Re$ as S_r. Observe that there can be two monitor robots on S_r at once and the `MONITOR`-colored robots that are nearest to S_r are visible within the two line segments S_r and $\frac{1}{8}S_r$. So, r can identify the other `MONITOR`-colored robot, if exists (coming from S_r) within this band. Moreover, it can also identify the robots with color `FINISH` on $\frac{1}{4}S_r$ (if any). Similarly, r can easily identify the robots with color `MONITOR` which are nearest to S_r^{opp}. Now, it calculates c_1 and c_2 where c_1 (and c_2) is the number of the robots on S_r (S_r^{opp}) including the robots within the band S_r and $\frac{1}{4}S_r$ (S_r^{opp} and $\frac{1}{4}S_r^{opp}$) with color `MONITOR` or `FINISH`. We have three sub-cases for `MONITOR`-colored robots.

- $c_1 < c_2$**:** It means that there are less robots around S_r than S_r^{opp}. In this case, r needs to move to S_r^{opp}. If the point of intersection p_r^{opp} of L_r and S_r^{opp} is visible to r and contains no robot, r moves to p_r^{opp} with $r.color =$ `OFF` which is shown in Fig. 4. Otherwise, it chooses a target point t_r on S_r^{opp} such that $d(p_r^{opp}, t_r) = \frac{1}{4}\min_{r' \in \mathcal{V}_r} d(r', L_r)$, where $\mathcal{V}_r$ is the set of all visible robots not lying on the line L_r. Then, r changes its color to `OFF` and moves to t_r.

- $c_1 > c_2$**:** r moves back to a point on S_r with color `OFF` by following the same strategy as the above case (instead of S_r^{opp}, r considers S_r). The movement of the robots r_1 and r_2 depicted in Fig. 4 illustrates this case.
- $c_1 = c_2$**:** In this case, r first chooses the shortest side SS_r adjacent to S_r such that there is no robots lying between the point p_r and the common end point of S_r and SS_r. Now, it computes a target point t_r on $\frac{1}{4}S_r$ such that $d(t_r, SS_r) = \frac{len(S_r)}{2c_1}$ and changes it current color to `MOVE`, as shown in Fig. 5. Finally, r moves to t_r from its current position.

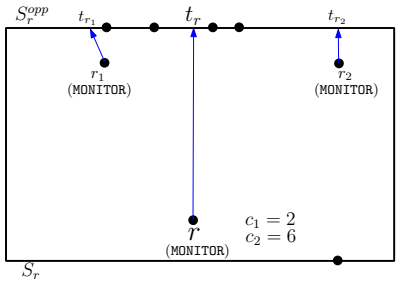

Fig. 4: r moves to t_r from its apex point since S_r^{opp} has more robots than the side S_r

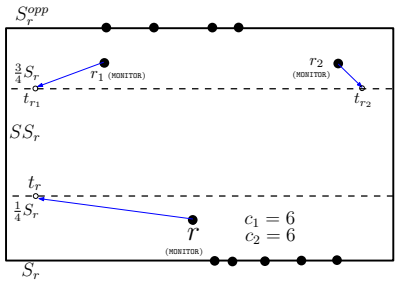

Fig. 5: r moves to t_r on $\frac{1}{4}S_r$ because of same number of robot around S_r and S_r^{opp}

Case 2 (S_r^{opp} **has no robots):** In this situation, r needs to move to an apex point to count the number of robots on S_r. Here, r calculates the apex point t_r on $\frac{1}{8}S_r$ satisfying $d(r, t_r) = \frac{1}{8}d(S_r, S_r^{opp})$ (shown in Fig. 6) and moves to it after changing the color to `MONITOR` from `OFF`. When r gets activated with color `MONITOR` and finds that S_r^{opp} does not have a robot on it, it first identifies the nearest longest side as S_r and finds c, the number of robots within $\Re$. Now, r chooses the shortest side SS_r adjacent to S_r such that no robot is lying between the point p_r and the common end point of S_r and SS_r. Finally, it calculates the target point t_r on $\frac{1}{2}S_r$ such that $d(t_r, SS_r) = \frac{len(S_r)}{2c}$. r changes the color to `MOVE` from `MONITOR` and moves to t_r, as shown in Fig. 7 and 8.

Finally, when r gets activated with color `MOVE`, it changes its color to `FINISH`.

The positions of the `FINISH`-colored robots help other robots with color `OFF` to decide their final position inside $\Re$. At this point, an `OFF`-colored robot r lies on the side S_r. If r is a non-terminal robot on S_r, r waits till it becomes terminal on S_r and does not see any `MONITOR` or `MOVE`-colored robot. When terminal, r checks whether all the robots with color `FINISH` lie on either $\frac{1}{4}S_r$, $\frac{3}{4}S_r$ or $\frac{1}{2}S_r$. r first identifies the sides SS_r and SS_r^{opp} (as described in Case 2) and verifies whether there is a `FINISH`-colored robot on $\frac{1}{2}S_r$ or not. If yes, r sets $L = \frac{1}{2}S_r$. Otherwise, it sets $L = \frac{1}{4}S_r$.

r considers a set $\mathcal{F}_r$, that consists of all the `FINISH`-colored robots. It now finds $d = \min_{r' \in \mathcal{F}_r} \{\min\{d(r', SS_r), d(r', SS_r^{opp})\}\}$. Let $r_1, r_2, ...r_k$ be a sequence of

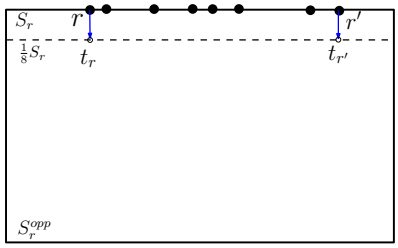

Fig. 6: r moves to its apex point t_r on $\frac{1}{8}S_r$ when S_r^{opp} has no robot

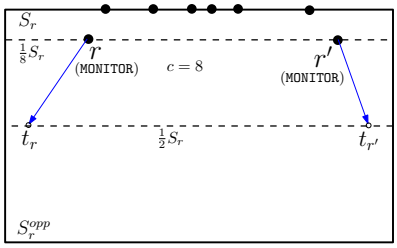

Fig. 7: r moves to $\frac{1}{2}S_r$ from apex point when no robots on opposite side

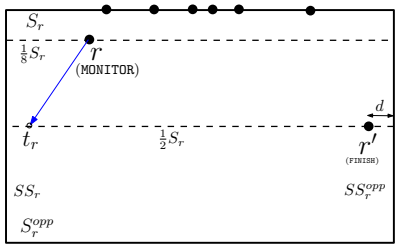

Fig. 8: Movement of r with color MONITOR in presence of FINISH- colored robot

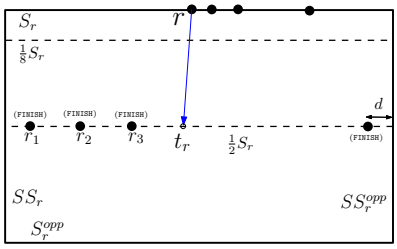

Fig. 9: The movement of a OFF-colored robot in presence of FINISH-colored robots

robots of maximum length on L starting from the robot r_1 with $d(r_1, SS_r) = d$ till the robot r_k such that two consecutive robots in the sequence are exactly $2d$ distance apart from each other, as shown in Fig. 9. Finally, r moves to t_r on L such that $d(t_r, SS_r) = (2k+1)d$ with color MOVE from OFF. It changes its color to FINISH after getting activated with color MOVE.

3.2 Analysis of the Algorithm

Here, we analyse the proposed algorithm when $\Re$ is a rectangular region. We prove the correctness and the time complexity of the algorithm. We also show that the movement of the robots is free from collision.

Lemma 1. *All the robots not lying on one of the two longest sides of $\Re$, move to one of the longest side of $\Re$ without any collision.*

Proof. Let the rectangle $ABCD$ be the region $\Re$ and $\overline{EF}$ divide the region into two equal halves, as shown in Fig. 10. The robots lying inside the rectangle $ABFE$ chooses the side $\overline{AB}$ as their target and move to some points on it. Similarly, the robots inside the rectangle $CDEF$ move to some point of the side $\overline{CD}$. The robots on $\overline{EF}$ chooses any of the two sides $\overline{AB}$ and $\overline{CD}$ as target side and moves to it. Let us consider a robot r on $\overline{EF}$ that chooses the side $\overline{AB}$

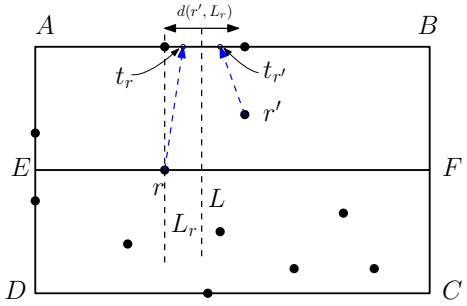

Fig. 10: Collision free movement of the robots r and r'

as target side, but there are robots on the line segment $\overline{rp_r}$ which are nearer to $\overline{AB}$. In this case, r waits till all these robots move to some point on $\overline{AB}$. So, eventually r gets the chance to move to a point on $\overline{AB}$. Now we choose another robot r' lying inside $ABFE$. For both r and r', if the two points p_r and $p_{r'}$ remain empty, they follow the path $\overline{rp_r}$ and $\overline{r'p_{r'}}$ to move to $\overline{AB}$. Since, $\overline{rp_r}||\overline{r'p_{r'}}$, r and r' cannot collide in this movement. Let us assume that both p_r and $p_{r'}$ are non-empty and r' is the nearest robot in $\mathcal{V}_r$. So, r calculates a point t_r on $\overline{AB}$ such that $d(p_r, t_r) = \frac{1}{4}d(r', L_r) < \frac{1}{2}d(r', L_r)$. Even if r' calculates its target point $t_{r'}$ on the line segment $\overline{AB}$, we have $d(p_{r'}, t_{r'}) \leq \frac{1}{4}d(r', L_r)$. Since, the two points t_r and $t_{r'}$ are separated by the line L that passes through the midpoint of $\overline{p_r p_{r'}}$, the two robots r and r' cannot collide. In case of r and r' lying on a same line, they move sequentially to the side $\overline{AB}$, because of which they cannot meet a collision.

Lemma 2. *A* `MONITOR`*-colored robot r sees all other robots in $\Re$.*

Proof. Observe that when S_r and S_r^{opp} both have robots on them, there can be at most four `MONITOR`-colored robots at any time, two from S_r and the other two from S_r^{opp}. Let r_1 and r_2 be the two other robots with color `MONITOR`, as shown in Fig 4. It means all of these three robots got activated and execute their LCM cycles in sync. Since, Let r^{prev} be the position of the robot r in the previous LCM cycle before the movement. Notice that both the half-planes delimited by the line $\overleftrightarrow{r^{prev}r_2^{prev}}$ contains some robots in $\Re$, as shown in Fig. 3. So, r chooses the robot r_1 at the time of calculating its apex point before their movement in the previous LCM cycle, so that the apex point for r lies on the line segment $\overline{r^{prev}r_1^{prev}}$. Since, the line segment $\overline{r^{prev}r_1^{prev}}$ does not have any robots other than r and r_1, the movement of these two robots cannot create an obstruction for each other. Let r' be the terminal robot on S_r other than r^{prev}. Now, the movement of the robot r_2 cannot create any obstruction for r because the apex point for r_2 lies on the line segment $\overline{r'r_2^{prev}}$ and the rest of the robots on S_r^{opp} lie on a different half plane delimited by the line $\overleftrightarrow{r^{prev}r_2^{prev}}$. Additionally, r can see all the robots on S_r, as S_r lies below it and no other robot lies between r and S_r. No other robot would be eligible for movement till all the `MONITOR`-colored

robots move to their target position and change their color to FINISH or OFF. When S_r^{opp} does not have any robot, a similar argument proves that r, being a MONITOR-colored robot, can see all other robots in $\Re$.

Lemma 3. *If one of the longest side LS_1 of $\Re$ contains less number of robots than the other longest side LS_2, all the robots on LS_1 move to LS_2 without collision.*

Proof. Without loss of generality, let LS_1 has less number of robots than LS_2 and r be a robot on LS_1. For r, we have $S_r = LS_1$. When r is a monitor robot on S_r, it first moves to its apex point to count the number of robots on S_r and S_r^{opp}, which is guaranteed by Lemma 2. When it sees that S_r^{opp} has more robots, it finds p_r^{opp} and check if the point is visible or not. If not, it waits till it is visible. When it is visible, r moves to p_r^{opp}, if it is empty. If not, r finds a point t_r on S_r^{opp} such that $d(t_r, p_r) = 1/4 \min_{r' \in \mathcal{V}_r} d(r', L_r)$ and moves to the point t_r. Lemma 1 guarantees that this type of movement is collision free. Thus every robot after becoming monitor a robot on LS_1, move to some point on LS_2.

Lemma 4. *If a MONITOR-colored robot r changes its color to FINISH, it moves to a unique partition where no other FINISH-colored robot resides.*

Proof. Before the movement, there can be two cases for r. (Case i) One of the longest sides (let us consider LS_1) contains no robots. In this case, there can be at most one other MONITOR-colored robot r' on $\frac{1}{8}LS_1$. r and r' chooses SS_r and $SS_{r'}$ which are two different shortest sides of $\Re$. r chooses its destination point at $\frac{len(LS_1)}{2c}$ distance away from SS_r on $\frac{1}{2}LS_1$. Similarly r' chooses its destination point at $\frac{len(LS_1)}{2c}$ distance away from $SS_{r'}$ on $\frac{1}{2}LS_1$, where c is the number of robots in the region. These two destination points are $len(LS1) - \frac{len(LS_1)}{c}$ distance apart from each other and the movement is free from any collision. So r and r' both move to separate partitions of area $len(SS_r) \times \frac{len(LS_1)}{c}$. (Case ii) Both of the longest sides contain same number of robots on them. In this case, the partitions will be of Type II. By similar argument, it can be proved that two MONITOR-colored robots reach different partitions of area $\frac{len(SS_r)}{2} \times \frac{len(LS_1)}{c}$.

Lemma 5. *If a OFF-colored robot r changes its color to FINISH, it moves to a unique partition where no other FINISH-colored robot resides.*

Proof. Similar as previous, there can be two cases. (Case i) One of the side LS_1 has all the robots. In this case, r first chooses a side SS_r to find the number k and computes its destination point on $\frac{1}{2}LS_1$ such that $d(t_r, SS_r) = (2k+1)d$, as described in the algorithm. If there is any other OFF-colored robot on S_r which is simultaneously executing its movement with r, we must observe that SS_r and $SS_{r'}$ is different which guarantees that the destination points for the two robots r and r' are different. Hence, they two finish at two different partitions. (Case ii) Both of the longest sides have same robots. By similar argument as above, it can be proved that a OFF-colored finishes at a unique partition.

Theorem 1. *Our algorithm solves uniform partitioning for the rectangular region in $O(N)$ epochs.*

Proof. Lemma 4 and 5 ensure that every robot chooses a unique partition in the rectangle.

In the worst case, initially, all robots lie on a line inside $\Re$. It takes $O(N)$ epochs for all the robots to reach the boundary, as the movement of the robots will be sequential. Monitor robots on the sides of $\Re$ move to their apex point in one epoch. Moreover, if one of the longest side of $\Re$ contains more robots, the robots from the side S having less number of robots move to the other longest side. This process takes $O(N)$ epochs, as the robots sequentially execute this move. In case of all robots lying on one longest side or robots are distributed equally among two longest sides of $\Re$, it takes $O(N)$ epochs to reach to the final destination point for all the robots. So, overall it requires $O(N)$ epochs to achieve uniform partitioning when the region is a rectangle.

4 Algorithm when $\Re$ is a Square

Here, the region $\Re$ is considered to be a square region. Our proposed algorithm uses 7 colors which are described in the Table 1 with their specification.

4.1 Description of the Algorithm

From any initial deployment of the robots inside $\Re$, our target is to move the interior robots to the boundary. A robot r, after its activation with color `OFF`, determines whether it is a corner, boundary or interior robot. In case of r being a corner or an interior robot, it finds a target side S_r and moves to a point on it with maintaining its color `OFF`. Then r waits for other interior robots and the visible corner robots to move to one of the sides.

If r is a corner robot, it chooses any of the incident sides as S_r. In case of r being an interior robot on $\Re$, it selects one of the nearest sides from its current position as S_r. The strategy is the same as the rectangle. Since there is no shortest side in a square, the difference here is when r is a boundary robot in $\Re$, it does not move. In all other cases, a target point t_r is calculated on S_r and moves to it with color `OFF`.

After this, the outline of the strategy for square region has similarities with rectangular region. In rectangle, we ask the monitor robots to move at a particular distance from the longest sides and make them compare the number of robots situated on two longest sides of the region. If the number is not same, we bring the robots from the side with a smaller number to the side with a larger number of robots. In case of square region, we follow similar strategy, but here the robots can be situated on any side of $\Re$. If all the robots are distributed among the two sides S_r and S_r^{opp}, r follows the algorithm described in Section 3. For other cases, the Definition 2 of monitor robots needs a little modification. A robot r on a side S_r, is called a *monitor robot* if the following three conditions are satisfied. (i) r is a terminal robot on S_r. (ii) $Int(\Re)$ contains no robot.

(iii) There is no corner robot visible to r. We define some new notations and conventions to describe the algorithm for square.

- C is the center of the square $\Re$.
- The triangle $\Delta C e_S^1 e_S^2$ is called the *side triangle* of the side S where e_S^1 and e_S^2 are the endpoints of the side S.
- When we say that r lies on the side triangle of S, we mean that r lies either on the side S or inside the side triangle of S, but not on $\overline{Ce_S^1}$ or $\overline{Ce_S^2}$.
- $|S|$ is the number of robots lying on side triangle of a side S.
- CH_r is the local convex hull of all the visible robots to r.
- r_{Nbr} denotes the neighbor of the robot r on CH_r.
- For the robot r, S_r^L and S_r^R are the edges of $\Re$ other than S_r and S_r^{opp}.
- D_r and D_r^{opp} are the two diagonals of $\Re$.
- Max_r represents a set consisting of all the sides that have the maximum number of robots lying on their corresponding side triangle.

Max_r helps r to decide which sides have the maximum number of robots on them. For example, $Max_r = \{S_r, S_r^{opp}\}$ if $|S_r| = |S_r^{opp}| > |S_r^L|,\ |S_r^R|$. Here, the sides S_r and S_r^{opp} have the maximum number of robots. Our aim is to move all the robots from the sides not in Max_r to the sides in Max_r, but it is not always possible. When Max_r contains all the sides of $\Re$, all the four sides have $\frac{N}{4}$ robots. We differentiate the rest of the algorithm into three major cases for a monitor robot r on the side S_r.

Case 1 ($Int(\Re)$ has no robot with color `FINISH1` or `FINISH2`): Computing Max_r is possible when r moves to a point in $Int(\Re)$, referred as *apex point*, to calculate the number of robots on each sides. This movement should not be at a point from where r does not get to see all the robots in $\Re$ or it obstructs the visibility of any other robot. If r qualifies to be a monitor robot and S_r has no other robot, it does not need to move to an apex point, rather it can find out Max_r without any movement and follow the movement strategy for the `MONITOR1`-colored robots, which is described later in this case. We now discuss the movement of a monitor robot r lying on S_r to its apex points where $|S_r| \geq 2$.
Movement of the Monitor Robots: r finds the neighbor r_{Nbr} on the convex hull CH_r which does not lie on the side S_r. r also finds another neighbour r' (if exists) on CH_r other than r_{Nbr} which lies on S_r. It calculates the apex point t_r on $\overline{rr_{Nbr}}$ such that $d(r, t_r) = \frac{1}{2}\min\{d(r, D_r), d(r, D_r^{opp}), d(r, \frac{1}{4}S_r), d(r, r')\}$. The point t_r is chosen on the line segment $\overline{rr_{Nbr}}$ just to ensure that r does not become an obstruction for other monitor robots after its movement. Moreover, it also ensures that r does not cross D_r and D_r^{opp}, so that after the movement to the apex point, r can calculate S_r, the side where it belongs in its previous LCM cycle. We also want r to lie either on or below the line segment $\frac{1}{8}S_r$, as our aim is to terminate r either on $\frac{1}{2}S_r$, $\frac{1}{4}S_r$, $\frac{1}{3}S_r$ or $\frac{1}{6}S_r$. Additionally, we do not cross the line $L_{r'}$ after the movement to ensure that all other robots on S_r must be on one side of the half plane delimited by L_r. Finally, r changes its current color to `MONITOR1` and moves to t_r.

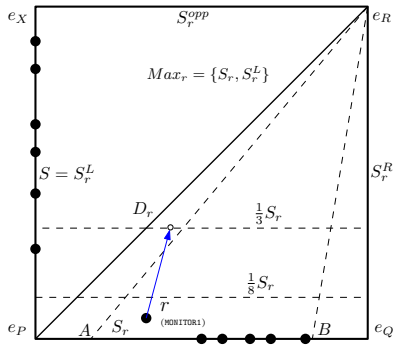

Fig. 11: The movement of r for Type III partitioning when $r.color$ = `MONITOR1`

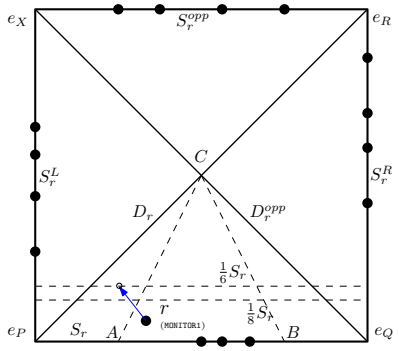

Fig. 12: The movement of r for Type IV partitioning when $r.color$ = `MONITOR1`

Next we explain how a `MONITOR1`-colored robot r decides its destination based on Max_r. It moves from its apex point either to one of the side or to the final point of a partition.

Movement Strategy for the `MONITOR1`-colored Robots: When r gets activated with color `MONITOR1`, it calculates Max_r. This helps the robot r to understand what should be the type of partitioning of the region $\Re$. We identify seven sub-cases, out of which the first two sub-cases discuss the process of choosing the type of partitioning and the movement of r to the final position in its partition. We explain rest of the algorithm using a proper example and figure for better comprehension. We consider a square with corners e_P (common corner of S_r and S_r^L), e_Q (common corner of S_r and S_r^R), e_R (common corner of S_r^{opp} and S_r^R) and e_X (common corner of S_r^{opp} and S_r^L), as depicted in Fig. 11. D_r and D_r^{opp} are the diagonal passing through e_P and e_Q, respectively.

- **Case 1.1 ($Max_r = \{S_r, S\}$ and both the sides S_r^{opp} and S^{opp} contains no robot where $S \in \{S_r^L, S_r^R\}$):** r waits if it sees any robot with color `MONITOR1` lying on the side triangle of S_r^{opp} or S^{opp}. It also waits till all `FINISH1`-colored robot reach $\frac{1}{3}S_r$ or $\frac{1}{3}S$. Let c be number of `FINISH1`-colored robot on $\frac{1}{3}S_r$. Note than $c \leq 1$ in this case. Without loss of generality, let us assume $S = S_r^L$. r calculates two points A and B on S_r which are $\frac{len(S_r)}{|S_r|+c}$ distance away from e_P and e_Q, respectively, as shown in Fig. 11. If S^{opp} lies on the half plane delimited by L_r where other robots of side triangle of S_r lies, r chooses the triangle $\mathcal{T} = \Delta Ae_Pe_R$. Otherwise, it chooses $\mathcal{T} = \Delta Be_Qe_R$. r moves to the centroid of the triangle $\mathcal{T}$ after changing its current color to `FINISH1` from `MONITOR1`.
- **Case 1.2 ($Max_r = \{S_r, S_r^{opp}, S_r^L, S_r^R\}$):** Similar as Case 1.1, r computes the two points A and B as shown in Fig. 12. If e_P and all the other robots on S_r lie on the different half plane delimited by L_r, the triangle $\mathcal{T}$ is chosen that satisfies $\mathcal{T} = \Delta Ae_PC$. Otherwise, it chooses $\mathcal{T} = \Delta Be_QC$. Then r moves

to the centroid of the triangle $\mathcal{T}$ after changing its current color to FINISH2 from MONITOR1.

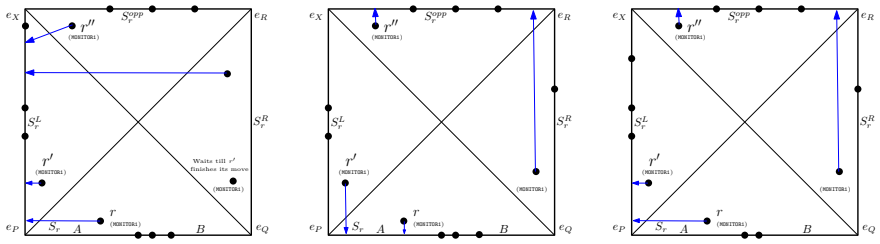

Fig. 13: Illustrating movement of the MONITOR-colored robots to their target side.

For the remaining five sub-cases, r chooses a side before its movement. Our aim is to gather all the robots to one of the four sides of $\Re$. If due to symmetry, it is not possible, robots should be gathered on two sides of $\Re$. When Case 1.1 and Case 1.2 do not hold, the following cases may occur for different Max_r. The robot r decides a side for its movement based on Max_r.

- **Case 1.3** ($Max_r = \{S_r, S_r^{opp}, S\}$ **where** $S \in \{S_r^L, S_r^R\}$)**:** Without loss of generality, if $S = S_r^L$, the side S_r^R has the minimum number of robots. We target to move r to the side S_r^L which is opposite to the side having minimum number of robots. So, r sets the side S as its target side.
- **Case 1.4** ($Max_r = \{S\}$ **or** $\{S, S'\}$**, where** $S, S' \in \{S_r^L, S_r^R\}$)**:** In this case, either one or two adjacent sides to S_r contain maximum number of robots. So r targets to move on one of them accordingly. The target side is chosen to be the side S.
- **Case 1.5** ($Max_r = \{S_r^L, S_r^R, S_r^{opp}\}$ **or** $\{S_r^{opp}\}$)**:** The target side is the side S_r^{opp}.
- **Case 1.6** ($Max_r = \{S, S_r^{opp}\}$**, where** $S \in \{S_r^L, S_r^R\}$))**:** The target side for the robot r is S.
- **Case 1.7** ($Max_r = \{S_r\}$ **or** $\{S, S_r\}$ **or** $\{S_r, S_r^L, S_r^R\}$**, where** $S \in \{S_r^{opp}, S_r^L, S_r^R\}$)**:** In this case, r chooses S_r its target side.

r moves with color OFF to the target side, as shown in Fig. 13. The above cases can interchangeably occur in different LCM cycles of a robot r. If at some point, r finds that all the robots are gathered either on one side or two opposite sides of $\Re$, it follows the strategy described in the Section 3 for rectangular region.

Case 2 ($Int(\Re)$ **has FINISH1-colored robots):** r needs to figure out the type of partitioning by looking at the positions of the robots with color FINISH1.

- $\frac{1}{3}S_r^L$ **has FINISH1-colored robot:** r chooses the side $S = S_r^L$.
- $\frac{1}{3}S_r^R$ **has a FINISH1-colored robot:** r chooses S_r^R as S.

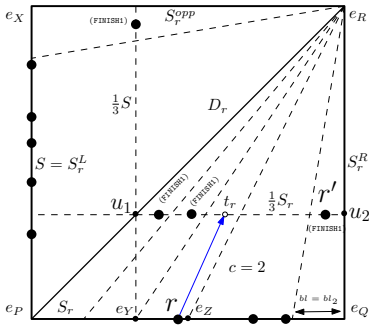

Fig. 14: r's movement for Type III partitioning for $r.color$ = OFF

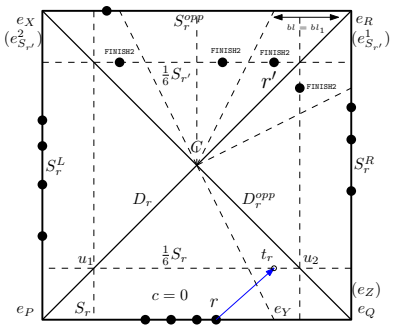

Fig. 15: r's movement for Type IV partitioning for $r.color$ = OFF

– **A robot lying on the side triangle of S_r^L:** r chooses the side S_r^L as S.
– **A robot lying on the side triangle of S_r^R:** The side S_r^R is chosen as S.

The robot r waits till all the FINISH1-colored robots lie either on $\frac{1}{3}S_r$ or on $\frac{1}{3}S$. Without loss of generality, we assume $S = S_r^L$. At this point, r understands that the partitioning is of Type III and the vertex of this partitioning is e_R. Now it computes the length of the base bl of each triangular partition depending on the positions of the FINISH1-colored robots, as shown in Fig. 14. So, there could be two sub-cases.

– **Case 2.1 (There is a FINISH1-colored robot on $\frac{1}{3}S_r$):** Let r' be a terminal robot on $\frac{1}{3}S_r$. Let bl_1 be the length of the base of the triangle whose one side is D_r and the centroid is r'. bl_2 is the length of the base of the triangle whose one side is S^{opp} and the centroid is at r'. So, r computes bl such that $bl = \min\{bl_1, bl_2\}$.
– **Case 2.2 (There is no FINISH1-colored robot on $\frac{1}{3}S_r$):** In this case FINISH1-colored robot (say r') must be on $\frac{1}{3}S$. Similar as the previous case, bl is computed by finding bl_1 and bl_2. We calculate bl_2 by considering the triangle whose one side is S and the centroid is at r'.

Let u_1 be the point of intersection of $\frac{1}{3}S_r$ and the diagonal D_r. u_2 is the point of intersection of $\frac{1}{3}S_r$ and S^{opp}. If the side S^{opp} lies in the half plane delimited by $\overleftrightarrow{re_R}$, where the other robots on S_r reside, r finds the number of FINISH1-colored robots c on $\frac{1}{3}S_r$ starting from the robot $bl/3$ distance apart from u_1 towards u_2 such that two consecutive robots are at $2bl/3$ distance away from each other. Then, it changes its color to FINISH1 and moves to the point t_r on $\frac{1}{3}S_r$ such that $t_r = Centroid(\Delta e_Y e_R e_Z)$, where e_Y and e_Z are the points on S_r satisfying $d(e_Y, e_P) = c \cdot bl$ and $d(e_Z, e_P) = (c+1) \cdot bl$. Otherwise, c is calculated as the number of robots on $\frac{1}{3}S_r$ starting from the robot $bl/3$ distance apart from u_2 towards u_1 to a robot such that two consecutive robots are at $2bl/3$ distance away from each other. r moves to t_r on $\frac{1}{3}S_r$ such that $t_r = Centroid(\Delta e_Y e_R e_Z)$,

where e_Y and e_Z are the points on S_r satisfying $d(e_Y, e_Q) = c \cdot bl$ and $d(e_Z, e_Q) = (c+1) \cdot bl$ with color `FINISH1`.

Case 3 ($Int(\Re)$ has `FINISH2`-colored robots): In this case, r understands that the partitioning would be of Type IV after seeing `FINISH2`-colored robots. r waits till any robot with color `FINISH2` lying on the side triangle of a side S reaches $\frac{1}{6}S$. It now needs to calculate the base length bl of the triangular partition. Let r' be a `FINISH2`-colored terminal robot on $\frac{1}{6}S_{r'}$, in Fig. 15. Let us also consider that bl_1 is the length of the base of the triangle whose centroid is r' and the two vertices are C and $e^1_{S_{r'}}$. bl_2 is the length of the base of the triangle whose centroid is r' and the two vertices are C and $e^2_{S_{r'}}$. Now, r calculates the base length $bl = \min\{bl_1, bl_2\}$. Let us further assume u_1 (and u_2) is the point of intersection of $\frac{1}{6}S_r$ and D_r (and D_r^{opp}). If the line segment $\overline{Ce_Q}$ lies on the half plane delimited by the line $\overleftrightarrow{rC}$, where other robots on S_r reside, r finds the number of `FINISH2`-colored robots c on $\frac{1}{6}S_r$ starting from the robot $bl/3$ distance away from u_1 towards u_2 such that two consecutive robots are $2bl/3$ distance apart from each other. Finally, r changes its color to `FINISH2` and moves to the point t_r such that $t_r = Centroid(\Delta Ce_Y e_Z)$, where e_Y and e_Z are the points on S_r satisfying $d(e_P, e_Y) = c \cdot bl$ and $d(e_P, e_Z) = (c+1) \cdot bl$. Otherwise, if the line segment $\overline{Ce_Q}$ lies on the other half plane delimited by the line $\overleftrightarrow{rC}$, where other robots on S_r reside, r finds the number of `FINISH2`-colored robots c on $\frac{1}{6}S_r$ starting from the robot $bl/3$ distance away from u_2 towards u_1 such that two consecutive robots are $2bl/3$ distance apart from each other. Finally, r changes its color to `FINISH2` and moves to the point t_r such that $t_r = Centroid(\Delta Ce_Z e_Y)$, where e_Z and e_Y are the points on S_r satisfying $d(e_Q, e_Z) = c \cdot bl$ and $d(e_Q, e_Y) = (c+1) \cdot bl$.

4.2 Analysis of the Algorithm

In this subsection, we analyse the above algorithm described for the region $\Re$, when it is a square. The following lemmas and theorems provide the correctness and time complexity of the algorithm.

Lemma 6. *All interior and corner robots move to the boundary.*

Proof. The proof of this lemma follows from Lemma 1.

Lemma 7. *All robots form either a one-side configuration (where all robots gather on one side), or a two-sides configuration (where robots gather on either two adjacent or two opposite sides), or a four-sides configuration (where robots are equally distributed on four sides).*

Proof. After reaching all the robots on the boundary, a robot r will move to the apex point to calculate Max_r. If Max_r has only one element (side), r moves towards the side having maximum robots (follows from the algorithm for the rectangular region and Case 1.4 and Case 1.7 of the square region) which leads to gathering of all robots on one side. If Max_r contains two sides, then r moves

to one of those two sides in Max_r (follows from Case 1.4, 1.6, and 1.7 of the algorithm for square), leading all the robots lying on two sides of $\Re$. If Max_r has three sides, our target is to move r to the side opposite to the side having minimum robots (Case 1.3, 1.5, and 1.7) and eventually Max_r contains exactly one side leading to gathering of all robots on one side. Otherwise, Max_r contains all the sides.

Theorem 2. *All robots terminate in distinct partitions in* $\Re$.

Proof. If all the robots gather at either only one side or two opposite sides, then they will follow the TYPE I or TYPE II partitioning and terminate in distinct partitions (follows from Theorem 1). Let us consider two robots r and r'. If the partitioning is of Type III, it means the robots were distributed among two adjacent sides. Let us consider two robots r and r' lying on same side in that configuration. If there is no `FINISH1`-colored robots, they move to their apex point with color `MONITOR1`. There are two partitions available for these two robots, one is $\Delta e_R e_P A$ and the other is $\Delta e_R e_Q B$, refer Fig. 11. In this situation, one of the two robots r and r' chooses the diagonal while other chooses a side of $\Re$, because of which one of them moves to $\Delta e_R e_P A$ and the other moves to $\Delta e_R e_Q B$. The choice of partitions depends on the choice between the diagonal and the side. If e_R is the vertex of the Type III partitioning, the line $\overleftrightarrow{re_R}$ always separates the diagonal and the other robots of S_r. r' does the same as r. This enable them to choose different partition for their final movement. When r has color `OFF` and lies on S_r, r similarly chooses a triangle $\Delta e_R e_Y e_Z$, as shown in Fig. 14. The similar logic prevents r and r' to choose same partitions. By similar argument as above, r and r' choose different partitions in case of Type IV partitioning.

Theorem 3. *Our algorithm solves uniform partitioning for the square region in* $O(N)$ *epochs.*

Proof. In worst case, it is possible that all robots lie on a straight line on $Int(\Re)$, in which it takes $O(N)$ epochs for all robots to reach the boundary. After moving to the boundary, it takes one epoch for monitor robots to move to apex points and count the number of robots in all four sides. If all four sides have same number of robots, robots lying on a side S of $\Re$, move to their final positions in Type IV partitioning in at most $O(\frac{N}{4}) \approx O(N)$ epochs. If all the robots lie on exactly one side or three sides of $\Re$, the target is to gather the robots in one side, which can be done in at most $O(N)$ epochs. If two sides of $\Re$ have the maximum robots, robots on the remaining sides move to these two sides in $O(N)$ epochs. In this case, it is possible that the robots gathered on these two sides find that one of these two sides having maximum robots. Then, the robots again move from one side to the other which makes the whole process run in $O(N) + O(N) \approx O(N)$ epochs. If the robots need to follow the Type III partitioning, it takes $O(N)$ epochs to move to their final positions. For Type I and II, it takes $O(N)$ epochs that follows from Theorem 1. So, overall it takes $O(N)$ epochs to run the algorithm.

5 Algorithm when $\Re$ is a Circle

In this section, the region $\Re$ is considered to be a circle of radius rad. Similar as earlier, we bring all the interior robots on the boundary of the circular region $\Re$. From there, robots collaboratively move on the boundary in such a way that two consecutive robots become $(2\pi \cdot rad)/N$ arc-length apart from each other. There are 9 colors used in this algorithm which are listed in the Table 1 with their specification.

5.1 Description of the Algorithm

Initially, all robots are with color `OFF`. We define the following notations for this case.
Notations: For a robot r,

- O is the center of $\Re$.
- p_r and p_r^{opp} are the two diametrically opposite points of intersection of the line $\overleftrightarrow{rO}$ and the boundary of $\Re$, out of which p_r is the nearest to r.
- $\widehat{AB}$ represents the segment of the circumference of $\Re$, starting from the point A to B such that the segment does not have any robot except on the endpoints.
- $alen(AB)$ is the arc length of $\widehat{AB}$.
- r^{Nbr_1} and r^{Nbr_2} are the two neighbors of r on the boundary of $\Re$, when r is a boundary robot.

Moving to the boundary of $\Re$: If r is a boundary robot and there is at least one interior robot visible to it, r does not move or change its current color. If r is an interior robot and lies on the center O, it waits till all other interior robots with color `OFF` reach the boundary of $\Re$. When there are no other interior robots left, r does not changes its color and moves to any target point t_r on the boundary that does not contain a robot. Otherwise, r considers the point p_r and checks whether the point p_r is visible or not. If not, r remains in place with no change in color. If p_r is visible and no robot lies on it, r simply moves to p_r with current color `OFF`. In case of p_r is visible and contains a robot, r computes the set $\mathcal{V}_r$, that consists of all visible robots to r not lying on the line $\overleftrightarrow{rO}$. Here, we can have two sub-cases.

- $\mathcal{V}_r$ **is non-empty:** The target point t_r is the point on the boundary such that $alen(t_r p_r) = \frac{1}{4}\min\{alen(p_r p_{r'}) |\ r' \in \mathcal{V}_r$ and $\widehat{p_r p_{r'}}$ is defined$\}$, as shown in Fig. 16.
- $\mathcal{V}_r$ **is empty:** It happens when all the robots lie on one line passing through O. Here, r finds a target point t_r on the boundary of $\Re$ such that $alen(t_r p_r) = \frac{1}{4} alen(p_r p_r^{opp})$, as illustrated in Fig. 17.

Finally, r moves to t_r with the current color `OFF`. Observe that when all robots are on the boundary of $\Re$, they all can see each other, as no three robots

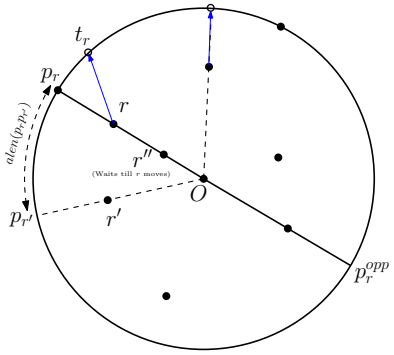

Fig. 16: r's movement to boundary if $\mathcal{V}_r$ is non-empty

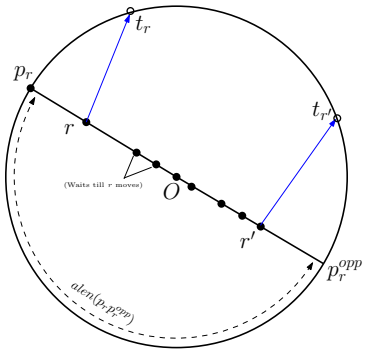

Fig. 17: Movement of r to boundary when $\mathcal{V}_r$ is empty

are collinear. We will turn this to our advantage. Each of the robot can calculate N, the total number of robots in $\Re$. After all robots are at the boundary, the aim is to arrange the robots on the boundary such that every two consecutive robots are $(2\pi \cdot rad)/N$ arc-length apart from each other. We now define a *cluster* that will be useful for the rest of the algorithm.

Definition 3. *(Cluster) A sequence of robots $Cl = \{r_1, r_2, \cdots, r_k\}$ $(k \geq 1)$ lying on the boundary of $\Re$, is called a cluster if the following conditions hold. (i) r_i and r_{i+1} are consecutive robots on the boundary. (ii) $alen(r_i r_{i+1}) = \frac{2\pi \cdot rad}{N}$. (iii) $alen(r_1 r_1^{Nbr}), alen(r_k r_k^{Nbr}) \neq \frac{2\pi \cdot rad}{N}$, where r_1^{Nbr} and r_k^{Nbr} are the neighbors of r_1 and r_k not in Cl.*

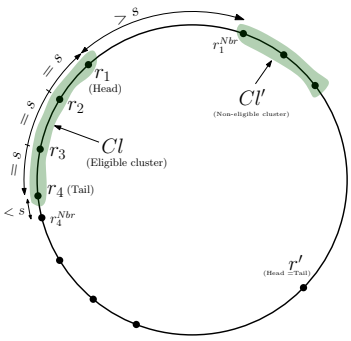

Fig. 18: Illustrating eligible clusters with head and tail

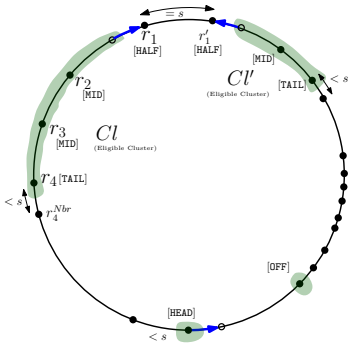

Fig. 19: Movement of the heads of the eligible clusters

For convenience, we denote $(2\pi \cdot rad)/N$ by s. Any robot can detect its own cluster, as it is able to see all the robots on the boundary. A robot r is called the

head of a cluster Cl, if $alen(rr^{Nbr_1}) = s$ for $r^{Nbr_1} \in Cl$ and $alen(rr^{Nbr_2}) > s$. The robot r is called *tail* of the cluster Cl, when $alen(rr^{Nbr_1}) = s$ for $r^{Nbr_1} \in Cl$ and $alen(rr^{Nbr_2}) < s$. It is possible that the head or tail of a cluster does not exist. Moreover, a cluster can be made of a single robot r. In such cases, if $alen(rr^{Nbr_1}) > s$ and $alen(rr^{Nbr_2}) < s$, r itself is called the head and tail of the cluster. An example is given in Fig. 18.

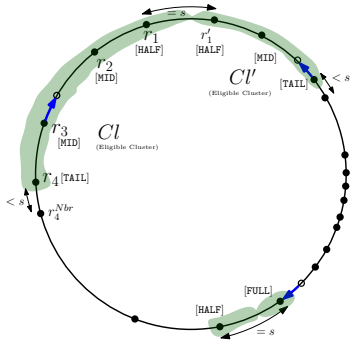

Fig. 20: Movements of the robots with color MID & TAIL

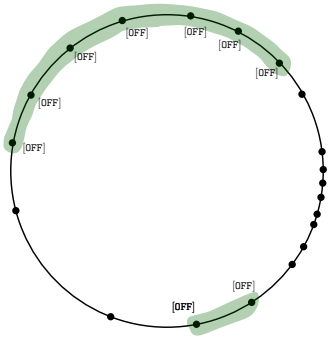

Fig. 21: Two clusters are merged with each other

Definition 4. *(Eligible Cluster) A cluster is eligible for movement if it satisfies the following two conditions.*

1. *The cluster has both head and tail robots.*
2. *All the robots in that cluster are with color* `OFF`*.*

In other words, clusters having only heads or tails is not eligible for a movement. In an eligible cluster, the head initiates the movement along the perimeter of $\Re$. By movement of a cluster $Cl = \{r_1, r_2, \cdots, r_k\}$, we mean that all the robots in Cl moves in the direction of the head sequentially along the perimeter. Note that the robots do not have identifiers. We use $r_1, r_2, \cdots, r_k$ for better comprehension. The movement of a cluster is explained below.

Movement of an Eligible Cluster Cl: Let us consider that r_1 is the head, r_k is the tail of Cl and r'_1 is neighbor of r_1, not in Cl. Let Cl' be the cluster of r'_1, as shown in Fig. 19. When r_1 gets activated with color `OFF` and all other robots in Cl are with color `OFF`, it changes its color to `HEAD` and waits till all robots in its cluster change their color either to `MID` or `TAIL`. r_k sets its color to `TAIL` and all other robots in Cl change their color to `MID` after seeing r_1 with `HEAD`. After this, r_1 determines the eligibility of Cl'. We have two cases.

Case 1 (Cl' is eligible and $r'_1.color$ = `OFF`): In this case, our strategy is to move each heads of Cl and Cl' a distance $\frac{1}{2}(alen(r_1r'_1) - s)$ towards each other, so that the distance between them becomes s. All other robots of the respective clusters sequentially move in the direction of the head's movement. The process

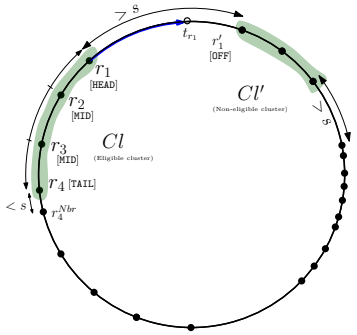

Fig. 22: Cl is an eligible cluster and Cl' is the non-eligible cluster

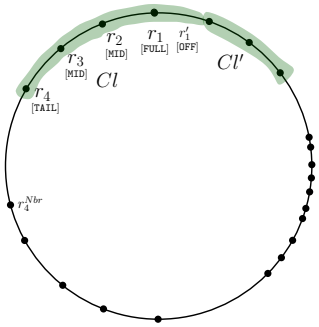

Fig. 23: The two clusters got merged after the movement of Cl

is as follows and illustrated in Fig. 19. The head r_1 changes its color to `MOVE-H` and moves towards r'_1 to a point t_{r_1} such that $alen(t_{r_1}r_1) = \frac{1}{2}(alen(r_1r'_1) - s)$. At this moment, r_k and all `MID`-colored robots do not change their position or color. When r_1 is activated with color `MOVE-H`, it changes its color to `HALF` without any movement. It waits till the tail of the cluster (if exists) r_k sets its color to `OFF`. After seeing the color `HALF` on r_1, r_2 starts moving towards r_1 without changing its current color to a point t_{r_2} on the boundary such that $alen(t_{r_2}r_2) = alen(r_1r_2) - s$. Other robots in Cl except r_2 remain in place at this time. Similarly, for any robot r_i $(3 \leq i \leq k)$, if $alen(r_{i-1}r_i) > s$ and $alen(r_{i-1}r_{i-2}) = s$, r_i moves to t_{r_i} towards r_{i-1} with its current color such that $alen(t_{r_i}r_i) = alen(r_{i-1}r_i) - s$, as shown in Fig. 20. After the movement, r_k finds the cluster Cl with $r_1.color =$ `HALF`. It changes its color to `OFF`. The `MID`-colored robots in Cl change their color to `OFF` after seeing the tail with color `OFF`. The head r_1 changes its current color to `OFF` from `HALF` as illustrated Fig. 21, when both of its neighbor are s arc-length away from it and all the robots in its cluster are with color `OFF`.

Case 2 (Cl' is eligible and $r'_1.color =$ `HALF`) or (Cl' is non-eligible): Here, we move the head r_1 with color `MOVE-F` towards r'_1 to a point on the boundary t_{r_1} such that $alen(t_{r_1}r_1) = (alen(r_1r'_1) - s)$, as shown in Fig. 22. When it is activated with color `MOVE-F`, it changes its color to `FULL`. Rest of the strategy is same as the previous case (Case 1) for other robots in the cluster. Finally, Cl gets merged with Cl', as shown in Fig. 23.

When r finds that all robots are in one cluster i.e., every two consecutive robots on the boundary are at s distance apart from each other, it moves to the point t_r on the line segment $\overline{rO}$ such that $d(r, t_r) = \frac{1}{2}d(r, O)$ with color `FINISH`. Even if a boundary robot r sees a `FINISH`-colored robot in $Int(\Re)$, it follows the same strategy.

5.2 Analysis of the Algorithm

In this subsection, we discuss the correctness and the time complexity of the above-mentioned algorithm for circular region. We also prove that the robots do not meet collision during any movement.

Lemma 8. *Any interior robot in $\Re$ with color* `OFF` *moves to its boundary without collision.*

Proof. Let r be any interior robot with color `OFF`. If r is in the center of $\Re$, it moves to a non-occupied point on the boundary, only when all the other interior robots reach the boundary. This is a trivial collision-free movement. If r lies in the interior of $\Re$ but not in the center, r first calculates p_r to move on its boundary. There can be two cases based on the visibility of p_r. If p_r is visible and does not contain a robot, r moves to p_r. For any other robot r', either $p_{r'}$ is different from p_r or $p_r = p'_r$ but not visible to r'. Therefore, r' can't move to p_r. If p_r is visible, but there is a robot on it, r calculates $\mathcal{V}_r$. If $\mathcal{V}_r$ is empty, then all the robots lie on the line segment $\overline{rO}$ and r is terminal on it. So r moves to a boundary point with $\frac{1}{4}alen(p_r p_r^{opp})$ distance away from p_r, which restricts it from the collision with the other terminal robot on $\overline{rO}$. Otherwise ($\mathcal{V}_r$ is non-empty), r also prevents itself from a collision from its neighbour r' by moving a distance $\frac{1}{4}d(p_r, p_{r'})$ apart from $p_{r'}$. If p_r is not visible, there exists r' on $\overline{rp_r}$ such that p_r is visible to r'. For this case, r' will move to the boundary before any other robot on $\overline{rr'}$, when it activates. After r' moves to the boundary, the robot behind r' (the position before the movement) will be eligible to move on the boundary. So, r will eventually become terminal on $\overline{rO}$ and get a chance to move on the boundary of $\Re$.

Lemma 9. *There always exists an eligible cluster or all the robots are in the same cluster.*

Proof. If all the robots are in the same cluster, the statement of the lemma follows trivially. Let us assume that there are at least two clusters. If all the clusters are non-eligible, then the arc-length between any two clusters is always greater than s, which is a contradiction to the fact that the sum of the distances between two consecutive robots on the boundary is $2\pi \cdot rad = s \cdot N$.

Lemma 10. *After all the robots in a cluster complete their movement, they remain as a cluster.*

Proof. For an eligible cluster $Cl = \{r_1, r_2, \cdots r_k\}$, the head r_1 executes its movement towards another cluster. After that, $alen(r_1 r_2) > s$, since r_2 is a member of Cl. When r_1 reaches its final position and changes its color to either `HALF` or `FULL`, r_2 moves to a point t_{r_2} towards r_1 along the perimeter such that $alen(r_1 t_{r_2}) = s$. All other robots in Cl sequentially execute the same process which leads to the completion of one movement of the whole cluster. Hence the statement follows.

Lemma 11. *Let k ($< N$) be the length of an eligible cluster Cl. The length of Cl gets increased at least by one without collision in $O(k)$ epochs.*

*Proof (**Proof of Lemma 11**).* First the head of an eligible cluster having all robots with color `OFF`, changes its color to `HEAD`. Upon seeing this, all other robots in the cluster change their color to either `MID` or `TAIL`. This operation takes total 2 epochs. For a cluster $Cl = \{r_1, r_2, \cdots r_k\}$ with r_1 as head and r_k as tail, r_1 moves first in the direction of the another cluster. Then `MID`-colored robots move sequentially (first r_2 moves in the direction of r_1, then r_3 and at last the tail r_k). This step takes $O(k)$ epochs. If the cluster Cl moves towards another cluster Cl', the heads of the two clusters become s arc-length apart in just one epoch (either both the heads move towards each other or one head moves towards the other). After the movement of the whole cluster, the tail of that cluster changes its color to `OFF`, then the `MID`-colored robots change their color to `OFF` and finally the head changes its color to `OFF` when both of its neighbors are s arc-length apart from it and all other robots in its cluster are with color `OFF`. This process takes 3 epochs. So, the movement of a cluster of length k takes $O(k)$ epochs. After the movement, both the neighbor of r_1 are s distance apart from each other. Therefore the length of the cluster gets increased at least by one. Hence the proof.

Theorem 4. *Our algorithm solves uniform partitioning for the circular region in $O(N^2)$ epochs.*

Proof. The robots lying on $Int(\Re)$, moves to the boundary. In worst case, it is possible that all the robots lie on a single line passing through the center of $\Re$. The process of moving all the interior robots to the boundary takes $O(N)$ epochs in this case. In the worst case, it is possible that the length of an eligible clusters gets increased only by one in every movement of the cluster. Lemma 11 proves that the movement of an eligible cluster takes $O(k)$ epochs, where k is the length of the cluster. So, it takes $\sum_{k=1}^{N-1} O(k) \approx O(N^2)$ epochs for all robots to become one cluster. Finally, all robots moves to their final positions in one epochs. Thus, our algorithm takes overall $O(N^2)$ epochs to partition the region.

6 Discussion

In this section, we highlight that the above three algorithms do not require the colors used while movement of robots in case of the *semi-synchronous* (SSYNC) activation schedule. In SSYNC setting, a subset of the robots are activated with a fairness assumption that a robot is activated infinitely often. If there are three robots and two of them activating at the same time, execute their respective LCM cycles in sync. Another robot who is not activated yet, can be activated only when the the former two robots complete their cycle. The colors used while a robot is moving, are not required in SSYNC setting, as the robot cannot be seen while moving by other robots under this setting. ASYNC setting does not give that facility because of which a robot in motion can be seen by other robots which might lead to erroneous calculation for the robots. So, the same algorithms proposed in previous sections can also be used in SSYNC setting, but with a lesser number of colors (3 colors for rectangle, 6 colors for square and 7 colors for circle).

7 Conclusion

We studied the distributed version of uniform partitioning of a bounded region using mobile robots. The problem becomes interesting and challenging due to the robot model that is considered in this paper. The robots are opaque and do not have enough memory to store the past information. They have a persistent memory in form of a light. The ASYNC activation schedule which is considered in this paper, is the most general form of activation of a robot network, where any robot can be activated any time. Moreover, the robots do not have any coordinate axes agreement or global orientation. We solved this problem when the region is either a rectangle, a square or a circle. The reason behind this, is the application oriented point of view, as the regions, we deal with in our daily life, are known geometric figures. We believe that the problem can be extended to other convex regions under the same model. Also, the problem can even be studied when some of the robots become faulty.

References

1. Acevedo, J.J., Arrue, B.C., Maza, I., Ollero, A.: A distributed algorithm for area partitioning in grid-shape and vector-shape configurations with multiple aerial robots. Journal of Intelligent & Robotic Systems 84, 543–557 (2016)
2. Bhagat, S., Mukhopadyaya, K.: Fault-tolerant gathering of semi-synchronous robots. In: Proceedings of the 18th International Conference on Distributed Computing and Networking. ICDCN '17, Association for Computing Machinery, New York, NY, USA (2017)
3. Das, D., Mukhopadhyaya, S.: Distributed painting by a swarm of robots with unlimited sensing capabilities and its simulation. ArXiv abs/1311.4952 (2013)
4. Das, D., Mukhopadhyaya, S.: Distributed algorithm for painting by a swarm of randomly deployed robots under limited visibility model. International Journal of Advanced Robotic Systems 15(5), 1729881418804508 (2018)
5. Das, D., Mukhopadhyaya, S., Nandi, D.: Swarm-based painting of an area cluttered with obstacles. International Journal of Parallel, Emergent and Distributed Systems 36(4), 359–379 (2021)
6. Di Luna, G.A., Flocchini, P., Chaudhuri, S.G., Poloni, F., Santoro, N., Viglietta, G.: Mutual visibility by luminous robots without collisions. Information and Computation 254, 392–418 (2017)
7. Flocchini, P., Prencipe, G., Santoro, N., Widmayer, P.: Arbitrary pattern formation by asynchronous, anonymous, oblivious robots. Theoretical Computer Science 407(1), 412–447 (2008)
8. Pavone, M., Arsie, A., Frazzoli, E., Bullo, F.: Distributed algorithms for environment partitioning in mobile robotic networks. IEEE Transactions on Automatic Control 56(8), 1834–1848 (2011)
9. Pramanick, S., Mandal, P.S.: Fault-tolerant mutual visibility for async mobile robots with local coordinate. In: 2023 IEEE Guwahati Subsection Conference (GCON). pp. 1–6 (2023)
10. Pramanick, S., Spourgeon, G.J., Raj, K., Mandal, P.S.: Asynchronous line formation in presence of faulty robots. p. 75–82. NSysS '22, Association for Computing Machinery, New York, NY, USA (2022)

11. Saha, D., Pattanayak, D., Mandal, P.S.: Surveillance of uneven surface with self-organizing unmanned aerial vehicles. IEEE Transactions on Mobile Computing 21(4), 1449–1462 (2022)
12. Sharma, G., Vaidyanathan, R., Trahan, J.L.: Constant-time complete visibility for robots with lights: The asynchronous case. Algorithms 14(2), 56 (2021)