International Journal of Networking and Computing – www.ijnc.org, ISSN 2185-2847
Volume 15, Number 2, pages 199–219, July 2025

Asynchronous Separation of Unconscious Colored Robots

Paola Flocchini
School of Electrical Engineering and Computer Science,
University of Ottawa, Ottawa, Canada

Debasish Pattanayak
School of Electrical Engineering and Computer Science,
University of Ottawa, Ottawa, Canada

Francesco Piselli
Department of Mathematics and Computer Science,
University of Perugia, Perugia, Italy

Nicola Santoro
School of Computer Science, Carleton University,
Ottawa, Canada

Yukiko Yamauchi
Faculty of ISEE, Kyushu University,
Fukuoka, Japan

Received: February 14, 2025
Revised: April 25, 2025
Accepted: June 9, 2025
Communicated by Akihiro Fujiwara

Abstract

We consider the recently introduced model of autonomous computational mobile entities called *unconscious colored robots*. The entities are the traditional oblivious silent mobile robots operating in the Euclidean plane in Look-Compute-Move cycles. However, each robot has a permanent external mark (or color) from a finite set, visible by the other robots, but not by the robot itself. The basic problem for these robots is *separation*, requiring all the robots with the same color to separate from the other robots, each group forming a recognizable geometric shape (e.g., circle, point, line); this task must be performed in finite time, in spite of the robots being unconscious of their own color, unable to communicate, and oblivious. This problem has been studied and solved in the synchronous setting (SSS 2023). In this paper we show that the problem is solvable also under the more difficult asynchronous adversary, provided the robots agree on the orientation of one axis, and no robot is uniquely colored. The proof is constructive: we present a distributed algorithm that allows unconscious colored robots with one-axis agreement to separate into parallel lines under the asynchronous scheduler.

Keywords: Mobile robots, Separation, Externally visible colors, Line Formation.

1 Introduction

1.1 Framework

In distributed computing, the theoretical interest in multi-robot systems and robotic swarms has focused on computational and complexity issues arising in systems of autonomous mobile computational entities. Several models, emphasizing different computational aspects, have been considered, including *population protocol* models (e.g., [1, 2, 3]), *metamorphic robotic system* models (e.g.,[4, 5, 6]), and the de facto standard, *oblivious silent mobile robots* model (e.g.,[7, 8, 9]), which we consider here.

In the basic version of this last model, called $\mathcal{OBLOT}$, the robots are simple mobile computational entities operating in `Look-Compute-Move` cycles in the Euclidean plane, where they are viewed as points. In each cycle, an active entity observes the position of other robots in its local coordinate system (`Look`), determines a destination (`Compute`), and moves toward it (`Move`). Movements are *rigid* if they always reach their destination, and *non-rigid* if guaranteed only to traverse at least a distance δ (unknown to the robots). The entities are *oblivious* (have no memory of activities performed in previous cycles), *silent* (they have no direct means of communication), and are *homogeneous* (they have no internal identifiers nor external distinguishing features, and execute the same protocol to determine their destination). Furthermore, there might be no agreement between their local coordinate systems nor on clockwise orientation of the plane (called *chirality*).

Regarding time and synchronization of the robots, two settings, *synchronous* and *asynchronous*, are considered, each controlled by an adversary called *scheduler*: in the *synchronous* setting, time is divided into rounds and, in each round, the adversarial scheduler *SSYNC* selects a non-empty subset of the robots to be active and execute their cycle simultaneously; in the *asynchronous* setting, there is no common notion of time, and the decision of when a robot is activated and executes its cycle, as well as the duration (arbitrary but finite) of each operation in that cycle, is made by the adversarial scheduler *ASYNC*. The adversarial schedulers are however fair: they activate each robot infinitely often. The fairness condition allows to measure time in terms of successive sequences of activations of robots, called *epochs*: the first epoch starts with the first activation[] each epoch ends as soon as all robots have been activated; and the successive epoch starts with the first activation after the end of the previous epoch.

An extensive amount of research has been carried out on the computational and complexity aspects of problem solving by robots in the $\mathcal{OBLOT}$ model. The majority of the research has focused on the class of *pattern formation* problems (e.g., [7, 8, 9, 10, 11]), and in particular on the *point formation* (also known as *gathering* or *rendezvous*) problem (e.g., [12, 13, 14, 15]). Complete characterization of formable patterns has been established when the robots agree on *chirality* based on the symmetry of initial and target patterns [9, 11], and, starting from any initial configuration, based on the level of agreement on the local coordinate systems [8].

1.2 Heterogeneity

Each of the defining properties of the $\mathcal{OBLOT}$ model severely restricts the computational capability of the robots. The extent and impact of the restriction is immediately evident for properties such as lack of persistent memory and absence of direct means of communication. Less obvious but equally damaging is homogeneity of the robots; indeed, the combination of anonymity (lack of distinguishing features) and uniformity (executing the same protocol) may render symmetry breaking impossible, which in turn renders many problems unsolvable for these robots.

This has motivated the investigation of systems where the robots, still oblivious and silent, are however *heterogeneous*, and different models have been examined: each robot has a distinct identifier (externally visible or invisible) [10]; every robot is assigned an identifier from a small set (hence there are replications) which are visible by all [16]; the identifiers from a small set are externally invisible [17, 18, 19].

In this line of research, a particularly interesting theoretical model is that of *unconscious colored robots* recently introduced by Seike and Yamauchi [20]. This model, which we shall denote $\mathcal{UCR}$, extends the computational capabilities of $\mathcal{OBLOT}$ by endowing each robot with an identifier, called color, from a small set, which is visible by the other robots but unknown to the robot (hence the

term "unconscious"); more precisely, in the `Look` operation, a robot can see the colors of all the other robots, but not its own. This model in practice describes a heterogeneous system of non-identical robots lacking self-awareness; this means that, unlike the other heterogeneous models, $\mathcal{UCR}$ has the unique property that, even if the robots have distinguishing features, they are still uniform in the sense that they execute the same algorithm.

Further observe that the model of each entity having an input value, visible to all but not to itself, is a central model in the field of Communication Complexity; the model, called *Number-on-Forehead* (NOF), was introduced by Chandra, Furst, and Lipton [21], and is known to have connections to circuit lower bounds and to additive combinatorics (e.g., [22, 23, 24]).

In this paper, we continue the investigation started in [20] by examining and solving a basic problem described next.

1.3 The Separation Problem

The problem we consider is that of *separation*. At an abstract level, this is the basic problem in a multi-robot system of having the robots to form groups based on certain identifiable property.

In addition to its inherent theoretical interest, this problem is also of practical interest. For example, a solution to the separation problem can act as an intermediate procedure to determine faults in the absence of internal diagnosis. There are several works [25, 26, 27] on detection of faults by diagnosis by other robots in the system. Another application is partitioning of tasks [28, 29] among a group of robots, where separation can be used as a precursor.

In the $\mathcal{UCR}$ model, the *separation* problem has a natural definition as the requirement that all the robots with the same color separate from the other robots, each group forming a recognizable geometric shape (e.g., point, circle, line); this task must be performed in finite time, despite the robots being unconscious of their own color, unable to communicate, disoriented and oblivious.

This version of the problem has been introduced and studied in [20]. They first showed that separation into points is not generally feasible, as there are initial configurations (those symmetric with respect to the positions and colors of the robots) where the robots cannot separate into points irrespective of the adversarial synchronization scheduler.

Then, they showed that, under the synchronous adversarial scheduler, *SSYNC*, the robots can separate into concentric circles, starting from any arbitrary initial configuration with no uniquely colored robots (i.e., where there are at least two robots for each color); they are able to do so assuming that the robots agree on chirality (no other assumption on the coordinate system) and that movements are rigid. That is, the *separation* is indeed solvable under *SSYNC*. This leaves open whether separation is possible under the more powerful adversarial scheduler *ASYNC*, and under what conditions. In this paper we provide definite answers to these questions.

1.4 Contributions

In this paper, we prove that it is possible for the robots to separate under *ASYNC* starting from any initial configuration with no uniquely colored robots; they are able to do so under a weak assumption on the level of agreement among the local coordinate systems, and with non-rigid movements. Furthermore, we prove that, without any such agreement, the problem becomes unsolvable.

More precisely, we consider robots that agree on the direction and orientation of just one axis. We show how, starting from any arbitrary initial configuration with no uniquely colored robots, under the *ASYNC* scheduler, they can separate into parallel lines, one for each color. We present special solutions for $n < 5$ robots, and a general solution for $n \geq 5$.

We also analyze the complexity of the proposed algorithm and show that its execution time is $O(n\lceil(h+w)/\delta\rceil)$ epochs under the *ASYNC* scheduler, where n is the total number of the robots, h and w are the height and width of the smallest enclosing rectangle of the initial configuration, respectively, and δ is the minimum distance a robot moves in one activation.

We conclude by proving that, without any form of agreement on direction and orientation of at least one axis, the separation into lines problem is not always solvable, even if the robots agree on chirality (that is, they agree on the clockwise and counterclockwise directions).

2 Model

We consider a set of n punctiform mobile robots $\mathcal{R} = \{r_1, r_2, \ldots, r_n\}$ located and operating in the Euclidean plane $\mathbb{R}^2$. Let $p_i(t)$ denote the position of robot r_i at time t, expressed in a global coordinate system (unknown to the robots). Associated with each robot r_i is an identifier c_i, called color, from a totally ordered set $\mathcal{H}$ known by all robots. Let $C \subseteq \mathcal{H}$ be the subset of colors for which there exist robots holding those colors. Let $|C| = \kappa$ be the size of C. The robot configuration at time t is a multiset $P(t) = \{(p_1(t), c_1), \ldots, (p_n(t), c_n)\}$. We assume that all robots are initially located in distinct locations and that each color in C is held by more than one robot.

The robots are *oblivious*, i.e., they do not retain memory of past computations; *anonymous*, i.e., they do not possess identifiers; *silent*, i.e., they do not communicate explicitly; and *uniform*, i.e., they execute the same algorithm. When a robot is activated, it performs a `Look-Compute-Move` cycle, composed of three phases.

In the `Look` phase, a robot obtains a snapshot of the positions of all the other robots, expressed in its local coordinate system; the snapshot contains also the colors; however, it cannot perceive its own color. We shall denote the snapshot obtained by r_i at time t by $Z_i(t)$. The snapshot is egocentric, that it considers the position of r_i as the origin; however chirality and unit distance might differ from one robot to another.

In the `Compute` phase, using the obtained snapshot as input, it executes a deterministic algorithm ψ (the same for all robots) to compute a destination.

In the `Move` phase, the robot moves toward the computed destination with *non-rigid* movements: the robot movement can be stopped at any point after traversing a distance δ (unknown to the robots).

The activation of the robots and the duration of the phases of their cycles are under the control of an adversary, called *scheduler*. There are two types of settings, and thus schedulers, *synchronous* and *asynchronous*. In the *synchronous* setting, time is divided into rounds and, in each round, the adversarial scheduler *SSYNC* selects a non-empty subset of the robots; the selected robots become active and execute their cycle simultaneously. In the *asynchronous* setting, there is no common notion of time, and the decision of when a robot is activated and executes its cycle, as well as the duration (arbitrary but finite) of each operation in that cycle, is made by the adversarial scheduler *ASYNC*. In both settings, the scheduler is *fair*, i.e., it activates each robot infinitely often. Time is measured in terms of successive *epochs*, each ending when all robots have completed at least one cycle.

The robots agree on the positive direction of one axis (say the y-axis), but not necessarily on that of the other.

The problem of *separation into lines* asks the following. A robot group located at arbitrary initial distinct positions on the Euclidean plane $\mathbb{R}^2$, must position themselves on parallel lines L_{c_i} for $c_i \in C$, such that each line contains only robots of the same color. In other words, the robots must satisfy the following geometric predicate:

$$\begin{aligned}\texttt{SepL} \equiv \{&\exists t : (\forall t' > t, P(t') = P(t)) \ \textbf{and} \\ &(\forall c_i \in C, \exists L_{c_i} : \forall (p_j(t), c_j) \in P(t), \\ &c_j = c_i \iff p_j(t) \in L_{c_i}) \ \textbf{and} \\ &\big(\forall c_i, c_j \in C, L_{c_i} \parallel L_{c_j} \ \textbf{and} \ L_{c_i} \neq L_{c_j} \ \textbf{if} \ c_i \neq c_j\big)\}\end{aligned}$$

where $L_{c_i} \parallel L_{c_j}$ means that the lines L_{c_i} and L_{c_j} are parallel. Note that, these lines are not determined *a priori*.

3 Separation into Lines: $n \geq 5$

In this section, we present a solution to the separation problem for $n \geq 5$; the case $n \leq 4$ will be addressed in Section 4.

3.1 Terminology

Given a configuration P, where $p_j = (x_j, y_j)$ $(1 \leq j \leq n)$, let $l_x = \min_{1 \leq j \leq n}(x_j)$, $r_x = \max_{1 \leq j \leq n}(x_j)$, $d_y = \min_{1 \leq j \leq n}(y_j)$, and $u_y = \max_{1 \leq j \leq n}(y_j)$. The rectangle with the four corners (l_x, d_y), (l_x, u_y), (r_x, u_y), and (r_x, d_y) (ref. Fig. 1a) shall be called the *Smallest Enclosing Rectangle* of P, and denoted by $SER(P)$. The robots can always agree on the $SER(P)$ since they agree on the direction of one axis and thus obtain a clear ordering on the coordinates along the agreed axis, and simultaneously determine extreme coordinates perpendicular to it.

The rectangle $SER(P)$ is said to be *improper* if it degenerates to a horizontal line segment (i.e., if $u_y = d_y$) or vertical line segment (i.e., if $l_x = r_x$) or a point (i.e., if $u_y = d_y$ and $l_x = r_x$), and *proper* otherwise. For a proper $SER(P)$, we define the height h as $h = |u_y - d_y|$ and the width w as $w = |r_x - l_x|$.

For a proper $SER(P)$, we define two significant lines: L_u is defined as the upper side of the rectangle; i.e., the line segment joining (l_x, u_y) and (r_x, u_y). L_d is defined as the bottom side of the rectangle i.e., the line segment joining (l_x, d_y) and (r_x, d_y).

3.2 Outline of Algorithm SeparateLines for $n \geq 5$

Ideally, the solution algorithm is composed of three stages: *Initialization*, *Signaling*, and *Finalization*.

In the *Initialization* stage, the first objective of the robots is to move so that they all are on distinct locations of the same line, denoted Ξ, perpendicular to the y-axis.

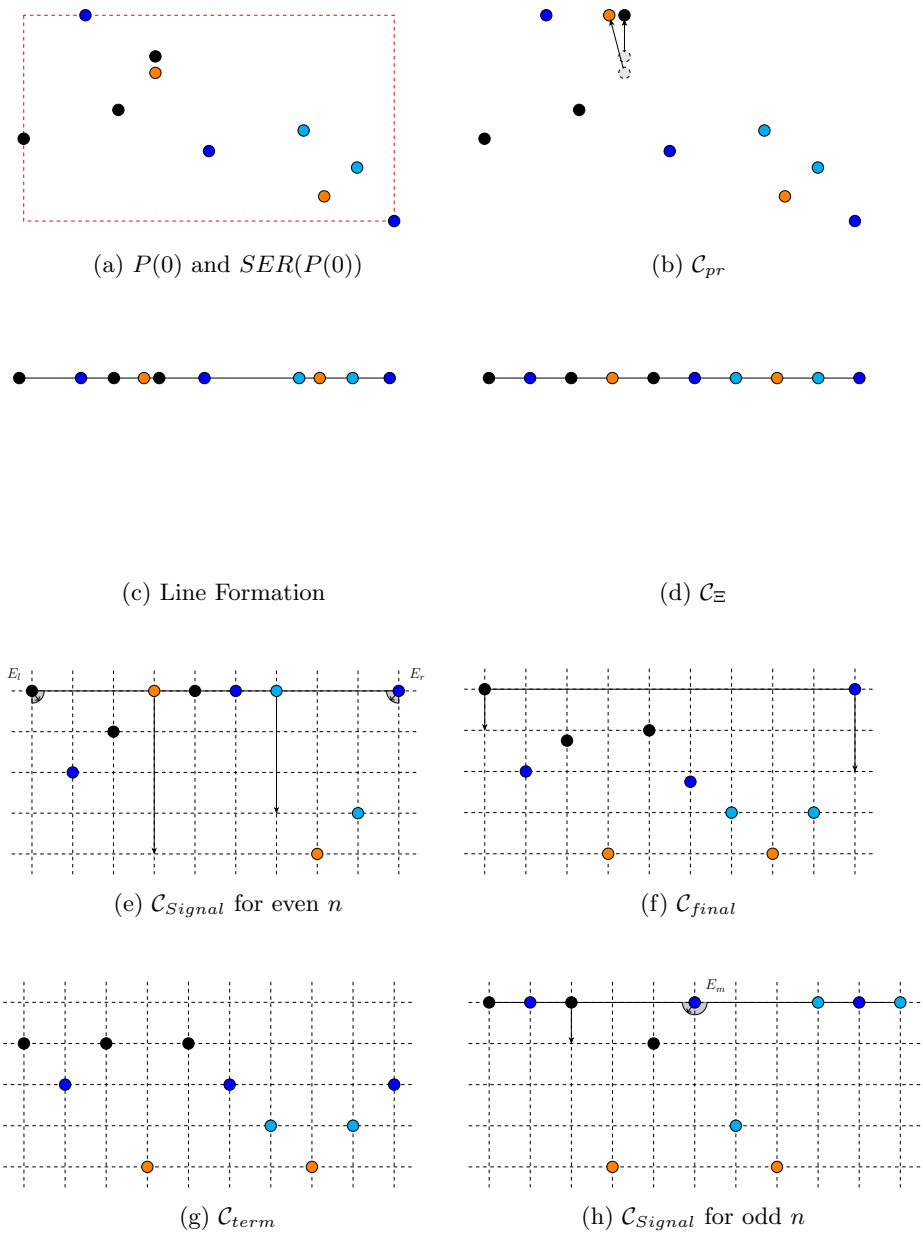
If the SER of the initial configuration is proper, then Ξ coincides with L_u, the top side of the smallest enclosing rectangle (ref. Fig. 1a). In this case, the robots will all move to reach L_u (ref. Fig. 1b). If the SER of the initial configuration is improper because all robots are on the same horizontal line, Ξ is exactly that line. If instead the SER of the initial configuration is improper because all robots are on the same vertical line, Ξ is the horizontal line passing through the position of the topmost robot; note that, in this last case, the second robot in the vertical order of the robots will be made to move horizontally to create a proper SER.

The second and final goal of the robots in this stage is to move to equidistant positions on Ξ in the segment delimited by the two extreme robots (i.e., the two robots at maximum horizontal distance as depicted in Fig. 1c). Once the robots are equidistant on Ξ (ref. Fig. 1d), the following grid G is uniquely defined: the width w of the grid is $w = \alpha(n-1)$, where α is the horizontal distance between two consecutive robots on Ξ; the topmost horizontal points of the grid are precisely those occupied by the robots on Ξ; the height h of the grid is $h = \alpha(\kappa + 1)$, where κ is the size of color set (ref. Fig. 1e). At this point, the robots proceed to the *Signaling* stage of the algorithm.

The *Signaling* stage begins by choosing the leader robots. If the number of robots is odd, there exists a unique median robot on Ξ (robot at E_m in Fig. 1h), and that is naturally chosen as the leader. If the number of robots is even, the two extreme robots on Ξ (robots at E_l and E_r in Fig. 1e) are chosen as leaders; each will be responsible for its closer half of the other robots. The elected leader robots then guide the other robots to reach an empty appropriate grid position in the same column as the robot. The goal is that, at the end of this stage, all non-leader robots are located at grid points such that each row of the grid only contains robots with the same color.

Thus the objective of this stage is to move each non-leader robot to a grid point corresponding to its color. This is achieved by the leader(s) signaling its closest (non-leader) robot by moving to a *signal point* at a predefined distance, chosen to be $\alpha/3$. Any robot can determine the signaled robot and the corresponding destination grid point (of the signaled robot) based on the angle the signal point makes at the closest grid point (the location of the leader on Ξ) with Ξ. Upon observing the leader at a signal point, the signaled robot moves vertically downwards to the grid point indicated by the signal angle. The leader stops signaling and returns to its grid point on Ξ after it observes that the signaled robot has reached the destination grid point. Subsequently, it continues the signaling process for the next closest (non-leader) robot on Ξ. Once all the non-leader robots reach their corresponding grid points, the *Finalization* stage begins.

In the *Finalization* stage, each leader robot moves to the grid point on its column corresponding to the row of its color.



(a) $P(0)$ and $SER(P(0))$ (b) $\mathcal{C}_{pr}$

(c) Line Formation (d) $\mathcal{C}_{\Xi}$

(e) $\mathcal{C}_{Signal}$ for even n (f) $\mathcal{C}_{final}$

(g) $\mathcal{C}_{term}$ (h) $\mathcal{C}_{Signal}$ for odd n

Figure 1: Figures showing a sample execution of Algorithm SeparateLines.

Consider first when the number of robots is odd, and let r_l be the only leader robot. Let r_c be a robot of the same color as the leader. Such a robot r_c always exists, since there are at least two robots with the same color. The robot r_c moves vertically downwards (for a fixed distance, chosen to be $\alpha/4$) to signal r_l, and r_l moves to the grid point corresponding to the grid row where r_c was located. Once r_c observes that r_l has reached the point, it returns to its grid point concluding the separation.

When the number of robots is even, and thus there are two leaders, r_l and r'_l, the procedure is a little bit more involved. If r_l and r'_l are of the same color and no other robot exists with the same color as r_l, then they do not need to move, and consequently do not need to receive signals. Otherwise, each leader needs to be signaled by a robot of the same color.

Let r_c be a robot with the same color as r_l. If r_c and r_l are on the same half of Ξ, then r_c moves

down for a distance $\alpha/4$, otherwise it moves up for the same distance (ref. Fig. 1f). This movement of r_c acts as a signal for r_l. Notice that, r_c can realize its own color by the vertical distance from Ξ. Using this signal r_l moves to the row of r_c, and analogously for r'_l (which reaches the same row of its signaling robot r'_c). Once r_l (r'_l) reaches the corresponding grid row of r_c (r'_c), r_c (r'_c) moves back to its nearest grid point. This concludes the separation into lines (ref. Fig. 1g).

There are some exceptional cases that we need to consider when an initial configuration is similar to a configuration from Signaling stage or Finalization stage, but the robots in the same row do not have the same color. We call these configurations *invalid signal* and *invalid terminal* configurations. In case of an invalid signal configuration, a leader can recognize the invalidity, and it signals *reset* to form Ξ by moving to a special signal point (reserved for reset). Then the signaling stage restarts again from configuration with equidistant robots on Ξ. In case of an invalid terminal configuration, a leader robot may move to a row (due to a wrong signal). Then a robot (other than the leader) realizing the invalidity, moves horizontally to break the grid. This restarts the Initialization stage.

3.3 Classes of Configurations

Let $\mathcal{C}$ be the set of all possible configurations $P(t)$ such that all robots occupy distinct positions. Let $p_i(t) = (x_i, y_i)$ be the position of robot r_i. Define $\mathcal{C}_{pr}$ as the set of all possible configurations such that $x_i \neq x_j$ for any pair of robots r_i and r_j ($i \neq j$). We define $\mathcal{C}_{arb}$ as $\mathcal{C} \setminus \mathcal{C}_{pr}$. For a configuration in $\mathcal{C}_{pr}$, let $(p_1, p_2, \ldots, p_n)$ denote the robots' positions in the increasing order of x-coordinates. Note that two different robots may have different orders, since they do not agree on the positive direction of x-axis (also referred to as the right side). If n is even, we designate the leaders as the two extremes (i.e., robots at p_1 and p_n); if n is odd, the leader is the median robot (i.e., the one at $p_{\lceil \frac{n}{2} \rceil}$).

We define the horizontal distance between two robot positions as $|x_i - x_j|$. Let α be $\frac{|x_2 - x_{n-1}|}{n-3}$. An α-grid on the plane is a set of points where any two points are at a horizontal distance $i\alpha$ and vertical distance $j\alpha$, for $i, j \in \mathbb{N}$. We call the grid lines parallel to the y-axis as columns, and the ones perpendicular as rows.

First, we establish configurations that belong to the class $\mathcal{C}_\alpha$. Intuitively, this class of configurations represents the configurations where each robot is located on some grid point of the α-grid except the leader(s) and at most two additional robots. More precisely, in a configuration in $\mathcal{C}_\alpha$, at most one non-leader robot can occupy a non-grid point on each side of the vertical line of symmetry. However, all non-leader robots in $\mathcal{C}_\alpha$ must remain on the grid columns.

The class of configurations $\mathcal{C}_\Xi$ contains the configurations where all the robots are in the same row of an α-grid; in such a case the row where they are located is called the principal line Ξ.

For a configuration $P(t) \in \mathcal{C}_\alpha$, we define the *leader base* as the point that contains the leader(s): $E_m = ((l_x + r_x)/2, u_y(0))$ when n is odd; $E_l = (x_2 - \alpha, u_y(0))$ and $E_r = (x_{n-1} + \alpha, u_y(0))$ when n is even, where $u_y(0)$ represents the maximum value of y-coordinate in the initial configuration $P(0)$.

Consider $P(t) \in \mathcal{C}_\alpha$. Let L_u be the uppermost row of the α-grid. Let $\tau = \pi/n\kappa$ be the *unit angle*. The *signal circle* is a circle of radius $\alpha/3$ centered at the leader base. A *signal angle* is always an integral multiple of the unit angle. A *signal point* s is a point on the signal circle that makes a signal angle with L_u at the leader base. We define a *signal segment* as the line segment joining a signal point and the leader base, not including these endpoints.

We define $\mathcal{C}_{signal} \subset \mathcal{C}_\alpha$ for odd n as the class of configurations where either

(i) a leader is on the signal segment and all non-leader robots are on α-grid points, *or*

(ii) a leader is at a signal point and at most one non-leader robot is not at an α-grid point.

When n is even, $\mathcal{C}_{signal}$ has at most one non-leader robot on either side of the vertical line of symmetry not on the α-grid.

We define $\mathcal{C}_{final} \subset \mathcal{C}_\alpha$ as the class of configurations where at most the leaders are not on an α-grid point, and there are at most two robots that are vertically at most a distance of $\alpha/4$ away from a grid point of the α-grid.

The class of configurations where all the robots are on the α-grid points is denoted by $\mathcal{C}_{term}$.

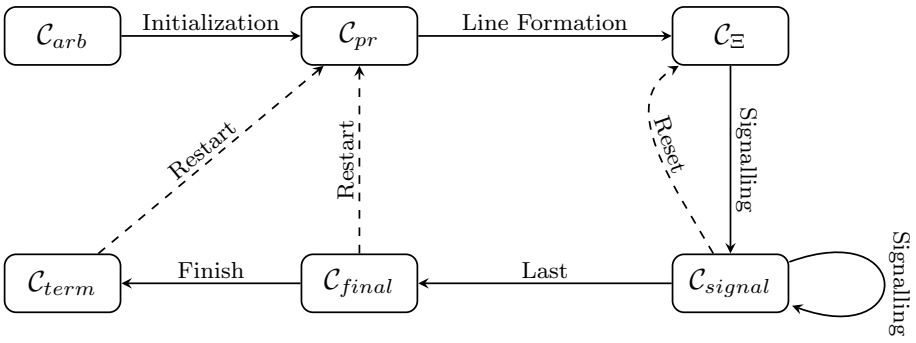


Figure 2: Transition Diagram.

We defined all the classes of configurations for the set of points occupied by the robots. These classes do not consider the color of the robots. To ensure that the robots satisfy the geometric predicate `SepL`, the colors of the robots on the same row must be the same. While the positions of the robots may belong to a particular class of configuration, there are many configurations where the colors of the robots do not satisfy the required criteria. In that case, we show that there exists at least one robot that identifies the invalidity of the configuration and leads to Restart(Reset) of the configurations to $\mathcal{C}_{pr}(\mathcal{C}_{\exists})$.

We present a transition diagram in Fig. 2 that shows the evolution of configurations over these classes of configurations, and a series of figures showing the execution of Algorithm SeparateLines in Fig. 1.

Procedure 1: Initialization

Input: Configuration $Z_i(t) \in \mathcal{C}_{arb}$
Output: destination

Let u_y be the maximum y-coordinate in $Z_i(t)$;
Let SER be the smallest enclosing rectangle;
if $\exists r_j \neq r_i : u_y > y_j > y_i$ **then**
  return (x_i, y_i); `/* Higher priority robot moves */`
else if $\exists r_j \neq r_i : x_j = x_i$ **then**
  if $y_i > y_j$ *and* (x_i, u_y) *is empty* **then**
    return (x_i, u_y); `/* Higher robot moves up */`
  else
    $r_k \leftarrow$ closest horizontal robot to r_i;
    if r_k *exists* **then**
      $x'_i \leftarrow (x_k + 2x_i)/3$;
      return (x'_i, u_y); `/* Move toward closest horizontal robot */`
    else
      $d \leftarrow u_y - y_i$;
      return $(x_i + d, u_y)$; `/* No horizontal robot case */`
else
  return (x_i, u_y); `/* Move up to` L_u `*/`

3.4 Procedure Initialization

Procedure Initialization transforms a configuration in class $\mathcal{C}_{arb}$ into a configuration in class $\mathcal{C}_{pr}$.

While doing so, it may form a line containing all the robots. The procedure operates moving the robots in a sequential manner based on the decreasing order of the y-coordinates. Since the

robots agree on the y-axis, they agree on such an order. All robots having the same y-coordinate are allowed to move at the same time. The objective of this movement is to send the robots to the upper-most line L_u of the $SER(P(t))$.

In $\mathcal{C}_{pr}$, all robots must have distinct x-coordinates; hence, the case that needs special attention is when there are multiple robots with the same x-coordinate. We remind that, in $P(0)$ the robots occupy distinct positions, hence no two robots share both x and y-coordinates.

When some robots have the same x-coordinate, then they move one by one as follows. Let r_1 and r_2 be two robots located at (x_1, y_1) and (x_1, y_2) such that $y_1 > y_2$, and no other robot is located on the line segment joining them. Now, the robot r_1 has higher priority of movement among the two. First r_1 moves vertically such that it reaches (x_1, u_y). After r_1 reaches L_u, r_2 chooses a destination that is on L_u as follows. Let r_3 at (x_3, y_3) be a robot such that $|x_3 - x_1| > 0$ is the smallest among all such robots, i.e., the closest horizontal robot. Now, r_2 sets the destination to (x_2, u_y), where $x_2 = (x_3 + 2x_1)/3$ (ref. Fig. 1b).

We need to consider a special initial configuration when all robots share the same x-coordinate but different y-coordinates. Let r_1 be the topmost robot located at (x_1, u_y). Let r_2 located at (x_1, y_2) be the robot closest to r_1. Robot r_2 sets its destination to $(x_1 + u_y - y_2, u_y)$. Once r_2 reaches this point, or stops on its way due to non-rigid movement, the configuration allows the execution of the above process, where a closest horizontal robot with a different x-coordinate is needed. The pseudocode is given in **Procedure 1:** Initialization.

3.5 Procedure Line Formation

The input configuration for Procedure Line Formation is a configuration in $\mathcal{C}_{pr}$ at time $\bar{t}$, where all robots have distinct x-coordinates. First step of line formation asks all the robots to move to the upper line L_u of $SER(P(\bar{t}))$. Since no two robots are on the same vertical line, all robots can move in parallel to reach L_u adhering to their activation schedule. Once, all the robots are located on L_u, the next step is to form an equidistant line to achieve a configuration in class $\mathcal{C}_{\Xi}$.

Without loss of generality, let $(r_1, \ldots, r_n)$ be the ordering of the robots from left to right. First they compute the equidistant positions on the line by dividing the distance between r_1 and r_n into $n-1$ parts. The ith equidistant position is at $\ell_i = (x_1 + (i-1)\beta, u_y)$, where $\beta = (x_n - x_1)/(n-1)$. A robot at r_i that is the ith robot from the left, moves to ℓ_i if no other robot is located between p_i and ℓ_i. This results in a configuration of the class $\mathcal{C}_{\Xi}$. The pseudocode is given in **Procedure 2:** Line Formation.

Procedure 2: Line Formation

Input: Configuration $Z_i(t) \in \mathcal{C}_{pr}$
Output: destination

Let u_y be the maximum y-coordinate in $Z_i(t)$;
Let $(r_1, \ldots, r_n)$ be robots ordered by increasing x-coordinate;
if $y_i \neq u_y$ **then**
  return (x_i, u_y); /* Move to upper line */
else if $\nexists r_j \neq r_i : y_j \neq u_y$ **then**
  if $Z_i(t) \notin \mathcal{C}_{\Xi}$ **then**
    $\beta \leftarrow (x_n - x_1)/(n-1)$; /* Distance between points */
    $\ell_i \leftarrow (x_1 + (i-1)\beta, u_y)$; /* Target equidistant position */
    if $p_i \neq \ell_i$ ***and*** *no robot in* $\overline{p_i \ell_i}$ **then**
      return ℓ_i; /* Move if path is clear */
else
  return (x_i, y_i); /* Stay in position */

3.6 Procedure SIGNAL

Notice that configurations of $\mathcal{C}_{\Xi}$ are included in $\mathcal{C}_{signal}$. First, we describe the behavior of the leader in case of n being odd, and then we present additional requirements when n is even. The Procedure SIGNAL works in a sequential manner, with leader(s) giving "signals" to non-leader robots by moving to a signal point.

The signaling works in three steps. Let τ be the smallest signal angle defined by $\pi/(n\kappa)$, where $\kappa = |C|$. Notice that any robot can determine κ from their observation, since no robot has a unique color. When n is odd, the leader is the median robot located at E_m and it determines the signal point. A signal point $s_{(i,k)}$ is a point at a distance $\alpha/3$ from E_m such that it makes a signal angle θ with Ξ. If the closest robot r_x at p_x with color k located at $i\alpha$ distance from E_m on the right side is in the next turn to move, the leader computes the clockwise signal angle $\theta = ((i-1)\kappa + k)\tau$. If the closest robot is on the left side, then θ is counterclockwise signal angle from Ξ. Precisely, $\angle s_{(i,k)} E_m p_x = \theta$.

The robot located at r_x observes the leader at the signal point $s_{(i,k)}$. Then it computes the corresponding signal rank i and signal color k, from θ, as follows.

$$\text{Let } \eta = \frac{\theta}{\tau}. \text{ Then } i = \left\lceil \frac{\eta}{\kappa} \right\rceil, \; k = ((\eta - 1) \bmod \kappa) + 1.$$

The robot at horizontal distance $i\alpha$ from the leader moves to the line at vertical distance $k\alpha$ from Ξ. Once the leader has verified that the signaled robot has reached its destination, it returns to E_m. Similarly, in the even case, the signal angles are determined by the smallest angle $\tau = \pi/(n\kappa)$, and the ranks are determined by the distance $i\alpha$ from the closest leader robot E_l (or E_r). At the end of Procedure SIGNAL, the configuration belongs to the class $\mathcal{C}_{final}$.

Function 3: CHECKLEADER

Input: Robot position (x_i, y_i) of r_i, Configuration $Z_i(t)$
Output: $(isLeader, E)$ where $isLeader$ is boolean and E is leader base

```
if n is odd then
    E_m ← median position on Ξ ;
    return (r_i = closest robot to E_m, E_m) ;
else
    E_l, E_r ← leftmost and rightmost positions on Ξ ;
    E ← closest of E_l, E_r to r_i ;
    return (r_i = closest robot to E, E) ;
```

Function 4: SIGNALCONVERSION

Input: θ or (i, k), mode $\in$ {angle2rank, rank2angle}
Output: (i, k) or θ

```
if mode = angle2rank then
    η ← |θ/τ| ;
    i ← ⌈η/κ⌉ ;                                  /* Target rank */
    k ← ((η − 1) mod κ) + 1 ;                    /* Target color */
    return (i, k)
else
    θ ← ((i − 1)κ + k)τ ;
    return θ
```

Procedure 5: SIGNAL

Input: Configuration $Z_i(t) \in \mathcal{C}_\Xi \cup \mathcal{C}_{signal}$
Output: destination

$\tau \leftarrow \pi/(n|C|)$; /*Unit angle between signals*/
$\theta_{reset} \leftarrow \pi/2 - \tau$; /*Special angle for reset signal*/
$(isLeader, E) \leftarrow$ CHECKLEADER$((x_i, y_i), Z_i(t))$;
if *isLeader* **then**
  if *on signal segment or at signal point* **then**
    $\theta \leftarrow \arctan((y_i - y_E)/(x_i - x_E))$; /*Current signal angle*/
    if $\theta \in \{\theta_{reset}, \pi - \theta_{reset}\}$ **then**
      if *all robots on* Ξ **then**
        return E ; /*Reset complete*/
      return $(x_E + \frac{\alpha}{3}\cos(\theta), y_E + \frac{\alpha}{3}\sin(\theta))$; /*Continue reset signal*/
    $(i, k) \leftarrow$ SIGNALCONVERSION(θ,angle2rank) ;
    $r_t \leftarrow$ robot at horizontal distance $i\alpha$ from E with color k; /*Target robot*/
    if r_t *does not exist or is at correct position* **then**
      return E ; /*Signal complete*/
    return $(x_E + \frac{\alpha}{3}\cos(\theta), y_E + \frac{\alpha}{3}\sin(\theta))$
  else
    if *different colored robots in the same row* **then**
      return $(x_E + \frac{\alpha}{3}\cos(\theta_{reset}), y_E + \frac{\alpha}{3}\sin(\theta_{reset}))$
    $r_{next} \leftarrow$ closest unpositioned robot on Ξ ;
    if r_{next} *exists* **then**
      $\theta \leftarrow$ SIGNALCONVERSION($|x_{next} - x_E|/\alpha, color(r_{next})$,rank2angle) ;
      if *(n odd &* $x_{next} < x_E$*) or (n even &* $E = E_r$*)* **then**
        $\theta \leftarrow \pi - \theta$; /*Adjust for side*/
      return $(x_E + \frac{\alpha}{3}\cos(\theta), y_E + \frac{\alpha}{3}\sin(\theta))$
else if *leader at signal point* **then**
  $\theta \leftarrow \arctan((y_s - y_E)/(x_s - x_E))$;
  if $\theta \in \{\theta_{reset}, \pi - \theta_{reset}\}$ **then**
    return (x_i, y_Ξ)
  $(i, k) \leftarrow$ SIGNALCONVERSION(θ,angle2rank) ;
  if *at* $i\alpha$ *from* E **then**
    return $(x_i, y_\Xi - k\alpha)$
return (x_i, y_i) ; /*Default: maintain position*/

3.6.1 SPECIAL SIGNAL

The SPECIAL SIGNAL handles initial configurations in $\mathcal{C}_{signal}$ that are invalid and may appear as an intermediate stage of signaling. These invalid initial configurations are detected in two distinct scenarios:

1. When the leader observes itself at a signal point in the configuration:
   - The leader checks if its signal targets the closest non-positioned robot,
   - If the signal is invalid, the leader immediately returns to Ξ.
2. When the leader is on Ξ in the configuration and the configuration contains non-leader robots at wrong positions, i.e., robots of different colors occupy the same row in the α-grid.

The leader initiates a reset on observing this invalid configuration. The reset is initiated by the leader moving to a special signal point at angle $\theta_{reset} = \pi/2 - \tau$ from Ξ, where τ is the unit angle. This angle is specifically chosen to be distinct from any regular signal angle used during normal operation.

Upon observing the leader at the reset signal point, all non-leader robots return to Ξ. The leader maintains its position at the reset signal point until all robots have returned to Ξ. Once all robots are back on Ξ, the leader returns to its base position, and the signaling process can start properly from a valid configuration of the class $\mathcal{C}_\Xi$. The pseudocode is given in **Procedure 5:** SIGNAL, which uses Function 3 CHECKLEADER and Function 4 SIGNALCONVERSION.

3.7 Procedure LAST SIGNAL

Procedure LAST SIGNAL begins when only the leaders r_l and r'_l are on Ξ, and the non-leader robots lie on their color lines.

If the two leaders have the same color and no non-leader shares their color, then `SepL` is already achieved. Otherwise, there exists a robot r_a with the same color as a leader. That robot moves vertically downwards by a distance $\alpha/4$. This indicates the leader with the same color to go to the grid row corresponding to r_a and stop. Then the robot r_a returns to its original grid point.

When the number of robots is even, let r_a be the closest robot to r_l with c_l, located at distance $|\overline{r_l r_a}|$. The robot r_a signals the leader by moving $\alpha/4$ downwards if $|\overline{r_l r_a}| < w/2$, otherwise it moves $\alpha/4$ upwards. Once the robot r_l reaches the grid row corresponding to r_a, it stops. Analogously, the robot with the same color as r'_l does the same. This achieves `SepL`. There could also be the case where the two leaders have the same color and only one robot shares their color, i.e., one robot must signal both leaders. In that case, the robot first signals its closest leader and when it is correctly positioned it signals the other one. Once both leaders are in place the signaling robot goes back to its original position on the α-grid. Note that when the number of robots is odd, there may be two robots signaling the median leader if both of them are at the same horizontal distance on either side of the leader.

Another special situation could happen when the initial configuration is part of $\mathcal{C}_{final}$, but a signaling robot r_u that is not located on the α-grid has a different color from the leader r_l. Since the leader cannot realize its own color, it moves to the grid-row corresponding to r_u, which is not desirable. In this case, any other robot r_v that observes the configuration can identify that the signaling robot r_u and the leader robot r_l are of different color. Then r_v can move horizontally by a distance $\alpha/4$ to break the α-grid. This results in a configuration in class $\mathcal{C}_{pr}$ and the robots restart Algorithm SEPARATELINES with Procedure LINE FORMATION.

An analogous situation is also applicable to configurations in $\mathcal{C}_{term}$, when the leader r_l is in the same grid-row with a robot r_w of different color. While both r_l and r_w cannot realize that they are of different colors, but any other robot r_v can realize this, and break the α-grid, forcing the robots to restart Algorithm SEPARATELINES with Procedure LINE FORMATION.

Using all the Procedures above, we present the pseudocode of Algorithm 7 SEPARATELINES.

3.8 Correctness and Complexity

We establish the correctness and complexity of Algorithm SEPARATELINES using the following lemmata and theorem. The complexity is denoted in terms of *epochs*. Remember that, an *epoch* is a minimal time interval in which every robot is activated at least once. Due to the fairness of the *ASYNC* scheduler, such an interval is guaranteed to be finite.

Observation 1. *We have the following observations with respect to the classes of configurations:*

- $\mathcal{C} = \mathcal{C}_{pr} \bigcup \mathcal{C}_{arb}$ *and* $\mathcal{C}_{pr} \bigcap \mathcal{C}_{arb} = \emptyset$*;*
- $\mathcal{C}_{signal} \subset \mathcal{C}_\alpha \subset \mathcal{C}_{pr}$*;*
- $\mathcal{C}_{term} \subset \mathcal{C}_{final} \subset \mathcal{C}_\alpha$*.*

Procedure 6: LAST SIGNAL

Input: Configuration $Z_i(t) \in \mathcal{C}_{final}$
Output: destination

$(isLeader, E) \leftarrow$ CHECKLEADER$((x_i, y_i), Z_i(t))$;
$w \leftarrow$ width of configuration ;
if *isLeader* **then**
  $r_s \leftarrow$ robot at distance $\alpha/4$ from grid point ;
  if r_s *exists* **then**
    $d \leftarrow |x_s - x_i|$; /*Horizontal distance to signaling robot*/
    if $d < w/2$ **then**
      if r_s *moved downward* **then**
        return $(x_i, y_s + \alpha/4)$; /*Accept close downward signal*/
    else
      if r_s *moved upward* **then**
        return $(x_i, y_s - \alpha/4)$; /*Accept far upward signal*/
  return (x_i, y_i) ; /*Stay if no valid signal*/
if *on grid point* **then**
  if *two non-leader robots of different color in the same row* **then**
    return $(x_i + \alpha/4, y_i)$; /*Break invalid config*/
  $d \leftarrow |x_i - x_l|$; /*Distance to nearest leader*/
  if *closest robot of the same color as nearest leader and not signaling* **then**
    if $d < w/2$ **then**
      return $(x_i, y_i - \alpha/4)$; /*Signal down if close*/
    else
      return $(x_i, y_i + \alpha/4)$; /*Signal up if far*/
if *at signal position* **then**
  if *corresponding leader at correct grid row* **then**
    return nearest grid point ; /*Return to grid*/
return (x_i, y_i) ; /*Default: maintain position*/

Algorithm 7: SEPARATELINES

Input: Robot's local view $Z_i(t)$ of configuration $P(t)$
Output: destination

if $P(t) \in \mathcal{C}_{term}$ **then**
  if *different colored robots in same row* **then**
    return $(x_i + \alpha/4, y_i)$; /*Break invalid config*/
else if $P(t) \in \mathcal{C}_{final}$ **then**
  return LASTSIGNAL$(Z_i(t))$;
else if $P(t) \in \mathcal{C}_{\Xi}$ *or* $P(t) \in \mathcal{C}_{signal}$ **then**
  return SIGNAL$(Z_i(t))$;
else if $P(t) \in \mathcal{C}_{pr}$ *and* $P(t) \notin \mathcal{C}_{\Xi}$ **then**
  return LINEFORMATION$(Z_i(t))$;
else if $P(t) \in \mathcal{C}_{arb}$ **then**
  return INITIALIZATION$(Z_i(t))$;
return (x_i, y_i) ; /*Default: maintain position*/

Lemma 1. *Given a configuration $P(t) \in \mathcal{C}_{arb}$, Algorithm* SeparateLines *using Procedure* Initialization *reaches $\mathcal{C}_{pr}$ in at most $O(n\lceil h/\delta \rceil)$ epochs, where h is the height of $SER(P(t))$.*

Proof. Procedure Initialization is invoked if the configuration is in class $\mathcal{C}_{arb}$. The objective of Procedure Initialization is to ensure that all robots end up with distinct x-coordinates.

Since the robots agree on the orientation of y-axis, they can always obtain an order with respect to the y-coordinates. The robot with the second highest y-coordinate moves to line L_u of $SER(P(t))$. In the worst case, no two robots share y-coordinates, and hence the robot movements become sequential.

In $\mathcal{C}_{pr}$, all robots must have distinct x-coordinates. Special attention is required when there are multiple robots with the same x-coordinate. By the assumption on $P(0)$, we already have distinct initial positions for all the robots, hence no two robots share both x and y-coordinates initially.

When two robots share x-coordinates, the bottom robot r_1 at (x_1, y_1) moves to a point on L_u such that it is at distance $|x_1 - x_2|/3$ from the closest horizontal robot r_2 at (x_2, y_2). This maneuver ensures that r_1 will reach a distinct point on L_u even when r_2 maybe moving similarly as r_1. Certainly, r_1 may stop due to non-rigid movement. In that case, it will have a clear path to L_u.

The maximum vertical distance that a robot needs to travel is h, and in case of non-rigid movement, the first δ movement may not contribute entirely toward vertical movement. Thus a robot needs at most $\lceil h/\delta \rceil + 1$ activations.

In the worst case, Procedure Initialization is invoked $O(\lceil h/\delta \rceil)$ times for $n-1$ robots to reach $\mathcal{C}_{pr}$. Thus in total the time complexity is $O(n\lceil h/\delta \rceil)$ epochs. □

Lemma 2. *Given a configuration $P(t) \in \mathcal{C}_{pr} \setminus \mathcal{C}_{\alpha}$, Algorithm* SeparateLines *using Procedure* Line Formation *reaches $\mathcal{C}_{\Xi}$ in at most $O(n(\lceil h/\delta \rceil + \lceil w/\delta \rceil))$ epochs, where h and w are the height and width of $SER(P(t))$, respectively.*

Proof. Any robot r_i can verify that $P(t) \in \mathcal{C}_{pr} \setminus \mathcal{C}_{\alpha}$, from its own view $Z_i(t)$. Then Algorithm SeparateLines invokes Procedure Line Formation to form the principal line Ξ. To achieve that, all robots move toward L_u based on their ranking of y-coordinates. A robot must traverse a vertical distance of at most h to reach L_u, and hence it takes at most $O(\lceil h/\delta \rceil)$ epochs. In total, after at most $O(n\lceil h/\delta \rceil)$ epochs, all robots are on L_u. Once all the robots are on L_u, a robot must traverse a horizontal distance at most w to reach its corresponding equidistant position. In each configuration, there must exist at least one robot that has no other robot blocking its path from the destination. Thus it may take at most $O(n\lceil w/\delta \rceil)$ epochs for all the robots to be equidistant on L_u to form Ξ. Hence in total, it takes at most $O(n\lceil (h+w)/\delta \rceil)$ epochs to achieve $\mathcal{C}_{\Xi}$. □

Lemma 3. *Given a configuration $P(t) \in \mathcal{C}_{\Xi} \cup \mathcal{C}_{Signal}$, Algorithm* SeparateLines *using Procedure* Signal *reaches $\mathcal{C}_{final}$ in at most $O(n\lceil w/\delta \rceil)$ epochs, where w is the width of $SER(P(t))$.*

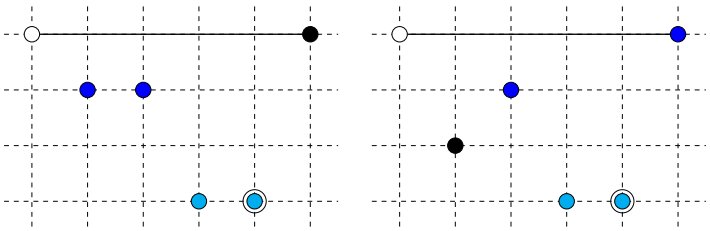
Proof. The leader robot(s) can verify whether the configuration is in $\mathcal{C}_{\Xi}$ or $\mathcal{C}_{Signal}$. If $P(t) \in \mathcal{C}_{\Xi}$, they initiate the signaling phase. If $P(t) \in \mathcal{C}_{Signal}$ and is invalid, they trigger the reset mechanism.

For the signaling phase, the leader moves to a signal point with $\alpha = w/(n-1)$, where w is the width of $SER(P(t))$. This movement requires at most $\lceil \alpha/3\delta \rceil$ epochs when $\delta < \alpha/3$. Upon observing the leader at a signal point, the signaled robot traverses at most $\kappa\alpha$ distance to reach its grid position, taking at most $\lceil w/2\delta \rceil$ epochs since $\kappa \leq n/2$ and $\alpha \leq w/(n-1)$. The leader then returns to its grid position in $\lceil \alpha/3\delta \rceil$ additional epochs.

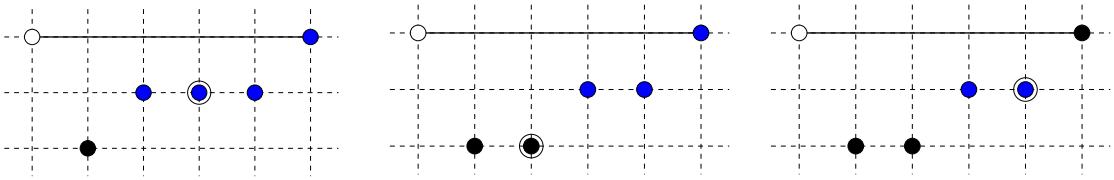
If the given configuration is invalid, the leader moves to a special signal point at angle $\theta = \pi/2 - \tau$ from Ξ in at most $\lceil \alpha/3\delta \rceil$ epochs. All non-leader robots then simultaneously return to Ξ, requiring $O(\lceil w/\delta \rceil)$ epochs to traverse the maximum vertical distance of $\kappa\alpha$. The leader completes the reset by returning to Ξ in $\lceil \alpha/3\delta \rceil$ epochs. Then the configuration is in $\mathcal{C}_{\Xi}$, and proceeds to signaling stage as before.

The procedure handles $O(n)$ robots through sequential signaling, with $n-1$ robots for odd n or $\frac{n}{2} - 1$ robots for even n. The movement protocol ensures that during signaling, only one robot moves at a time, while during reset, non-leader robots move simultaneously. Configuration validity is verified through α-grid compliance, except for leaders.

Therefore, regardless of the initial configuration ($\mathcal{C}_{\Xi}$ or $\mathcal{C}_{Signal}$), Procedure Signal reaches to $\mathcal{C}_{final}$ in $O(n\lceil w/\delta \rceil)$ epochs. □



(a) Three colors (v1) (b) Three colors (v2)



(c) Two colors (v1) (d) Two colors (v2.1) (e) Two colors (v2.2)

Figure 3: The observed configurations of a leader robot (top left) for $n = 6$ robots with different color combinations (three or two different colors, with distinct positions). The circled robot is always on the grid.

Lemma 4. *In Algorithm* SeparateLines*, a leader robot can always recognize the α-grid if the configuration is in $\mathcal{C}_{final}$.*

Proof. By definition of $\mathcal{C}_{final}$, at most two non-leader robots are not located at grid points. For configurations with $n \geq 7$, there are at least three robots at grid points, and hence a leader robot can always recognize the α-grid.

In case of $n = 5$, there are two cases: symmetric and asymmetric. In the asymmetric case, only one robot would move to signal the leader and hence there are three other robots on the grid points, which ensures that the leader can recognize the grid. In the symmetric case, two non-leader robots on either side may move to signal the leader. The grid is determined by the vertical distance between non-leader robots. If the vertical distance is $3\alpha/4$, then the top robots must have moved down. Thus the grid corresponds to the bottom robots. Similarly, if the vertical distance is $5\alpha/4$, it indicates the reverse.

This leaves us with the case for $n = 6$, where at most two non-leader robots are not on grid points and at least two robots are on grid points. Through a case-by-case analysis, we show that in all situations where exactly two robots have moved from the α-grid, the leader robots can always recognize the α-grid points.

Let us consider the cases for $n = 6$. We divide the cases based on the distinct color combinations that can be possible with 6 robots. Since no robot with unique color is present, there can be at most 3 colors for the robots. The distribution of colors leads to following three cases: $(2, 2, 2)$, $(4, 2)$ and $(3, 3)$. Since no robot is uniquely colored, a robot can always identify the case of $(2, 2, 2)$, but it cannot distinguish between $(4, 2)$ and $(3, 3)$. We consider all possible observation of a leader robot in each of the color distributions, and show that it can always establish the α-grid.

Three colors: There are two cases:

- If both leaders have the same color, then no non-leader robot moves to signal the leader robot. Hence all non-leader robots are on the α-grid (ref. Fig. 3a);

– If both leader robots have different colors, then there exists a color for which both robots are non-leaders. They remain on the α-grid point. Hence the leaders can establish the α-grid (ref. Fig. 3b).

Two colors: There are two different types of observed configuration for a leader robot:

– If there are three non-leader robots of the same color, then the middle of the three must be on the α-grid since that would not signal any of the leaders (ref. Fig. 3c);

– If there are two non-leader robots of each color, there always exists a non-leader robot which is not responsible for signaling either of the leaders. That robot is the closest non-leader robot with opposite color from the other leader. In Fig. 3d, it is the second non-leader red robot from the left. Similarly, in Fig. 3e, it is the second blue robot from the left. Since this robot would not move to signal to either of the leaders, the α-grid can be determined based on this robot's position and correspondingly the signals to the leaders.

We have shown that for all cases $n = 5$, $n = 6$, and $n \geq 7$, the leader robots can always recognize the α-grid in configurations belonging to $\mathcal{C}_{final}$. This is because there are always sufficient robots remaining on grid points to define the structure, even when the maximum number of robots have moved off the grid for signaling purposes. Therefore, we can conclude that in Algorithm SeparateLines, a leader robot can always recognize the α-grid if the configuration is in $\mathcal{C}_{final}$, for all $n \geq 5$. □

Lemma 5. *Given a configuration $P(t) \in \mathcal{C}_{final}$, Algorithm* SeparateLines *using Procedure* Last Signal *reaches $\mathcal{C}_{term}$ in at most $O(\lceil w/\delta \rceil)$ epochs starting from $\mathcal{C}_{final}$, where w is the width of $SER(P(t))$.*

Proof. Procedure Last Signal completes the final phase of robot separation through a signaling mechanism despite the robots' inability to determine their own colors. The foundation of this mechanism relies on the leader robots' ability to recognize the α-grid structure, as established by Lemma 4. This grid recognition capability is crucial as it provides a consistent reference for interpreting signals and validating configuration states throughout the procedure.

The signaling process begins with the identification of appropriate signaling robots for each leader. By the no-unique-color assumption, for each leader robot r_l, there exists at least one robot r_c of the same color. The robot r_c determines its relative position to r_l and initiates the signaling process by moving vertically by a precise distance of $\alpha/4$. For configurations with even n, the direction of this movement is determined by the horizontal distance to the leader: if $|\overline{r_l r_c}| < w/2$, the movement is downward; otherwise, it is upward. This movement pattern, coupled with the leader's ability to recognize the α-grid, ensures unambiguous signal interpretation.

The choice of $\alpha/4$ ensures that, even with two signaling robots, the grid row for the destination of the leader robot remains valid. Subsequently, the signaling robot always returns back to the closest grid point, ensuring the maintenance of the grid.

For configurations with even n and two leaders r_l and r'_l, the procedure handles several cases efficiently. If both leaders share the same color and no other robot has their color, they remain stationary. Otherwise, each leader receives signals from robots of their respective colors. In the special case where one robot must signal both leaders, it does so sequentially, ensuring the closest leader reaches its position before signaling the farthest. This sequential approach prevents signal interference and maintains the integrity of the separation process.

The complexity analysis follows from the movement constraints within the grid structure. Each leader's movement is bounded by the maximum vertical distance in the grid, which is $(\kappa + 1)\alpha \leq (n/2 + 1)w/(n - 1) < w$. Under non-rigid movement with minimum distance δ, this yields a time complexity of $O(\lceil w/\delta \rceil)$ epochs. The procedure's correctness is guaranteed by several key properties: the consistent recognition and interpretation of signals based on the α-grid structure, the unambiguous nature of the signaling protocol despite the absence of direct color knowledge, and the careful sequencing of movements that prevents interference between different color groups. These properties, combined with the grid-based movement strategy, ensure that the final configuration satisfies the line separation predicate. □

Lemma 6. *Given a configuration $P(t) \in \mathcal{C}_{final}$ or $P(t) \in \mathcal{C}_{term}$ that is invalid, there exists at least one non-leader robot r that can recognize invalidity and results in a configuration $P(t) \in \mathcal{C}_{pr}$ in one epoch.*

Proof. Consider an invalid configuration $P(t)$ in $\mathcal{C}_{final}$ or $\mathcal{C}_{term}$. $P(t) \in \mathcal{C}_{final}$ and it is invalid. This means that there exists a row in the α-grid of $P(t)$ that contains at least two robots of different color. For $n \geq 5$, there are at least four non-leader robots. Hence there exists a third non-leader robot r that can observe both of these robots. Once this robot r sees this invalidity, it moves horizontally toward its closest leader for a distance $\alpha/4$. This results in a configuration $P(t) \in \mathcal{C}_{pr}$ since the equidistant primitive is not valid anymore. This takes at most one epoch. □

Combining the above lemmata, we have the following theorem.

Theorem 1. *Given any configuration $P(t)$, Algorithm* SeparateLines *reaches $\mathcal{C}_{term}$ satisfying* `SepL` *in at most $O(n(\lceil h/\delta \rceil + \lceil w/\delta \rceil))$ epochs, where h and w are the height and width of $SER(P(t))$ respectively, and δ is the minimum distance traveled by a robot on each activation.*

Proof. We prove both correctness and complexity of Algorithm SeparateLines. First, we show that starting from any arbitrary configuration, the algorithm correctly transitions through the configuration classes to reach $\mathcal{C}_{term}$ satisfying `SepL`.

By Lemma 1, starting from any configuration in $\mathcal{C}_{arb}$, the algorithm reaches $\mathcal{C}_{pr}$ where all robots have distinct x-coordinates. From $\mathcal{C}_{pr}$, Lemma 2 ensures the algorithm reaches $\mathcal{C}_{\Xi}$ where all robots are equidistant on a horizontal line Ξ.

Once in $\mathcal{C}_{\Xi}$, Lemma 3 guarantees the algorithm reaches $\mathcal{C}_{final}$ through the signaling process, where all non-leader robots of the same color are positioned on the same row of the α-grid. If an invalid signal configuration is encountered initially, Lemma 3 also ensures that the algorithm returns to $\mathcal{C}_{\Xi}$ and subsequently reaches $\mathcal{C}_{final}$. By Lemma 4, the leader robots can always recognize the α-grid in $\mathcal{C}_{final}$, ensuring correct signaling and transitions. Similarly, if an invalid final configuration is encountered, Lemma 6 guarantees return to $\mathcal{C}_{pr}$ for restart. From a valid $\mathcal{C}_{final}$ configuration, Lemma 5 ensures the algorithm reaches $\mathcal{C}_{term}$ where all robots are positioned on their color-specific rows of the α-grid.

The resulting configuration $\mathcal{C}_{term}$ satisfies `SepL` since all robots occupy parallel horizontal lines (grid rows), each line contains only robots of the same color, and different lines contain robots of different colors.

For the time complexity, we analyze each stage: The transition from $\mathcal{C}_{arb}$ to $\mathcal{C}_{pr}$ takes $O(n\lceil h/\delta \rceil)$ epochs, from $\mathcal{C}_{pr}$ to $\mathcal{C}_{\Xi}$ requires $O(n(\lceil h/\delta \rceil + \lceil w/\delta \rceil))$ epochs, from $\mathcal{C}_{\Xi}$ to $\mathcal{C}_{final}$ needs $O(n\lceil w/\delta \rceil)$ epochs, and finally, from $\mathcal{C}_{final}$ to $\mathcal{C}_{term}$ takes $O(\lceil w/\delta \rceil)$ epochs.

Invalid configurations require at most $O(\lceil w/\delta \rceil)$ epochs for reset from invalid signal and one epoch for reset from invalid final configuration. Since each robot can detect invalidity immediately and the reset process is deterministic, reset occurs at most once during the entire execution.

Therefore, the total complexity of Algorithm SeparateLines is $O(n(\lceil h/\delta \rceil + \lceil w/\delta \rceil))$ epochs. □

4 Separation into Lines for $n \leq 4$

We consider the cases for small numbers of robots separately since in each of these cases a robot can always determine its own color due to the assumption that no robot is uniquely colored. We briefly describe the solution for each of the cases. For $n = 1$, the problem is trivially solved as a single robot already satisfies the separation predicate. For $n = 2$, since we assume at least two robots of each color exist, both robots must be of the same color. As they occupy distinct positions by assumption, they already form a line and satisfy `SepL`.

For $n = 3$, all robots must be of the same color. When at least two robots have distinct x-coordinates, they form a horizontal line akin to L_u of $SER(P(t))$, otherwise they are already collinear at $P(0)$.

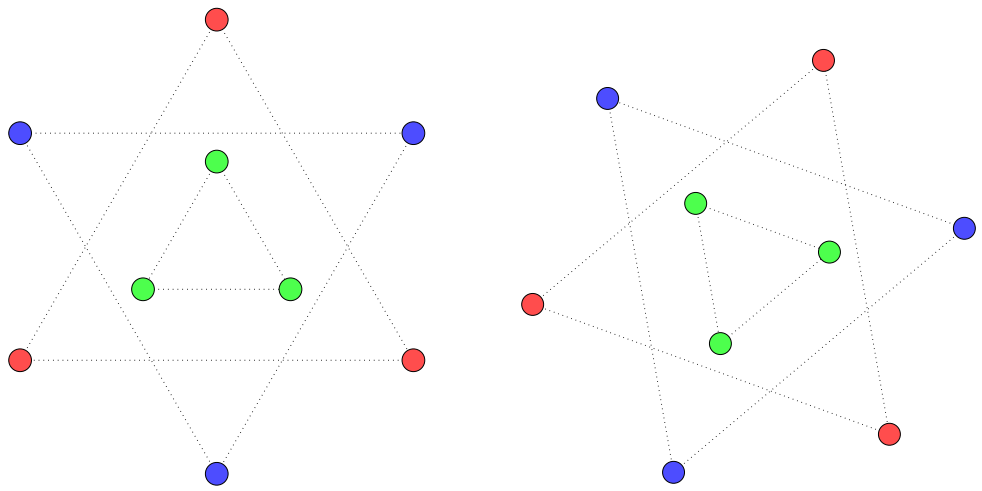
For $n = 4$, if all robots are of the same color, then they behave similarly to $n = 3$. Since no uniquely colored robot exists, the other case to consider is that there are two robots of each color

(say blue and red). In the second case, the robots form an α grid similar to the generic algorithm, and the blue robots move to occupy the bottom row, while red robots remain on the top row. The only special case to consider is a blue robot on L_u remains there until a red robot reaches L_u in a configuration that already has equidistant robots.

5 Necessity of Axis Agreement

Our result on the possibility of separation for $\mathcal{UCR}$ under the *ASYNC* adversarial scheduler holds if there is limited agreement on (at least) one axis. In this section we prove that such a requirement is necessary. We prove that, without any axis agreement the separation problem is generally unsolvable even with the additional assumption of chirality.

Theorem 2. *For a system of $n \geq 6$ unconscious colored robots with no agreement on the axis, it is impossible to achieve* `SepL` *starting from every initial configuration.*



(a) Initial symmetric views (b) Symmetry after movement

Figure 4: Nine robots: three cyan (green) at triangle corners, and six alternating black (red) and blue (blue) at hexagon corners.

Proof. Consider an initial configuration of $n \geq 6$ unconscious colored robots, where $n = g \cdot k$ for some integers $g > 1$ and $k > 2$. Assume there are g distinct colors, and k robots of each color. Position the robots such that they are located at the vertices of a regular n-gon. Furthermore, arrange the robots so that the k robots of the same color are themselves located at the vertices of a regular k-gon.

In the absence of agreement on the axis, each robot has its own local coordinate system, and there is no common frame of reference. Under the asynchronous scheduler, the adversary can always choose to activate the robots with the same color at the same time for the same duration of a Look-Compute-Move cycle.

Consider a robot r_i of color c. At any time t, in its Look phase, r_i observes the positions and colors of the other robots in its local coordinate system. Due to the symmetric initial configuration and the adversary's control over scheduling, for any two robots r_i and r_j of the same color c, the adversary can ensure that their local views are identical up to rotation and reflection. This is because there exists a symmetry of the robot configuration that maps r_i to r_j while preserving the colors.

Since the robots are homogeneous and execute the same deterministic algorithm ψ, if robots r_i and r_j (of the same color) have identical views, their Compute phases will result in the same destination relative to their local coordinate systems.

In the Move phase, even with non-rigid movements, if the computed destinations are equivalent in their local frames, and the adversary ensures similar movement durations, the actual movements of r_i and r_j will be symmetrically equivalent.

Therefore, if the robots start in this symmetric configuration, under an adversarial asynchronous scheduler, any deterministic algorithm will cause the robots of the same color to move in a coordinated, symmetric manner. This coordinated movement will maintain the symmetry of the configuration for each color group.

The degree of symmetry of a set of collinear points (forming a line) is at most 2 (identity and reflection). Since $k > 2$, the group of k robots of the same color initially forms a regular k-gon with a rotational symmetry of order k. As the algorithm progresses, the robots of the same color will maintain some degree of symmetry $s \geq k > 2$. A set of points forming a line has a symmetry of at most 2. Therefore, the group of robots of the same color cannot reach a line configuration while maintaining their symmetry of order $k > 2$. In Fig. 4, we show such a configuration for $k = 3$.

Since this argument holds for each color group, the robots cannot achieve a configuration where each color group forms a distinct line. Thus, starting from this symmetric initial configuration, it is impossible to achieve `SepL` without any agreement on the axis. □

Observe that this result implies that, for the separation problem, agreement on chirality is a computational condition strictly less powerful than agreement on one axis.

6 Conclusion

In this paper, we have presented a distributed algorithm that solves the separation problem for $\mathcal{UCR}$ under the asynchronous adversarial scheduler. Our algorithm successfully separates robots into parallel lines based on their colors, despite the robots being unaware of their own color, using a novel signaling mechanism without any explicit communication.

This algorithm can be considered as a building block for the development of complex coordination strategy of heterogeneous robots.

The algorithm works for any number of robots, provided that no robot is uniquely colored and the robots agree on the orientation of one axis. While, as we have shown, the latter condition is necessary, the necessity of the former is an open problem.

The results of this paper open several new research directions, including investigating the problem in the limited visibility and the obstructed visibility models. A particular variant may consider robots having a limited visibility for colors while having unlimited visibility for positions of the robots. An important future research direction is to extend the setting of the problem to higher dimensions (e.g., drones in 3D).

References

[1] D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta, "Computation in networks of passively mobile finite-state sensors," *Distributed Computing*, vol. 18, pp. 235—-253, mar 2006.

[2] D. Angluin, J. Aspnes, D. Eisenstat, and E. Ruppert, "The computational power of population protocols," *Distributed Computing*, vol. 20, pp. 279–304, 2007.

[3] G. A. Di Luna, P. Flocchini, T. Izumi, T. Izumi, N. Santoro, and G. Viglietta, "Fault-tolerant simulation of population protocols," *Distributed Computing*, vol. 33, no. 6, pp. 561–578, 2020.

[4] A. Dumitrescu, I. Suzuki, and M. Yamashita:, "Motion planning for metamorphic systems: feasibility, decidability, and distributed reconfiguration," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 3, pp. 409–418, 2004.

[5] N. Nokhanji and N. Santoro, "Self-repairing line of metamorphic robots," in *7th International Conference on Automation, Robotics and Applications (ICARA)*, pp. 55–59, 2021.

[6] R. Yamada and Y. Yamauchi, "Search by a metamorphic robotic system in a finite 3d cubic grid," in *1st Symposium on Algorithmic Foundations of Dynamic Networks (SAND)*, pp. 20:1–20:16, 2022.

[7] I. Suzuki and M. Yamashita, "Distributed anonymous mobile robots: Formation of geometric patterns," *SIAM Journal on Computing*, vol. 28, pp. 1347–1363, Jan. 1999.

[8] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer, "Arbitrary pattern formation by asynchronous oblivious robots," *Theoretical Computer Science*, vol. 407, pp. 412–447, Nov. 2008.

[9] N. Fujinaga, Y. Yamauchi, H. Ono, S. Kijima, and M. Yamashita, "Pattern formation by oblivious asynchronous mobile robots," *SIAM Journal on Computing*, vol. 44, pp. 740–785, Jan. 2015.

[10] S. Das, P. Flocchini, G. Prencipe, N. Santoro, and M. Yamashita, "Forming sequences of geometric patterns with oblivious mobile robots," *Distributed Computing*, vol. 28, no. 2, pp. 131–145, 2015.

[11] M. Yamashita and I. Suzuki, "Characterizing geometric patterns formable by oblivious anonymous mobile robots," *Theoretical Computer Science*, vol. 411, pp. 2433–2453, June 2010.

[12] M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro, "Distributed computing by mobile robots: Gathering," *SIAM Journal on Computing*, vol. 41, pp. 829–879, Jan. 2012.

[13] P. Flocchini, P. Prencipe, N. Santoro, and P. Widmayer, "Gathering of asynchronous mobile robots with limited visibility," *Theoretical Computer Science*, vol. 337, no. 1-3, pp. pages 147–168, 2005.

[14] N. Agmon and D. Peleg, "Fault-tolerant gathering algorithms for autonomous mobile robots," *SIAM Journal on Computing*, vol. 36, no. 1, pp. pages 56–82, 2006.

[15] P. Flocchini, N. Santoro, G. Viglietta, and M. Yamashita, "Rendezvous with constant memory," *Theoretical Computer Science*, vol. 621, pp. pages 57–72, 2016.

[16] Z. Liu, Y. Yamauchi, S. Kijima, and M. Yamashita, "Team assembling problem for asynchronous heterogeneous mobile robots," *Theoretical Computer Science*, vol. 721, pp. 27–41, Apr. 2018.

[17] S. Bhagat, P. Flocchini, K. Mukhopadhyaya, and N. Santoro, "Weak robots performing conflicting tasks without knowing who is in their team," in *21st International Conference on Distributed Computing and Networking (ICDCN)*, pp. 29:1–29:6, 2020.

[18] Y. Asahiro and M. Yamashita, "Minimum algorithm sizes for self-stabilizing gathering and related problems of autonomous mobile robots," in *25th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, pp. 312–327, 2023.

[19] P. Flocchini, D. Pattanayak, N. Santoro, and M. Yamashita, "The minimum algorithm size of k-grouping by silent oblivious robots," in *35th International Workshop on Combinatorial Algorithms (IWOCA)*, pp. 472–484, 2024.

[20] H. Seike and Y. Yamauchi, "Separation of unconscious colored robots," in *25th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, pp. 328–343, 2023.

[21] A. K. Chandra, M. L. Furst, and R. J. Lipton, "Multi-party protocols," in *15th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 94–99, 1983.

[22] P. Beame and T. Huynh, "Multiparty communication complexity and threshold circuit size of AC0," *SIAM Journal on Computing*, vol. 41, no. 3, pp. 484–518, 2012.

[23] T. Lee, N. Leonardos, M. Saks, and F. Wang, "Hellinger volume and number-on-the-forehead communication complexity," *Journal of Computer and System Sciences*, vol. 82, no. 6, pp. 1064–1074, 2016.

[24] N. Linial and A. Shraibman, "Larger corner-free sets from better NOF exactly-n protocols," *Discrete Analysis*, vol. 10, 2021.

[25] J. O'Keeffe, D. Tarapore, A. G. Millard, and J. Timmis, "Adaptive online fault diagnosis in autonomous robot swarms," *Frontiers in Robotics and AI*, vol. 5, 2018.

[26] E. Khalastchi and M. Kalech, "Fault detection and diagnosis in multi-robot systems: A survey," *Sensors (Basel, Switzerland)*, vol. 19, 2019.

[27] M. D. M. Kutzer, M. Armand, D. H. Scheidt, E. Lin, and G. S. Chirikjian, "Toward cooperative team-diagnosis in multi-robot systems," *The International Journal of Robotics Research*, vol. 27, pp. 1069 – 1090, 2008.

[28] M. Frison, N.-L. Tran, N. Baiboun, A. Brutschy, G. Pini, A. Roli, M. Dorigo, and M. Birattari, "Self-organized task partitioning in a swarm of robots," in *ANTS Conference*, 2010.

[29] G. Pini, A. Brutschy, C. Pinciroli, M. Dorigo, and M. Birattari, "Autonomous task partitioning in robot foraging: an approach based on cost estimation," *Adaptive Behavior*, vol. 21, pp. 118 – 136, 2013.