This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3089081, IEEE Access

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Solving the Pattern Formation by Mobile Robots with Chirality

SERAFINO CICERONE[1], GABRIELE DI STEFANO[1], and ALFREDO NAVARRA.[2]

[1]Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, Via Vetoio, I–67100, L'Aquila, Italy

[2]Department of Mathematics and Computer Science, University of Perugia, Via Vanvitelli 1, I–06123, Perugia, Italy.

Corresponding author: Alfredo Navarra (e-mail: alfredo.navarra@unipg.it).

This work was supported in part by ICT-AGRI-FOOD and MIPAAF under Grant CUP-J99C20000820003, through the project "HALYomorpha halys IDentification" (HALY-ID), and in part by the Italian National Group for Scientific Computation GNCS-INdAM, through the project "Traveling Salesman 2.0"–Fondo di ricerca 2020.

ABSTRACT Among fundamental problems in the context of distributed computing by mobile robots, the *Pattern Formation* (PF) is certainly the most representative. Given a multi-set F of points in the Euclidean plane and a set R of robots such that $|R| = |F|$, PF asks for a distributed algorithm that moves robots so as to reach a configuration similar to F. Similarity means that robots must be disposed as F regardless of translations, rotations, reflections, uniform scalings. In [Fujinaga et al. *SIAM J. Comput.*, 2015], PF has been approached by assuming asynchronous robots endowed with chirality, i.e. a common handedness. The proposed algorithm along with its correctness proof turned out to be flawed.
In this paper, we propose a new algorithm on the basis of a recent methodology studied for approaching problems in the context of distributed computing by mobile robots. According to this methodology, the correctness proof results to be well-structured and less prone to faulty arguments. We then ultimately characterize PF when chirality is assumed.

INDEX TERMS Distributed Algorithms; Mobile Robots; Asynchrony; Pattern Formation.

I. INTRODUCTION

One of the basic problems studied in distributed computing is certainly the *Pattern Formation* (PF) which is strictly related to *Leader Election* (see, e.g. [6], [13]).

Given a multi-set F of points in the Euclidean plane with respect to an ideal coordinate system, and a set R of mutually visible robots such that $|R| = |F|$, the Pattern Formation (PF) problem asks for a distributed and deterministic algorithm that moves robots so as to form F. As the global coordinate system might be unknown to the robots, a pattern is declared formed as soon as robots are disposed similarly to the input pattern, that is, regardless of translations, rotations, reflections, uniform scalings.

The PF problem has been largely investigated in the last years under different assumptions. Here we refer to the standard `Look`-`Compute`-`Move` model, where robots alternate between *Active* and *Inactive* states and, when active, a robot operates in cycles. In one cycle, a robot takes a snapshot of the current global configuration (`Look`) in terms of robots' positions according to its own coordinate system. Successively, in the `Compute` phase it decides whether to move toward a specific target or not, and in the positive case it moves (`Move`).

Different characterizations of the environment consider whether robots are fully-synchronous, semi-synchronous (cf. [25], [27], [28]), semi-asynchronous (cf. [5], [9]) or asynchronous (cf. [1], [3], [7], [15], [18], [19], [22]):

- *Fully-synchronous* (FSYNC): Robots are always Active and the execution of their Look-Compute-Move cycles can be logically divided into global rounds. In each round, all the robots perceive the same configuration, compute and perform their move.
- *Semi-synchronous* (SSYNC): It coincides with the FSYNC model, with the only difference that some robots might be Inactive during a round.
- *Semi-asynchronous* (SASYNC): robots still maintain a sort of synchronous behavior as each phase lasts the same amount of time, but robots can start their LCM cycles at different times. It follows that while a robot is performing a `Look` phase, other Active robots might be performing the `Compute` or the `Move` phases.
- *Asynchronous* (ASYNC): The robots are activated independently, and the duration of each phase is finite but unpredictable. As a result, robots do not have a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3089081, IEEE Access

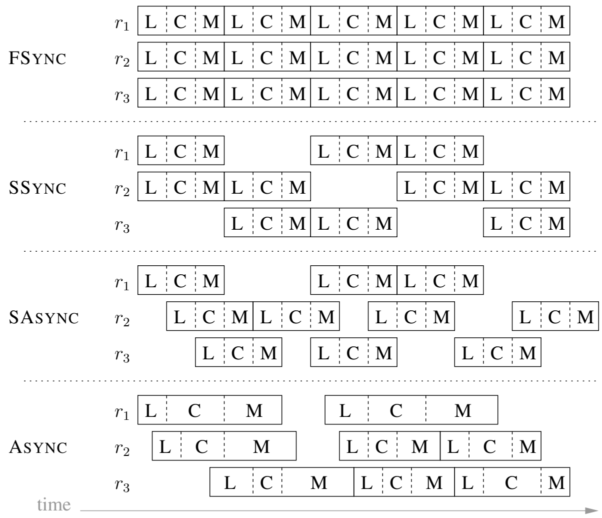


FIGURE 1: The execution model of computational cycles for each of FSYNC, SSYNC, SASYNC and ASYNC robots. The Inactive state is implicitly represented by empty time periods.

common notion of time. Moreover, they can be seen while moving, and computations can be made based on obsolete information about positions.

Figure 1 compares the four schedulers proposed in the literature.

Clearly, the four synchronization schedulers induce the following hierarchy (see, e.g. [5], [14], [16]): FSYNC robots are more powerful (i.e. they can solve more tasks) than SSYNC robots, that in turn are more powerful than SASYNC robots, that in turn are more powerful than ASYNC robots. This simply follows by observing that an ideal adversary can control more parameters in ASYNC than in SASYNC, and it controls more parameters in SASYNC than in SSYNC and FSYNC. In other words, protocols designed for ASYNC robots also work for SASYNC, SSYNC and FSYNC robots. Contrary, any impossibility result stated for FSYNC robots also holds for SSYNC, SASYNC and ASYNC robots.

Robot's capabilities are usually maintained as weak as possible so as to understand what is the limit for the feasibility of the problems. Moreover, the less assumptions are made, the more a resolution algorithm is robust with respect to possible disruptions.

To this respect, one of the minimal settings studied in [19], [25], considers robots to be:

- Autonomous: no centralized control;
- Dimensionless: modeled as geometric points in the plane;
- Anonymous: no unique identifiers;
- Oblivious: no memory of past events;
- Homogeneous: they all execute the same *deterministic* algorithm;
- Silent: no means of direct communication;
- Non-rigid: robots are not guaranteed to reach a destination within one move.

For SSYNC (and hence also for FSYNC) robots, in [25] a full characterization for PF has been provided when robots are assumed to be:

- Almost Disoriented: no common coordinate system but chirality, i.e. a common handedness.

The very same result but for ASYNC robots was claimed in [19]. Unfortunately, it comes out that the proposed algorithm is flawed, see [6]. Moreover, attempts by the same authors for fixing the problems were also not totally convincing [20]. By personnel communications (M. Yamashita, 2018), the authors admit they give up in trying publishing their erratum.

A. CONTRIBUTION

In this paper, we aim at studying PF when ASYNC robots endowed with chirality are considered. In particular, we exploit our experience in the field of distributed computing, and in particular our methodology [8] recently proposed for the designing of distributed algorithms along with the corresponding correctness proofs. We are in fact able to propose a new resolution algorithm for the considered variant of PF that along with the impossibility result provided in [25] determine a full characterization of the problem. This closes a long-standing question about the computability issue of ASYNC robots with respect to SSYNC and FSYNC, since we basically prove that in the context of PF with chirality the synchronization scheduler is irrelevant.

B. OUTLINE

The paper is organized as follows. The next section provides basic concepts necessary to formally define the addressed PF problem. Section III introduces all the basic definitions and notation. Section IV first provides a high-level description of our resolution algorithm, and then along with the subsequent Section V, formalize all the required details in a gradual way. Section V is in fact intended to explain how the proposed algorithm works by means of an extended example. Section VI concerns the correctness proof of the proposed algorithm. Finally, Section VII concludes the paper by highlighting some final remarks.

II. THE ROBOT MODEL

In this section, we first provide more details in order to complete the description of the adopted robot model and then introduce notation and properties about robot configurations.

A. ROBOTS' BEHAVIOR AND CAPABILITIES

Each robot in the system has sensory capabilities allowing it to determine the location of other robots in the plane, relative to its own location. Each robot refers in fact to a *Local Coordinate System* (LCS) that might be different from robot to robot. The robots also have computational capabilities which allow them to compute the location where to move along with the whole trajectory to trace. Each robot follows an identical algorithm that is preprogrammed into the robot. Robots alternate between *Active* and *Inactive* states. When Active, the behavior of a robot can be described according to the sequence of three phases: `Look`, `Compute`, and `Move`. Such phases form a computational cycle (or briefly a cycle) of a robot. The operations performed by each robot r in each phase will be now described in more details.

1) `Look`. The robot observes the world by activating its sensors which will return a snapshot of the positions of all other robots with respect to its LCS. Each robot is viewed as a point. Hence, the result of the snapshot (i.e., of the observation) is just a set of coordinates in its LCS.
2) `Compute`. The robot performs a local computation according to a deterministic algorithm $\mathbb{A}$ (we also say that the robot executes $\mathbb{A}$). The algorithm is the same for all robots, and the result of the `Compute` phase is a destination point along with a trajectory τ to reach it.
3) `Move`. If the destination point is the current location of r, then r performs a *nil* movement (i.e., it does not move); otherwise it moves toward the computed destination along the computed trajectory τ.

During the `Look` phase, robots can perceive *multiplicities*, that is whether a same point is occupied by more than one robot. As in [19], we assume the so-called *global-strong* multiplicity detection, that is robots can detect all multiplicities and also perceive the exact number of robots composing each multiplicity.

About movements, a strong assumption is about the so-called *rigid* movements where robots are always guaranteed to reach the destination within one LCM cycle. A weaker assumption is what we consider, that is about *non-rigid* movements: the distance traveled within a move is neither infinite nor infinitesimally small. More precisely, we can assume an adversary that has the power to stop a moving robot before it reaches its destination. However, there exists an unknown constant $\nu > 0$ such that if the destination point is closer than ν, the robot will reach it, otherwise the robot will be closer to it of at least ν. Note that, without this restriction on ν, an adversary would make it impossible for any robot to ever reach its destination.

We assume that cycles are performed according to the weakest Asynchronous scheduler (ASYNC): the robots are activated independently, and the duration of each phase is finite but unpredictable (the activation of each robot can be thought as decided by the adversary). As a result, robots do not have a common notion of time. Moreover, according to the definition of the `Look` phase, a robot does not perceive whether other robots are moving or not. Hence, robots may move based on outdated perceptions. In fact, due to asynchrony, by the time a robot takes a snapshot of the configuration, this might have drastically changed once the robot starts moving.

B. ROBOT CONFIGURATIONS

We consider a system composed by a set of n mobile *robots*. Let $\mathbb{R}$ be the set of real numbers, at any time the multiset $R = \{r_1, r_2, \ldots, r_n\}$, with $r_i \in \mathbb{R}^2$, contains the *positions* of all the robots. By abusing notation, we often refer to $r \in R$ as a *robot* instead of a robot position.

We arbitrarily fix an x-y coordinate system Z_0 and call it the *global coordinate system*. A robot, however, does not have access to it. It is used only for the purpose of description, including for specifying the input. All actions taken by a robot are done in terms of its local (and current) x-y coordinate system, whose origin always indicates its current position. Let $r_i(t) \in \mathbb{R}^2$ be the location of robot r_i (in Z_0) at time t. Then a multiset $R(t) = \{r_1(t), r_2(t), \ldots, r_n(t)\}$ is called the *configuration* of R at time t, and we simply write R instead of $R(t)$ when we are not interested in any specific time.

Regardless of the adversary, the activations of the robots determine specific ordered time instants. Let $R(t)$ be the configuration observed by some robots at time t during their `Look` phase. It follows that an *execution* of an algorithm $\mathbb{A}$ from an initial configuration R is a sequence of configurations $\mathbb{E} : R(t_0), R(t_1), \ldots$, where $R(t_0) \equiv R$, $t_{i+1} > t_i$, and $R(t_{i+1})$ is obtained from $R(t_i)$ by moving some robots according to the result of the `Compute` phase as implemented by $\mathbb{A}$. Moreover, given an algorithm $\mathbb{A}$ in ASYNC or SSYNC, there exist many executions from $R(t_0)$ depending on the activation of the robots, controlled by the adversary. It is worth to remark that initially robots are inactive, but once the execution of an algorithm $\mathbb{A}$ starts - unless differently specified - there is no instruction to stop it, i.e., to prevent robots to enter their LCM cycles. Then, the *termination* property for $\mathbb{A}$ can be stated as follows: once robots have reached the required goal by means of $\mathbb{A}$, from there on robots can perform only the *nil* movement. Sometimes termination is not even required as robots might be asked to execute infinite computations (e.g., perpetual exploration [2], [21] and patrolling [4], [12], [23]).

We now provide some definitions concerning special kinds of configurations obtainable during any execution.

Definition II.1 (Stationary robot)**.** *A robot is said to be stationary in a configuration $R(t)$ if at time t it is:*

- *inactive, or*
- *active, and during its current LCM cycle:*
  - *it has not taken the snapshot yet;*
  - *it has taken snapshot $R(t') = R(t)$, $t' \leq t$;*
  - *it has taken snapshot $R(t')$, $t' \leq t$, which leads to a nil movement.*

It is worth remarking that Definition II.1 is a refinement of the one provided in [19] that did not catch all the possible scenarios.

Definition II.2 (Stationary configuration)**.** *A configuration R is said to be stationary if all robots are stationary in R.*

Note that, according to Definition II.1, a robot r is *non-stationary* in a configuration $R(t)$, if at time t robot r is Active, has taken a snapshot $R(t') \neq R(t)$, $t' < t$, and is planning to move or is moving with a non-nil trajectory (i.e., r may give rise to what later will be better specified as a *pending move*).

1) Symmetric configurations

In the Euclidean plane, a map $\varphi : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ is called *isometry* or distance preserving if for any $a, b \in \mathbb{R}^2$ one has $d(\varphi(a), \varphi(b)) = d(a, b)$, where $d()$ denotes the standard Euclidean distance function. Examples of isometries in the plane are *translations*, *rotations* and *reflections*. An isometry φ is a translation if there exists no point x such that $\varphi(x) = x$; it is a rotation if there exists a unique point x such that $\varphi(x) = x$ (and x is called *center of rotation*); it is a reflection if there exists a line ℓ such that $\varphi(x) = x$ for each point $x \in \ell$ (and ℓ is called *axis of symmetry*).

Given an isometry φ different from the identity, the *cyclic subgroup* of order p generated by φ is given by $\{\varphi^0, \varphi^1 = \varphi \circ \varphi^0, \varphi^2 = \varphi \circ \varphi^1, \ldots, \varphi^{p-1} = \varphi \circ \varphi^{p-2}\}$, where φ^0 is the identity automorphism, $\varphi^i \neq \varphi^0$ for each $0 < i < p$, and $\varphi^p = \varphi^0$. A reflection always generates a cyclic subgroup of order $p = 2$. Whereas, the cyclic subgroup generated by a rotation can be of any finite order $p > 1$.

An *automorphism* of a configuration R is an isometry in the plane that maps robots into robots (i.e., points of R into R). The set of all automorphisms of R forms a group with respect to the composition denoted by $Aut(R)$ and called *automorphism group* of R. In general (i.e., for robots completely disoriented), the isometries in $Aut(R)$ are the identity, rotations, reflections and their compositions (translations are not possible as R contains a finite number of elements). If $|Aut(R)| = 1$, that is R admits only the identity automorphism, then R is said to be *asymmetric*, otherwise it is said to be *symmetric* (i.e., R admits rotations or reflections).

If a configuration R is symmetric due to an automorphism φ, two robots r, $r' \in R$ are *equivalent* if $r' = \varphi(r)$. As a consequence, no algorithm can distinguish between two equivalent robots, and then it cannot avoid that the two ASYNC robots start the computational cycle simultaneously. In such a case, there might be a so called *pending move*, that is one of the two robots performs its entire computational cycle while the other has not started or not yet finished its `Move` phase, i.e. its move is pending. Clearly, any other robot is not aware whether there is a pending move, that is it cannot deduce such an information from the snapshot acquired in the `Look` phase. This fact greatly increases the difficulty to devise algorithms for symmetric configurations.

2) Robots' view

According to the capabilities of the robots, by opportunely elaborating the configuration perceived with respect to its own LCS, a robot obtains what will be later called the *view* of a robot. Actually, sometimes a robot is asked to evaluate what would be the view of other robots, hence it is convenient that the view does not depend on the current LCS, as this might be completely different from cycle to cycle and from robot to robot. Hence, unless further knowledge is provided to the robots, the view should exploit only the information that all robots can equally perceive, like those concerning relative distances and angles among robots' positions. It follows that in general, in a symmetric configuration there are robots with the same view. For instance, by considering a configuration with a multiplicity, then the view cannot discriminate among the robots composing the multiplicity, i.e. a configuration with a multiplicity is always perceived as symmetric. Instead, in a symmetric configuration R without multiplicities, in the stronger model with robots aware of Z_0, R can be perceived as asymmetric by the robots as the view may exploit the coordinates of the robots to discriminate among all of them (as if they had unique identifiers).

III. PRELIMINARY CONCEPTS AND NOTATION

In this section we formalize the PF problem and provide all the notation, definitions and concepts that will be exploited later for designing our new resolution algorithm for solving the problem "PF with chirality".

A. THE PATTERN FORMATION PROBLEM

A configuration R is said *initial* if it is stationary and all elements in R are distinct, that is, no multiplicity occurs. The set of initial configurations is denoted by $\mathcal{I}$.[1]

Let P_1 and P_2 be two multisets of points: if P_2 can be obtained from P_1 by uniform scaling, possibly with additional translation, rotation and reflection, then P_2 is *similar* to P_1. Given a pattern F expressed as a multiset $Z_0(F)$, we say that an algorithm $\mathbb{A}$ *forms* F from an initial configuration R if for each possible execution $\mathbb{E} : R \equiv R(t_0), R(t_1), R(t_2), \ldots$, there exists a time instant $t_i > 0$ such that $R(t_i)$ is similar to F and no robots move after t_i, i.e., $R(t) = R(t_i)$ holds for each $t \geq t_i$.

The Pattern Formation (PF) problem can be formulated as follows:

- *Given any initial configuration R formed by n robots and any pattern F (i.e., a multiset of n elements) devise an algorithm $\mathbb{A}$, if any, able to form F from R.*

Considering the PF variant approached in [19], ASYNC robots are endowed with global strong multiplicity detection and with *chirality*, that is they share a common handedness. This of course changes their perception during the `Look` phase, as now the view can also exploit the chirality. For instance, by looking at the leftmost configuration in Figure 2, it is evident the only disposal of the robots induces a vertical axis of reflection passing through the five aligned robots.

[1]Throughout this paper, we assume that any initial configuration in $\mathcal{I}$ contains no multiplicity. This is a typical assumption since, for instance, it is impossible to ensure that robots composing a multiplicity reach different locations as all the robots execute the same algorithm.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3089081, IEEE Access

However, when chirality is assumed, the specular robots at the two sides of the axis can be associated with different views, as chirality discriminates among left and right. In particular, robots share a common clockwise direction. As a consequence, from now one we restrict the set $Aut(R)$ of all automorphisms for any configuration R to contain only the identity and possible rotations, as reflections are resolved by chirality.

Generalizing [19], we relax the requirement that the LCS specific of a single robot remains the same among different LCM cycles.

B. NOTATION

Given two distinct points u and v in the Euclidean plane, let $line(u, v)$ denote the straight line passing through these points, and let (u, v) ($[u, v]$, resp.) denote the open (closed, resp.) segment containing all points in $line(u, v)$ that lie between u and v. The half-line starting at point u (but excluding the point u) and passing through v is denoted by $hline(u, v)$. We denote by $\sphericalangle(u, c, v)$ the angle centered in c obtained by rotating clockwise $hline(c, u)$ until overlapping $hline(c, v)$. The angle $\sphericalangle(u, c, v)$ is measured from u to v in clockwise direction and the measure is always meant as positive.

Given an arbitrary multiset P of points in $\mathbb{R}^2$, $mult(p, P)$ denotes the number of occurrences of p in P, while $C(P)$ and $c(P)$ denote the smallest enclosing circle of P and its center, respectively. Let C be any circle concentric to $C(P)$. We say that a point $p \in P$ is *on* C if and only if p is on the circumference of C; ∂C denotes all the points of P that are on C. We say that a point $p \in P$ is *inside* C if and only if p is in the area enclosed by C but not in ∂C; $int(C)$ denotes all the points inside C. The radius of C is denoted by $\delta(C)$. The smallest enclosing circle $C(P)$ is unique and can be computed in linear time [24]. A useful characterization of $C(P)$ is expressed by the following property.

Property III.1. *[26] $C(P)$ passes either through two of the points of P that are on the same diameter (antipodal points), or through at least three points. $C(P)$ does not change by eliminating or adding points to $int(C(P))$. $C(P)$ does not change by adding points to $\partial C(P)$. However, it may be possible that $C(P)$ changes by either eliminating or changing positions of points in $\partial C(P)$.*

Given a multiset P, we say that a point $p \in P$ is *critical* if $C(P) \neq C(P \setminus \{p\})$.[2] It easily follows that if $p \in P$ is a critical point, then $p \in \partial C(P)$.

Property III.2. *[11] If $|\partial C(P)| \geq 4$ then there exists at least one point in $\partial C(P)$ which is not critical.*

Given a multiset P, consider all the concentric circles that are centered in $c(P)$ and with at least one point of P on them: $C^i_\uparrow(P)$ denotes the i-th of such circles, and they are ordered so that by definition $C^1_\uparrow(P)$ is the first one (which coincides with $c(P)$ when $c(P) \in P$), $C(P)$ is the last one, and the radius of $C^i_\uparrow(P)$ is greater than the radius of $C^j_\uparrow(P)$ if and only if $i > j$. Additionally, $C^i_\downarrow(P)$ denotes one of the same concentric circles, but now they are ordered in the opposite direction: $C^1_\downarrow(P) = C(P)$ is the first one, $c(P)$ is the last one when $c(P) \in P$, and the radius of $C^i_\downarrow(P)$ is greater than the radius of $C^j_\downarrow(P)$ if and only if $i < j$.

Finally, we provide some additional notation and terminology referred to a given configuration R and a given pattern F. The following definitions assume that $C(R) \equiv C(F)$ (cf. Figure 2):

- C^T the *parking circle at top level*, that is the median circle between $C(F)$ and $C^2_\downarrow(F)$ if $int(C(F)) \neq \emptyset$, otherwise the median circle between $C(F)$ and $c(F)$;
- C^B the *parking circle at bottom level*; it corresponds to the median circle between $c(R)$ and $\min\{\delta(C^2_\uparrow(R)), \delta(C^1_\uparrow(F))\}$ when $c(F) \notin F$, or the median circle between $c(R)$ and $\min\{\delta(C^2_\uparrow(R)), \delta(C^2_\uparrow(F))\}$ when $c(F) \in F$;
- Ann denotes the interior of the *annulus* comprised by $C(R)$ and C^T (hence, both the boundary circles $C(R)$ and C^T are excluded from Ann);
- given a robot $r \in \partial C(R)$, ℓ_r denotes the line segment $[c(R), r]$; ℓ_r is called *robot-ray*;
- given a point $f \in \partial C(F)$, the line segment $\ell_f = [c(R), f]$ is called *pattern-ray*;
- $Rob(\cdot)$ is a function that takes a region of the plane (e.g., annulus, sector, ray, ...) as input and returns all robots lying in the given region (e.g., $Rob(Ann)$ contains all robots in the annulus).

C. SYMMETRICITY

The PF with chirality problem was first introduced in [25] where a full characterization of the class of formable patterns for SSYNC (and for FSYNC as well) robots has been provided. The characterization makes use of the following notion of *symmetricity*.

Given a set of points P, consider a partition of P into k regular m-gons with common center $c(P)$, where $k = n/m$. Such a partition is called *regular*. The symmetricity $\rho(P)$ of P is the maximum m such that there is a regular partition of P into k regular m-gons. Notice that m points at $c(P)$ forms a regular m-gon,[3] any pair $\{p, q\}$ of points is a regular 2-gon with center the median point of the line segment $[p, q]$, and any point is a regular 1-gon with an arbitrary center. Since any P can be always partitioned into n regular 1-gons, the symmetricity $\rho(P)$ is well defined. Examples of ρ are depicted in Figure 3.(a)-(d). To this respect, notice the case in Figure 3.(c), where $\rho(P) = 1$ while P appears to be symmetric. This particular case means that whenever $c(P) \in P$, the robot on $c(P)$ can transform P into an asymmetric configuration P' with $\rho(P') = 1$ by leaving $c(P)$.

[2]Note that in this work we use operations on multisets.

[3]A multiplicity of m points, all at $c(P)$, is considered as a regular m-gon with radius zero.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3089081, IEEE Access

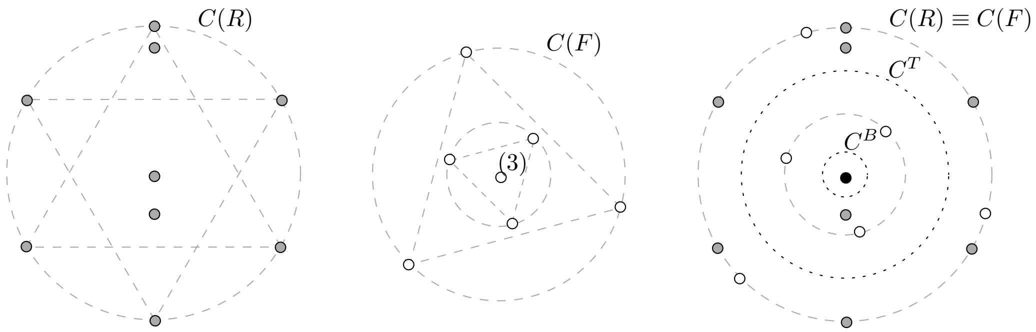


FIGURE 2: An example of input for the PF problem perceived by a generic robot according to its LCS, and related notation: on the left, an initial configuration R composed of 9 robots; on the middle, the pattern F, numbers close to points refer to multiplicities; on the right, the embedding of $C(F)$ on $C(R)$ and the parking circles C^T and C^B (robots located in points of F are represented as black points). Notice that in this example just one robot is located inside Ann.

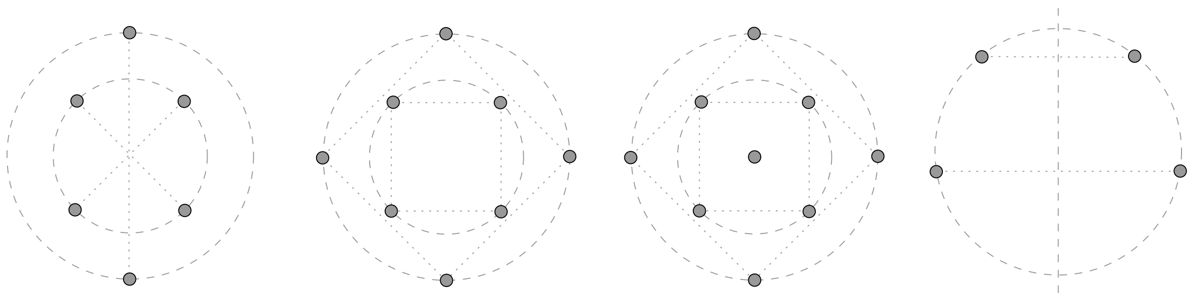


FIGURE 3: Examples of symmetricity of a set of points P. In (a), $\rho(P) = 2$; in (b), $\rho(P) = 4$; in (c), $\rho(P) = 1$; in (d), $\rho(P) = 1$.

We can now recall the characterization about formable patterns according to the notion of symmetricity.

Theorem III.3. *[25] Let R be an initial configuration of $n \geq 3$ robots and F be a pattern. F is formable from R by* FSYNC *or* SSYNC *robots with chirality if and only if $\rho(R)$ divides $\rho(F)$.*

This result states that PF highly depends on the symmetricity of both R and F, even for FSYNC robots; moreover, when robots have chirality the symmetricity is entirely represented by the parameter ρ. On the contrary, Figure 3.(d) shows that ρ is not useful when robots have no chirality since it does not take into consideration reflection symmetries. An interesting characterization about the symmetricity of points in the 3-dimensional space can be found in [28].

Notice that, the above theorem implies that, for PF, the set of unsolvable configurations, denoted as $\mathcal{U}(F)$, contains at least all configurations R such that $\rho(R)$ does not divide $\rho(F)$. Formally, $\mathcal{U}(F) \supseteq \{R : \rho(R)$ *does not divide* $\rho(F)\}$. Actually, as we will prove in Section VI by means of Theorem VI.12, $\mathcal{U}(F) \cap \mathcal{I} = \{R : \rho(R)$ does not divide $\rho(F)\}$. Concerning unsolvable configurations that are not initial, $\mathcal{U}(F)$ certainly contains those with a multiplicity composed by a number of robots greater than the number of robots composing the biggest multiplicity of F, as the adversary can always prevent to break multiplicities (i.e. to break such kind of symmetries).

Related to the symmetricity, we need to introduce one further parameter that will be exploited by our resolution algorithm.

Definition III.4. *Let C be any circle concentric to $C(R)$. $\mathcal{M}(C)$ denotes the set containing all the maximum cardinality subsets $M \subseteq \partial C$ such that all the following conditions hold:*

1) *robots in M form a regular $|M|$-gon;*
2) *$|M|$ divides $\rho(F)$;*
3) *$|M| > 1$.*

Moreover, $\mathcal{M}'(C) = \bigcup_{M \in \mathcal{M}(C)} M$, i.e., $\mathcal{M}'(C)$ is set of robots belonging to elements of $\mathcal{M}(C)$.

By referring to Figure 2, the initial configuration R (on the left) has symmetricity $\rho(R) = 1$ and the set $\mathcal{M}(C(R))$ contains two elements of three robots each, since the pattern F (on the middle) has symmetricity $\rho(F) = 3$.

The next lemma makes a relationship between $\rho(R)$ and the size of any element of $\mathcal{M}(C)$, being C any circle centered in $c(R)$ and with robots in ∂C.

Lemma III.5. *Let F be a pattern, R be a configuration such that $\rho(R)$ divides $\rho(F)$, and $M \in \mathcal{M}(C)$ with $C = C^i_{\downarrow}(R)$, $i \geq 1$. Then $\rho(R)$ divides $|M|$.*

Proof. Let r be a robot in M and let $\varphi \in Aut(R)$ be a rotation such that $\varphi^i(r)$ are distinct robots belonging to C, for each $i = 0, 1, \ldots, \rho(R) - 1$, with $r = \varphi^{\rho(R)}(r)$. If $\varphi(r) = r'$ belongs to M, then all the robots $\varphi^i(r)$ belong to M and this implies the claim.

We show by contradiction that the above case is the only possible one. In fact, if $r' \notin M$, by the equivalence of r' with r, also r' and any other robot in $\{\varphi(r) \mid r \in M\}$ must be part of a regular $|M|$-gon M', different from M. It comes out, in general, that $\{\varphi^i(r) \mid r \in M,\ i = 0, 1, \ldots, \rho(R)-1\}$ form a regular $\mathrm{lcm}(\rho(R), |M|)$-gon, where $\mathrm{lcm}(a, b)$ denotes the least common multiple of a and b. Since by hypothesis $\rho(R)$ divides $\rho(F)$ and, by definition of $\mathcal{M}(C)$, also $|M|$ divides $\rho(F)$, then $\mathrm{lcm}(\rho(R), |M|)$ divides $\rho(F)$ as well. If $\rho(R)$ does not divide $|M|$, then $\mathrm{lcm}(\rho(R), |M|) > |M|$ but this contradicts the maximality of $|M|$. □

D. VIEW OF ROBOTS

We now formalize the concept of *view* of a point in the Euclidean plane according to our needs (cf. Section II-B2). Let P be a generic multiset of points not including $c = c(P)$. For $p \in P$, we denote by $V(p)$ the view of P computed from p. This is a sequence of couples (angle, distance) defined as follows: first $(0, d(c, p))$ then, in order from the farthest to the closest point to c, all couples $(0, d(c, p'))$ for any $p' \neq p$ in $hline(c, p)$, and successively all couples $(\sphericalangle(p, c, p'), d(c, p'))$ arising from all other rays processed in clockwise order and points p' from the farthest to the closest ones to c, for each ray. If $p = c(P)$ then p is said the point in P of minimum view, otherwise any $p = argmin\{V(p') : p' \in P\}$ is said of minimum view in P.

These definitions naturally extend to any configuration R of robots and to a pattern F as well. In particular, as we are dealing with robots endowed with chirality, the clockwise direction used in the definition of the view is well-defined.

As already observed in Section II-B2, if each robot can be associated with a unique view, then the configuration is perceived as asymmetric. For instance, in Figure 3, configurations (a), (b) and (c) are all perceived as symmetric, whereas (d) is not as the clockwise direction produces different views to the potentially specular robots. In practice, the effect of assuming chirality results in breaking all reflection axes by means of the view. It comes out that if a robot views a configurations as symmetric, the only type of symmetry it can perceive is the rotation. In an asymmetric configuration, instead, each robot is associated with a different view and in particular there is only one robot associated with the minimum view. However, when there is a single robot r occupying $c(R)$ as in Figure 3.(c), then r is the only robot of minimum view by definition. This property can be exploited to break a possible rotation, if required. It follows that when $\rho(R) = 1$ then R is either perceived as asymmetric or there is a single robot in $c(R)$.

IV. ALGORITHM FOR PF: PRELIMINARIES

In this section, we present our algorithm for solving PF by ASYNC robots endowed with chirality. The algorithm and its correctness have been designed according to the methodology proposed in [8]. In the following, we describe how that methodology allowed us to break down the general problem into a set of well-defined tasks where each task can be performed by robots.

In general, a single robot has rather weak capabilities with respect to the general problem it is asked to solve along with other robots (we recall that robots have no direct means of communication). For this reason, any resolution algorithm should be based on a preliminary decomposition approach: the problem should be divided into a set of sub-problems so that each sub-problem is simple enough to be thought of as a "task" to be performed by (a subset of) robots. This subdivision could require several steps before obtaining the definition of such simple tasks, thus generating a sort of hierarchical structure.

Before presenting the algorithm, we recall that any initial configuration R does not contain multiplicities. Concerning the number of robots n, we assume $n \geq 3$, since for $n = 1$ the PF problem is trivial and for $n = 2$, either PF is trivial or unsolvable depending whether F is composed of two or one point [10], respectively. Concerning the pattern F to form, it might contain multiplicities. Moreover, according to Theorem III.3, we assume that $\rho(R)$ is a divisor of $\rho(F)$ (otherwise $R \in \mathcal{U}(F)$, that is R is unsolvable).

In the remainder, we first provide a high-level description of our strategy for the decomposition of the PF problem into tasks (cf. Section IV-A), then we summarize all the defined tasks (cf. Section IV-B), and finally we present all the details of our algorithm (cf. Section IV-C). Notice that in Section V we provide an explanatory example about the behavior of the proposed algorithm, and there we provide some missing details about moves.

A. SUBDIVISION INTO TASKS

As suggested by the methodology proposed in [8], here we describe a hierarchical decomposition of PF into sub-problems so that each sub-problem is simple enough to be formalized as a task realizable by (a subset of) robots.

The problem is initially divided into six sub-problems denoted as *Symmetry Breaking* (SB), *Reference System* (RS), *Partial Pattern Formation* (PPF), *Finalization* (Fin), *Special Cases* (SC), and *Termination* (Term). Some of these sub-problems are further refined until the corresponding tasks can be easily formalized. These initial six sub-problems are described by assuming an initial configuration R to be transformed into a pattern F.

*Symmetry Breaking (*SB*).* Consider the case in which the initial configuration admits a rotation due to an automorphism φ whose order p is not a divisor of $\rho(F)$. In this situation, by [25], $\rho(R)$ must be necessarily equal to one as otherwise the problem would be unsolvable. It follows that by the definition of symmetricity, there must be a robot occupying $c(R)$. It is mandatory for each solving algorithm to break this symmetry. In fact, without breaking the symmetry, any pair of symmetric robots may perform the same kind of movements and this may prevent the formation of the desired pattern.

In our strategy, a single task T_1 is used to address the problem SB. This task requires to carefully move the robot away from the center until to obtain a stationary asymmetric configuration. The main difficulties for SB are: (1) to avoid the formation of other symmetries that could prevent the pattern formation and (2) to correctly face the situation in which multiple steps are necessary to reach the target. In the latter case, the algorithm must detect whether there is a possible robot moving that has not yet reached a designed target.

Notice that we consider SB as a task of the Reference System sub-problem that we are going to describe in the next paragraphs.

*Reference System (*RS*) - (How to embed F on R).* This sub-problem concerns one of the main difficulties arising when the pattern formation problem is addressed: the lack of a unique embedding of F on R that allows each robot to uniquely identify its target (the final destination point to form the pattern). In particular, RS can be described as the problem of moving or matching some (minimal number of) robots into specific positions such that they can be used by any other robot as a common reference system. Such a reference system should imply a unique mapping from robots to targets, and should be maintained along all the movements of robots.

As preliminary embedding of F on R, it is assumed $C(F)$ matches with $C(R)$. Then, RS is solved by leaving on (or moving to) $C(R)$ a number $m \geq 2$ of robots so that m divides $\rho(F)$.[4] Successively, if required, the m robots left on $C(R)$ are rotated so as to form a regular m-gon. In doing so, the full embedding of F on R can be easily determined by matching the m robots on $C(R)$ with m points on $C(F)$: if there are exactly m points in $\partial C(F)$ the embedding is unique, if there are $k \cdot m$ points, with $k \geq 2$, the m robots on $C(R)$ are matched with the m points in $\partial C(F)$ having minimum view. As long as no further robots are moved to $C(R)$ and the m robots on $C(R)$ are not moved, the embedding of F on R remains well-defined. Finally, in order to guarantee stationarity before changing task, we require not only the formation of the regular m-gon but also that Ann - i.e., the annulus between $C(R)$ and C^T - does not contain robots.

Since RS is a complex problem, it is further divided into six sub-problems. As already pointed out, the first sub-problem is SB, then we need to specify $\mathrm{RS}_1, \mathrm{RS}_2, \ldots, \mathrm{RS}_5$. They are detailed as follows:

- RS_1 is responsible for opportunely moving toward C^T all robots in Ann, that is robots residing in the area between $C(R)$ and C^T - this problem is associated to task T_2.
- RS_2 is responsible for removing robots from $C(R)$ when too many robots reside there. Since such a removal can be performed in two different ways, this problem is further subdivided:
  - $\mathrm{RS}_{2.1}$ considers configurations where $\mathcal{M}(C(R)) \neq \emptyset$, that is configurations having regular m-gons on $C(R)$ such that $m > 1$ and m divides $\rho(F)$. This task removes robots from $C(R)$ until exactly one maximal regular m-gon of $\mathcal{M}(C(R))$ remains - this problem is associated to task T_3;
  - $\mathrm{RS}_{2.2}$ considers configurations where $\mathcal{M}(C(R)) = \emptyset$, that is configurations without regular m-gons on $C(R)$ such that $m > 1$ and m divides $\rho(F)$. Since such configurations are asymmetric, this task removes one non-critical robot at a time from $C(R)$ until exactly m robots remain, with m being the minimal prime factor of $\rho(F)$ or $m = 3$ (and subsequently two antipodal robots must be created by task T_6 in order to remove a non-critical robot from $C(R)$) - this problem is associated to task T_4.
- RS_3 is responsible for moving robots to $C(R)$ when there are too few robots on $C(R)$ with respect to $\rho(F)$. In particular, this task is responsible for moving robots from the interior toward $C(R)$ so as to obtain on $C(R)$ a number m of robots equal to the minimal prime factor of $\rho(F)$ - this problem is associated to task T_5.
- RS_4 is responsible for creating two antipodal robots on $C(R)$; it could be necessary as a next task of T_4 when three robots are on $C(R)$ but three is not a divisor of $\rho(F)$ - this problem is associated to task T_6.
- RS_5 is responsible for forming a uniform circle on $C(R)$ when the number m of robots on it is equal to the minimal prime factor of $\rho(F)$ - this problem is associated to task T_7.

*Partial Pattern Formation (*PPF*).* The main difficulties in this task are to preserve the reference system and to avoid collisions during the movements. The task concerns moving all robots inside $C(R)$ so as to form a preliminary pattern F' defined from F as follows. Pattern F' differs from F only for those possible points on $C(F)$ different from the m ones already matched by the resolution of problem RS - notice that PPF is addressed only once RS is solved. Such points, if any, are instead radially projected to C^T in F'. In our strategy, task T_8 is designed to solve this problem. For addressing this task we consider the area delimited by $C(R)$ as divided into m sectors. Within each sector we can guarantee that at most one robot per time is chosen to be moved toward its target: it is the one not on a target,

[4] Our strategy requires to solve RS only when $\rho(F) > 1$ and $\delta(F) > 0$. This will be explained at the end of Section IV-B.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3089081, IEEE Access

closest to an unoccupied target, and of minimum view in case of tie. We are ensured that always one single robot r per sector will be selected since the maximum symmetricity that the configuration can assume is m (we recall that, due to the solution provided for the RS problem, the robots on $\partial C(R)$ form a regular m-gon). For each sector, the selected robot is then moved toward one of the closest targets until it reaches such a point if it resides inside the same sector, or it reaches the successive (clockwise) sector. All moves must be performed so as to avoid the occurrence of collisions; hence, it follows that sometimes the movements are not straightforward toward the target point. To this end we exploit a kind of Manhattan distance (called here *Sectorial distance*) where moving between two points in the area delimited by $C(R)$ is constrained by rotating along concentric circles centered at $c(R)$ and moving along rays starting from $c(R)$.

In order to solve PPF, we make use of a procedure called `Distmin()` designed ad-hoc for computing the required trajectories according to the Sectorial distance. Once F' is formed, either F' coincides with F or it only remains to radially move robots from C^T to $C(R)$. To this aim problem Fin is addressed.

*Finalization (*FIN*).* It refers to the so-called finalization task. It occurs when the only robots not well positioned according to F are those on C^T. By guaranteeing radial movements of such robots toward $C(R)$, the formation of pattern F is completed. In our strategy, task T_9 is designed to solve this problem. It is worth to mention that while moving robots from C^T to $C(R)$, the common reference system might be loss. However, we are able to guarantee that robots can always detect they are solving Fin.

*Special Cases (*SC*).* This concerns the resolution of some easily identifiable sub-cases that have been already solved in the literature and hence can be treated apart by known algorithms. For the sake of convenience, in our strategy the resolution of the special case in which F is composed of one point with multiplicity $|R|$ (a.k.a. Gath) is delegated to the gathering algorithm provided in [10]. Similarly, when $\rho(F) = 1$ then the algorithm provided in [6] is applied as a subroutine. In both cases, the identification of the sub-problem is determined simply by looking at F, that is it does not depend on the robot movements. For such cases, our strategy considers a specific task T_{10}.

*Termination (*TERM*).* It refers to the requirement of letting robots recognize the pattern has been formed, hence no more movements are required. In our strategy, a task T_{11} is designed to address this problem. Clearly, only *nil* movements are allowed, hence if the task is started from a stationary configuration, then it won't be possible to switch to any other task.

B. THE DESIGNED TASKS

By summarizing the above analysis and according to the proposed methodology, we can say that our strategy partitions the PF problem into the following eleven tasks T_1, T_2, …, T_{11}:

- RS: Create a common reference system. General sub-problem further divided into SB, RS_1, RS_2, …, RS_5:
  - SB - Ensure $c(R)$ empty: *task* T_1.
  - RS_1 - Make Ann empty to ensure stationarity: *task* T_2.
  - RS_2: Sub-problem concerning the removal of robots from $C(R)$ until $|\partial C(R)|$ divides $\rho(F)$. It is further divided into two tasks according to the cardinality of $\mathcal{M}(C(R))$:
    - $\text{RS}_{2.1}$ - Case $\mathcal{M}(C(R) \neq \emptyset$: remove robots from $C(R)$ until exactly one maximal regular m-gon of $\mathcal{M}(C(R))$ remains: *task* T_3;
    - $\text{RS}_{2.2}$ - Case $\mathcal{M}(C(R)) = \emptyset$: remove robots from $C(R)$ until exactly m robots remain, with m being either the minimal prime factor of $\rho(F)$, or $m = 3$: *task* T_4.
  - RS_3 - Bring robots to $C(R)$ until $|\partial C(R)|$ divides $\rho(F)$: *task* T_5.
  - RS_4 - Create two antipodal robots on $C(R)$: *task* T_6.
  - RS_5 - Create a regular m-gon on $C(R)$: *task* T_7.
- PPF - Make a partial pattern formation: *task* T_8.
- Fin - Finalize the pattern formation: *task* T_9.
- SC - Solve PF by means of other algorithms when F is composed of one point with multiplicity $|R|$ or $\rho(F) = 1$: *task* T_{10}.
- Term - Identify that F is formed and hence maintain each robot without moving: *task* T_{11}.

We remark that task T_{10} uses known algorithms to address the cases in which (1) $\rho(F) = 1$ or (2) F is composed of one point with multiplicity $|R|$ (that is, $\delta(C(F)) = 0$). As a consequence, in each task different from T_{10}, our strategy can assume the following conditions: $\rho(F) > 1$ and $\delta(C(F)) > 0$.

Summarizing, our strategy will be based on the next properties maintained valid in each task different from T_{10}:

- points in $\partial C(F)$ form regular m-gons with $m \geq 2$;
- $C(F) \equiv C(R)$;
- robots movements never change the radius and the center of $C(R)$.

C. CHARACTERIZING TASKS AND MOVES

According to the LCM model and the robot obliviousness, during the `Compute` phase, each robot must be able to recognize the task to be performed just according to the configuration perceived during the `Look` phase and the input pattern F. This recognition can be performed by providing the algorithm with a logical *predicate* P_i for each task T_i. Given the perceived configuration and the input pattern F, the predicate P_i that results to be true reveals to robots that the corresponding task T_i is the task to be performed. This approach requires that the designed predicates must guarantee some properties (cf. [8]):

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3089081, IEEE Access

- Prop_1: given the pattern F, each P_i must be computable on the configuration R perceived in each `Look` phase;
- Prop_2: $P_i \wedge P_j = \texttt{false}$, for each $i \neq j$; this property ensures that at most one predicate is true;
- Prop_3: given the pattern F, for each possible perceived configuration R there must exist a predicate P_i evaluated as true. This property, along with Prop_2, allow robots to exactly recognize the task to be performed;

If we guarantee that all these properties hold, then during the `Compute` phase a robot can apply the following approach:

– if predicate P_i is detected as true, then perform move m_i associated with task T_i.

Concerning how to define the predicates, each task can be accomplished only when some *pre-conditions* are fulfilled. Hence, to define the predicates in general we need:

- *basic variables* that capture metric / topological / numerical / ordinal aspects of the input configuration which are relevant for the used strategy and that can be evaluated by each robot on the basis of its view;
- *composed variables* that express the pre-conditions of each task T_i.

All the needed basic variables useful for our algorithm are shown in Table 1. In particular, such variables capture all aspects that are relevant for our strategy.

Assuming $\texttt{pre}_i$ as the pre-conditions necessary to enter task T_i, for each $1 \leq i \leq 10$, then we propose to define predicate P_i as follows:

$$P_i = \texttt{pre}_i \wedge \neg(\texttt{pre}_{i+1} \vee \texttt{pre}_{i+2} \vee \ldots \vee \texttt{pre}_{11}) \quad (1)$$

This definition leads to the following remarks:

Remark IV.1. *During the* `Compute` *phase each robot evaluates – with respect to the perceived configuration R and the pattern F to be formed – the predicates starting from P_{11} and proceeding in the reverse order until a true pre-condition is found. In case all pre-conditions $\texttt{pre}_{11}, \texttt{pre}_{10}, \ldots, \texttt{pre}_2$ are evaluated as false, then task T_1 is performed. As such predicates are composed by the simple variables described in Table 1 which in turn are defined on the basis of rather simple properties easily observable by the robots, then predicates P_i fulfill Property* Prop_1.

Remark IV.2. *Predicates P_i fulfill Property* Prop_2. *This is directly implied by Equation 1.*

We now provide the details about the moves associated with the tasks, along with the potential tasks that are reachable after a move.

Table 2 summarizes all the ingredients necessary to define and analyze our algorithm: the first two (general) columns recall the hierarchical decomposition described in the previous section, the third column associates tasks names to sub-problems, and the fourth column defines precondition $\texttt{pre}_i$ for each task T_i. These preconditions must be considered according to Equation 1. As a consequence, such predicates are intended to be used in the `Compute` phase of each robot as described above.

The fifth column of Table 2 contains the names of the moves used in each task (we simply denote as m_i the move used in task T_i), and the specification of each move is provided in Table 3. Notice that in Table 3 some moves are directly specified, while a few of them are defined by means of specific procedures (namely, $\texttt{GoToC}^{\mathrm{T}}$, `Distmin`, `CircleForm`, `Gathering`, and `Leader` - formally defined in the next section). Moreover, all the trajectories defined in the moves are always straight lines, or arcs of circles centered in $c(R)$, or compositions of both in order to guarantee stationarity and to avoid collisions. More details that specify all target points and trajectories will be provided in Section VI.

Remark IV.3. *The defined algorithm fulfills Property* Prop_3. *This is simply implied by pre-condition $\texttt{pre}_1$ and the way predicates P_i have been defined according to Equation 1.*

The last column of Table 2 reports the possible transitions for each task. For instance, while performing task T_1 our algorithm may generate configurations belonging to the classes associated to tasks $T_1, T_2, \ldots, T_6$, and during task T_9 only configurations belonging to the classes T_9 and T_{11} may be generated.

The transitions might be classified according to different properties holding in the reachable configurations. From [8], we recall the classification of the transitions that can help in proving the correctness of the algorithm. To this aim, we first need to recall some types of configurations (beside the stationary ones defined in Section II-B):

Definition IV.4 (Almost-stationary configuration)**.** *A configuration R is said to be almost-stationary if each robot in R is either stationary or non-stationary, but in such a case the remaining part of the trajectory it has not yet traced is included into τ, where τ is the trajectory that r would compute from R.*

Definition IV.5 (Robust configuration)**.** *A configuration R belonging to a task T_i is said to be robust if each robot r in R is either stationary or non-stationary, but in such a case as long as r has not terminated its current LCM cycle the configuration still belongs to T_i.*

From the above definitions, it follows that each stationary configuration is also almost-stationary, and each almost-stationary configuration is also robust.

Definition IV.6 (Types of transitions)**.** *Let $T_j \rightarrow T_i$ be a transition. Then such a transition is stationary (almost-stationary, robust, resp.) if each $R \in T_i$ produced from any $R' \in T_j$ by applying move m_j is stationary (almost-stationary, robust, resp.).*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3089081, IEEE Access

TABLE 1: The basic Boolean variables used to define all the tasks' preconditions.

var	*definition*
$\mathtt{d}_1$	$\lvert\partial C(R)\rvert$ is not a divisor of $\rho(F)$
$\mathtt{d}_2$	$\lvert\partial C(R)\rvert$ is not the minimal prime factor of $\rho(F)$
$\mathtt{f}$	$\lvert\partial C(R)\rvert$ is smaller than the minimal prime factor of $\rho(F)$
$\mathtt{t}$	$\lvert\partial C(R)\rvert = 3$ and 2 is a divisor of $\rho(F)$
$\mathtt{u}$	Robots in $\partial C(R)$ form a regular m-gon
$\mathtt{c}$	$\partial C^1_\uparrow(R) = \{r\}$ and $d(r, c(R)) < \delta(C^B)$
$\mathtt{a}$	$Rob(Ann)$ is empty
$\mathtt{m}$	$\mathcal{M}(C(R))$ is empty
$\mathtt{p}$	F can be obtained by projecting radially on $C(R)$ all robots in $Ann \cup C^T$
$\mathtt{g}$	$\rho(F) = 1$ or F contains only one element with multiplicity $\lvert R\rvert$
$\mathtt{w}$	R is similar to F

TABLE 2: Algorithm for PF.

problem	*sub-problem*			*task*	*precondition*	*move*	*transitions*
PF	RS	SB		T_1	*true*	m_1	$T_1, T_2, T_3, T_4, T_5, T_6$
		RS_1		T_2	$\neg\mathtt{c}$	m_2	$T_2, T_3, T_4, T_6, T_7, T_8$
		RS_2	$\mathrm{RS}_{2.1}$	T_3	$\mathtt{a} \wedge \neg\mathtt{c}$	m_3	T_2, T_3, T_8
			$\mathrm{RS}_{2.2}$	T_4	$\mathtt{a} \wedge \neg\mathtt{c} \wedge \mathtt{m}$	m_4	T_2, T_4, T_6, T_7
		RS_3		T_5	$\neg\mathtt{c} \wedge \mathtt{f}$	m_5	T_2, T_5, T_7
		RS_4		T_6	$\mathtt{a} \wedge \neg\mathtt{c} \wedge \mathtt{m} \wedge \mathtt{t}$	m_6	T_3, T_6, T_9
		RS_5		T_7	$\mathtt{a} \wedge \neg\mathtt{d}_2 \wedge \neg\mathtt{u}$	m_7	T_7, T_8, T_9, T_{11}
	PPF			T_8	$\mathtt{a} \wedge \neg\mathtt{d}_1 \wedge \mathtt{u}$	m_8	T_8, T_9, T_{11}
	FIN			T_9	$\neg\mathtt{m} \wedge \mathtt{p}$	m_9	T_9, T_{11}
	SC			T_{10}	$\mathtt{g}$	m_{10}	T_{10}, T_{11}
	TERM			T_{11}	$\mathtt{w}$	*nil*	T_{11}

Note that, the types of transition form a hierarchy: each stationary transition is also almost-stationary, and each almost-stationary transition is also robust.

Remark IV.7. *Each time the creation of configuration $R(t)$, $t > 0$, determines a transition from a task T_j to task T_i (possibly $i = j$) and such a transition is stationary, almost-stationary or robust, then the analysis of the behavior of an algorithm $\mathbb{A}$ during the execution of task T_i is greatly simplified since possible movements due to past moves do not affect $\mathbb{A}$. In other words, when a transition is stationary/almost-stationary/robust, the complexity of the correctness analysis is somehow comparable to that occurring in case of* FSYNC / SSYNC *robots.*

It is worth noting that, when designing an algorithm, it is not so obvious that all the transitions can be classified according to the above defined types. For the sake of completeness, any other possible type of transition is called *unclassified*.

All the transitions reported in last column of Table 2 are summarized in the transition graph shown in Figure 4, along with the specification of the type of transitions.

Finally, in order to accomplish the designed tasks, it is possible that a resolution algorithm $\mathbb{A}$ generates (and hence must handle) configurations that are not initial, in particular not in $\mathcal{I}$. The set containing all the configurations taken as input or generated by $\mathbb{A}$ is denoted as $\mathcal{I}_{\mathbb{A}}$. Note that by definition $\mathcal{I} \setminus \mathcal{U}(D) \subseteq \mathcal{I}_{\mathbb{A}}$. Moreover, for the sake of correctness, $\mathcal{I}_{\mathbb{A}} \cap \mathcal{U}(F) = \emptyset$ must hold (i.e., no unsolvable configurations must be generated by $\mathbb{A}$).

V. ALGORITHM FOR PF: MOVES' DETAILS VIA AN EXPLANATORY EXAMPLE

In this section, we provide an explanatory example about the behavior of the proposed algorithm for the PF problem. We take advantage of this example to provide the missing details about moves. In particular, we provide the pseudo-code of procedures $\mathtt{GoToC^T}$, `Distmin`, and `CircleForm`,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3089081, IEEE Access

TABLE 3: Moves associated to tasks.

move	*definition*
m_1	Robot $r \in \partial C^1_\uparrow(R)$ moves radially to C^B
m_2	Let $C = C^i_\uparrow(R)$ be the circle contained in Ann and with minimum index i. If $\partial C \setminus \mathcal{M}'(C) \neq \emptyset$ then let R_2 be the set of robots in $\partial C \setminus \mathcal{M}'(C)$ of minimal view else let R_2 be the set of robots on C of minimal view – call $\texttt{GoToC}^\texttt{T}(R_2)$
m_3	If $\partial C(R) \setminus \mathcal{M}'(C(R)) \neq \emptyset$ then let R_3 be the set of robots in $\partial C(R) \setminus \mathcal{M}'(C(R))$ of minimal view else let R_3 be the set of robots on $C(R)$ of minimal view – call $\texttt{GoToC}^\texttt{T}(R_3)$
m_4	Let r be the non-critical robot in $\partial C(R)$ of minimal view and let $R_4 = \{r\}$ – call $\texttt{GoToC}^\texttt{T}(R_4)$
m_5	A point $p \in C(R)$ is said *forbidden* for $C(R)$ if it forms an angle of $\frac{2\pi}{n} \cdot k$ degrees in $c(R)$ with any robot on $C(R)$, for $k = 0, 1, \ldots, n$ (with n being the number of robots); Let r be the robot in $\partial C^1_\downarrow(R)$ having minimum view; r moves toward $C(R)$ avoiding forbidden points
m_6	The three robots on $C(R)$ form a triangle with angles $\alpha_1 \geq \alpha_2 \geq \alpha_3$ and let r_1, r_2 and r_3 be the three corresponding robots. For equal angles, the role of the robot is selected according to the view, i.e. if $\alpha_1 = \alpha_2$ then the view of r_1 is smaller than that of r_2. Robot r_2 rotates toward the point t such that α_1 becomes of $90°$
m_7	Call $\texttt{CircleForm}(\alpha)$, where $\alpha = 2\pi/\lvert\partial C(R)\rvert$
m_8	Call $\texttt{Distmin}()$
m_9	All robots in $Ann \cup C^T$ radially move toward $C(R)$
m_{10}	If F is composed of one point with multiplicity $\lvert R\rvert$ then call $\texttt{Gathering}()$; If $\rho(F) = 1$ then call $\texttt{Leader}()$

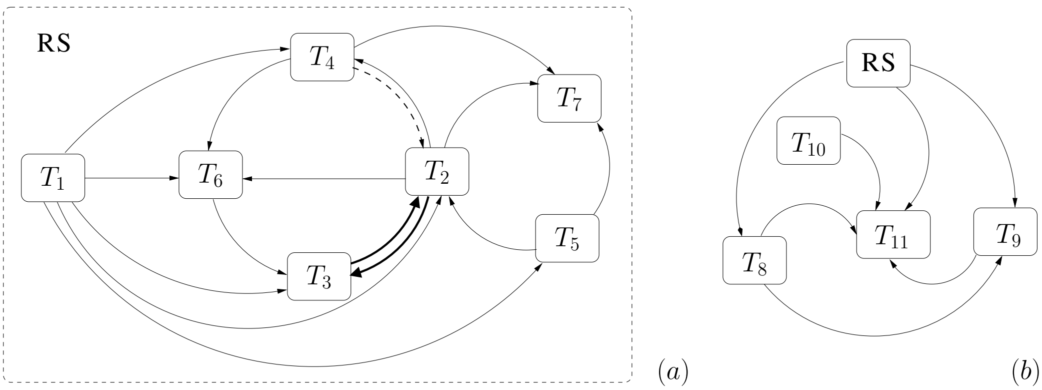


FIGURE 4: For sake of presentation, the transition graph is divided into two parts: (a) transitions among tasks in RS, and (b) transitions among RS and the tasks associated to sub-problems PPF, FIN, SC, TERM. The transitions represented by bold arrows (from/to T_2 to/from T_3) are unclassified, the one represented by the dashed arrow (from T_4 to T_2) is robust, all the others are stationary. The types of the self-loops - omitted from each task - will be discussed in the correctness proof in Section VI. Notice that apart for the self-loops, the only simple cycles are: (T_2, T_3), (T_2, T_4), (T_2, T_6, T_3), (T_2, T_4, T_6, T_3).

along with their correctness. We also briefly discuss how algorithms `Gathering` from [10] and `Leader` from [6] are exploited. Finally, we formally prove some properties about these procedures.

The example is based on the input defined in Figure 5. Notice that both the configuration R and the pattern F defined in the example are symmetric but $\rho(R) = 1$ and $\rho(F) = 4$. In the next subsections, we analyze each task separately, according to the order dictated by a possible execution of the algorithm.

A. TASK T_1

This task is associated to the sub-problem SB. As already remarked, this sub-problem is thought for breaking possible symmetries by moving a robot r from $c(R)$ (i.e., when $\rho(R) = 1$).

Concerning the current example, we now show that configuration R in Figure 5 belongs to task T_1. Each robot can detect this situation by evaluating the predicates characterizing each task. First, notice that variable $\mathtt{c}$ holds in R, and this immediately implies that the configuration does not belong to any of tasks $T_2, \ldots, T_6$ (in fact, from Table 2 it follows that variable $\mathtt{c}$ is negated in each precondition of these tasks). Since there are five robots on $C(R)$ and $\rho(F) = 4$, then each robot deduces that both $\mathtt{d}_1$ and $\mathtt{d}_2$ are true in R: this implies that R does not belong to T_7 nor to T_8. Variable $\mathtt{p}$ is false in R since F cannot be obtained by radially projecting on $C(R)$ all robots in $Ann \cup C^T$ (to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3089081, IEEE Access

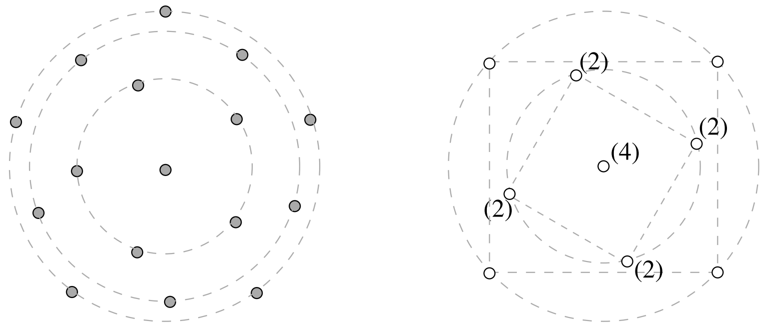


FIGURE 5: The input for the PF problem that we use as running example throughout Section V. Notice that the initial configuration R is composed of 16 robots and $\rho(R) = 1$, while the pattern F has symmetricity $\rho(F) = 4$ (numbers close to points refer to multiplicities).

observe Ann and C^T refer to Figure 6). According to the value of p, $R \notin T_9$. Variable g is false as $\rho(F) = 4$, hence $R \notin T_{10}$. Finally, w is false as R is not similar to F and hence $R \notin T_{11}$. By concluding this analysis, it follows that R does not belong to any of tasks $T_2, \ldots, T_{11}$ and according to the precondition of T_1 and to definition of predicate P_1 – cf. Equation 1, it follows that $R \in T_1$.

Since $R \in T_1$ then move m_1 is applied by the algorithm (cf. Figure 6, left side). Robot r located on $c(R)$ is moved radially along any direction to reach the parking circle C^B in order to guarantee stationarity.[5] It is worth remarking that, even when the initial configuration does not admit any symmetry but there is a robot at a distance from $c(R)$ smaller than $\delta(C^B)$, then it is moved to the parking circle C^B before starting any other task.

Once the robot in $c(R)$ has reached the specified target (possibly within multiple LCM-cycles), configuration in Figure 6, right side, is obtained. The obtained configuration is stationary and belongs to task T_2.

B. TASK T_2

This task is responsible for the correct removal of the robots from Ann, and their movement toward the parking circle C^T without generating unsolvable configurations. This removal is done in order to guarantee stationarity when later the algorithm starts removing robots from $C(R)$, when needed. Notice that there might be a number of robots equal to $\rho(R)$ that can move concurrently according to m_2 (this occurs when the processed configuration is symmetric).

To perform this task, all robots in Ann eventually move according to the trajectory computed by Procedure $\texttt{GoToC}^\texttt{T}$ specified in Algorithm 1 and used by move m_2.

When Procedure $\texttt{GoToC}^\texttt{T}$ is executed by a robot r, such robot is required to move toward a point of an arc of C^T denoted as A'_r. In particular, r is required to reach the leftmost endpoint (denoted as a_r) of A'_r or the middle point of A'_r according whether a_r is a "forbidden point for C^T" or not. Informally, a point of C^T is forbidden if it may form a regular n-gon along with the points occupied by some robots already located on C^T. The rationale underlying this definition is that when r reaches C^T all robots on such a circle are non-equivalent; this helps to ensure that no unsolvable configurations are created. Concerning the formal definition of A'_r, it depends on r and various other parameters (for a visualization of the most of them, refer to Figure 7). In what follows we formalize all such parameters. To this aim, assume that $\texttt{GoToC}^\texttt{T}$ takes as input a set of robots $R_x \subseteq Rob(Ann \cup C(R))$:

- Let $r \in R_x$ and $h = hline(c(R), r)$;
- Let r^- be the robot on $C(R)$ such that $h^- = hline(c(R), r^-)$ overlaps h by the minimal clockwise rotation;
- Let r^+ be a robot in $Ann \cup C(R)$ such that h overlaps $h^+ = hline(c(R), r^+)$ by the minimal clockwise rotation;
- Let α be the size of the smallest angle greater than $\sphericalangle(r^-, c(R), r)$, formed in $c(F)$ between two consecutive targets on $C(F)$;
- Let h' be the half-line obtained by rotating clockwise h^- of α degrees;
- Let A_r be the portion of C^T delimited by h and the closest half-line between h' and h^+. Let a_r and b_r the end points of A_r, such that b_r follows a_r in the clockwise order;
- A point $p \in C^T$ is said *forbidden* for C^T if it forms an angle of $\frac{2\pi}{n} \cdot k$ degrees in $c(R)$ with any robot on C^T, for $k = 0, 1, \ldots, n$ (we recall the reader that n denotes the number of robots);
- Let A'_r be the sub-arc of A_r starting from a_r and ending at the closest point between b_r and the first forbidden point for C^T different from a_r met in the clockwise order along A_r, if any.

By considering again our running example, we have configuration at Figure 6, right side, as input for the current task T_2. As done for the analysis of task T_1, we now formally show that such a configuration belongs to T_2.

As analyzed for task T_1 we have the same values for

[5] For the sake of completeness the exact direction toward which the robot moves will be specified in Section VI.

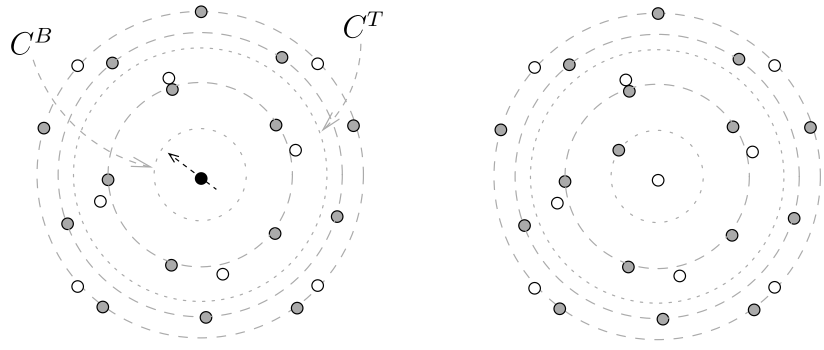


FIGURE 6: Task T_1: Ensure $c(R)$ empty. Notice the parking circles C^T and C^B.

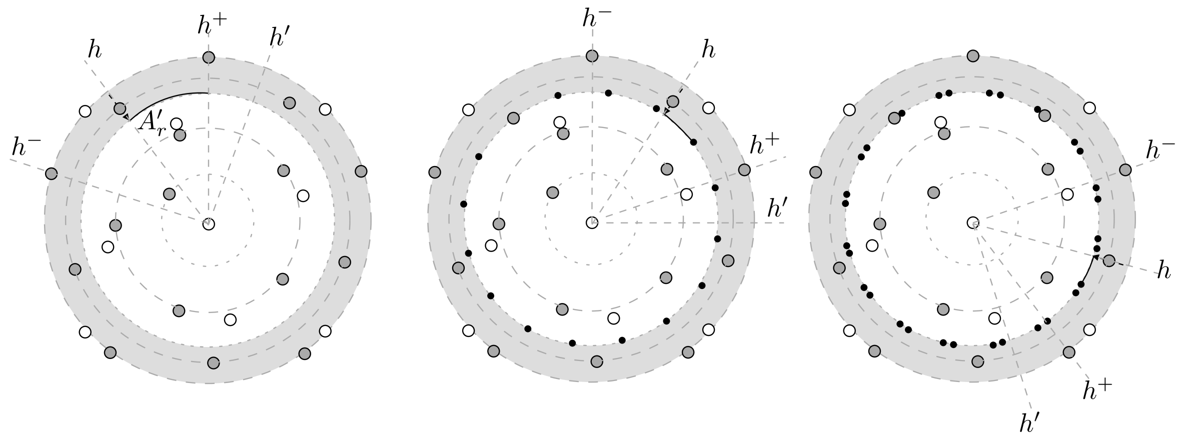


FIGURE 7: Task T_2: Make Ann (the light-gray corona) empty to ensure stationarity (notice that only the trajectories of the first three moving robots are shown). Small black dots represent forbidden points for C^T.

Algorithm 1 $\texttt{GoToC}^{\texttt{T}}(R_x)$

if $r \in R_x$ **then**
  if a_r is not forbidden for C^T **then**
    r straightly moves toward a_r
  else
    Let q be the middle point of arc A'_r;
    r straightly moves toward q until reaching C^T on the closest intersection point of C^T and $[r, q]$.

variables $\texttt{w}$, $\texttt{g}$, $\texttt{p}$, $\texttt{d}_1$, $\texttt{d}_2$, so the configuration is not in T_7, T_8, T_9, T_{10}, and T_{11}. Variable $\texttt{a}$ is false since Ann contains robots. Hence the configuration is not in T_6, T_4, and T_3. About T_5, we have that $\texttt{f}$ is false as there are too many robots on $C(R)$ with respect to $\rho(F)$. Since variable $\texttt{c}$ is now false, then the configuration belongs to T_2.

By applying move m_2, all robots in Ann eventually move toward C^T. In particular, since all robots in Ann reside on a single circle (say C) and $\mathcal{M}(C) = \emptyset$ (on C there are no regular m-gons such that $m > 1$ and m divides $\rho(F) = 4$), then m_2 calls $\texttt{GoToC}^{\texttt{T}}(R_2)$ with R_2 containing all robots on C. The trajectories performed by some robots in C are illustrated in Figure 7 (notice that, for sake of presentation, we assume that in such an example the asynchronous scheduler makes active one robot in C at a time - cf. Figure 7 where the first three executions of $\texttt{GoToC}^{\texttt{T}}$ are illustrated). Once all robots in Ann reach C^T, as we will show the obtained configuration (cf. Figure 8, left side) belongs to task T_4.

The next lemma gives important properties of Procedure $\texttt{GoToC}^{\texttt{T}}$ when applied to an initial configuration belonging to T_2.

Lemma V.1. *Let $R = R(t_0)$ be an initial configuration at time t_0 belonging to T_2, and $S(t_0)$ be the set of robots to move according to m_2. There exists a time $t_k > t_0$ where the reached configuration $R' = R(t_k)$ differs from R only for robots in $S(t_0)$ that are all on C^T in R', such that the following properties hold:*

1) *$R(t_i)$ belongs to T_2 for each $t_0 < t_i < t_k$;*
2) *$\rho(R(t_i))$ divides $\rho(F)$ for each $t_0 < t_i \leq t_k$;*
3) *R' is stationary;*
4) *$R(t_i)$, $t_0 \leq t_i \leq t_k$, has no multiplicities.*

Proof. We now prove the existence of R' and each property at items 1–4 in the statement.

- About the existence of R' and property at Item 1.
  Let $C^i_{\uparrow}(R)$ be the circle in Ann closest to C^T. Then

$S(t_0) \subseteq Rob(C^i_\uparrow(R))$ according to move m_2. The call $\texttt{GoToC}^\texttt{T}(S(t_0))$ aims to move all robots in $S(t_0)$ toward C^T. Let $\bar{R} = R(t_i)$, $t_0 < t_i < t_k$, and assume that some robots in $S(t_0)$ are not on C^T in $\bar{R}$. We now show that $\bar{R}$ is still in T_2.
Clearly $\bar{R}$ does not belong to T_{11} as there are robots in Ann. It does not belong to T_{10} because of g that only depends on F. In order to show it does not belong to T_9, it is sufficient to remind the area within a robot r is moving according to Procedure $\texttt{GoToC}^\texttt{T}$. In fact, this ensures that p remains false because of the limit established by angle α. Such a limit guarantees that along all its movement r cannot be in a position corresponding to the projection of a point from $C(F)$ to C^T. $\bar{R}$ is not in T_8, T_7, T_6, T_4 and T_3 because $\texttt{a} = \texttt{false}$. It is not in T_5 because from $\texttt{pre}_2 \wedge \neg\texttt{pre}_5$ we deduce that $\texttt{f} = \texttt{false}$ in R and since no robots are moved from $C(R)$ then f remains false. Then $\bar{R}$ is in T_2 because the value of c has not changed.
Being $\bar{R}$ in T_2, again Procedure $\texttt{GoToC}^\texttt{T}$ is applied. Note that, the input provided to the successive calls of Procedure $\texttt{GoToC}^\texttt{T}$ is constituted by a subset $S(t_i) \subseteq S(t_0)$. In fact, it involves robots lying on the current circle $C^j_\uparrow(R)$ in Ann closest to C^T, whose radius is certainly not greater than that of the initial $C^i_\uparrow(R)$ from where robots in $S(t_0)$ were selected. By applying the arguments above, we can state that by repeatedly applying $\texttt{GoToC}^\texttt{T}$, the algorithm will lead all robots in $S(t_i)$ to reach C^T. This implies there exists a time where the portion of Ann delimited by $C^i_\uparrow(R)$ and C^T, and excluding such circles, will not contain robots, eventually. At this time, either all robots contained in $S(t_0)$ have reached C^T or some of them are still on $C^i_\uparrow(R)$. In the latter case, move m_2 ensures to call $\texttt{GoToC}^\texttt{T}$ providing as input only the robots originally contained in $S(t_0)$.
By reconsidering the above analysis, we conclude that all configurations generated while robots in $S(t_0)$ are moved toward C^T belong to T_2. Once all such robots reach C^T, say at time $t_k > t_0$, then the requested configuration R' is obtained.

- About property at Item 2. Consider two different cases for $R = R(t_0)$: $Rob(C^T) = \emptyset$ and $Rob(C^T) \neq \emptyset$.
If $Rob(C^T) = \emptyset$, let us first analyze the case when $\partial C^i_\uparrow(R) \setminus \mathcal{M}'(C^i_\uparrow(R)) = \emptyset$. When m_2 is applied to configuration $R = R(t_0)$, at most $S(t_0)$ robots will move at the same time. If more than one robot moves, this is because they are of minimal view and, by Lemma III.5, if one of them belongs to an element $M \in \mathcal{M}(C^i_\uparrow(R))$, then all the other robots belong to the same regular $|M|$-gon. The robots move radially toward C^T, as so far there is no forbidden point for C^T. The robots that trace concurrently the same distance could form a regular $|M'|$-gon, but in this case, $|M'| \leq |M|$ and $|M'|$ divides $\rho(R)$. Then the symmetricity of the whole configuration divides $\rho(R)$, which in turn divides $\rho(F)$. Possibly, some robots reach C^T whereas some other are stopped before by the adversary or they do not start moving yet. In such cases, the trajectories of the robots might change in order to reach C^T by avoiding the forbidden points generated by robots arrived on C^T. Then, each configuration obtained while the remaining robots move toward C^T cannot have a symmetricity larger than $\rho(R)$ (this could be obtained only if the robots reach the forbidden points for C^T). Moreover the symmetricity of any of these configurations has to divide $\rho(R)$ because, otherwise, there is an automorphism φ such that one robot r of the first arrived on C^T should be equivalent to a robot $r' = \varphi(r)$, but this violates the requirement for r' to avoid forbidden points for C^T.
As the above property holds for each generated configuration $\bar{R}$, when robots reach C^T by successive calls of Procedure $\texttt{GoToC}^\texttt{T}$, then we conclude $\bar{R}$ is such that $\rho(\bar{R})$ divides $\rho(R)$ and then $\rho(F)$. The same considerations hold for $\rho(R')$.
Let us now analyze the case when $Rob(C^T) = \emptyset$ and $\partial C^i_\uparrow(R) \setminus \mathcal{M}'(C^i_\uparrow(R)) \neq \emptyset$. Let $\mathcal{M}'(C^i_\uparrow(R)) \neq \emptyset$. Similarly as above, Procedure $\texttt{GoToC}^\texttt{T}$ is called until all the robots in $S(t_0)$ are moved from $C^i_\uparrow(R)$ to C^T. By Lemma III.5, the symmetricity of each generated configuration $\bar{R}$ as well as R' divide $|M|$. Then both $\rho(\bar{R})$ and $\rho(R')$ divide $\rho(F)$. If instead $\mathcal{M}'(C^i_\uparrow(R)) = \emptyset$, then $|S(t_0)| = 1$, the configuration is asymmetric and it is maintained as such by means of Procedure $\texttt{GoToC}^\texttt{T}$ because the only moved robot cannot be equivalent to any other until it reaches C^T. Then for each generated configuration $\bar{R}$, $\rho(\bar{R}) = \rho(R') = 1$ that obviously divide $\rho(F)$.

Finally, consider the case when $Rob(C^T) \neq \emptyset$ in $R(t_0)$. The analysis is basically the same as above, with the only difference that now there are already some forbidden points for C^T and hence the trajectories of robots in $S(t_0)$ initially are not necessarily radial toward C^T.
- About property at Item 3. As shown above, starting from R, all calls of Procedure $\texttt{GoToC}^\texttt{T}$ only involve robots originally contained in $S(t_0)$. Any other robot does not move, that is it is stationary. Once all the robots in $S(t_0)$ reach C^T, $R(t_k) = R'$ is obtained which is then stationary.
- About property at Item 4. According to Procedure $\texttt{GoToC}^\texttt{T}$, configuration R' has no multiplicities since each robot r moves toward C^T in a region of Ann confined by: C^T, the rays from $c(R)$ passing through r itself, and the next robot r^+ in the clockwise direction on $Ann \cup C(R)$. In this region there are no robots and no other robots enter such a region. Moreover, the destination point on C^T cannot be occupied by a robot, as otherwise by definition it would be a forbidden point

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3089081, IEEE Access

for C^T.

□

C. TASK T_4

In order to solve the sub-problem RS, that is the creation of a common reference system, task T_4 is meant to manage the cases in which there are too many robots on $C(R)$ with respect to $\rho(F)$. In particular, task T_4 is specialized to manage the cases $\mathcal{M}(C(R)) = \emptyset$. We recall that $\mathcal{M}(C(R))$ denotes the set containing all the maximum cardinality subsets $M \subseteq \partial C(R)$ such that $|M| > 1$, robots in M form a regular $|M|$-gon, and $|M|$ divides $\rho(F)$. Since the input configuration R and the pattern to form must guarantee that $\rho(R)$ divides $\rho(F)$, then $\mathcal{M}(C(R)) = \emptyset$ implies that R is asymmetric. This allows the algorithm to remove one robot at a time from $C(R)$ until exactly m robots remain, with m being the minimal prime factor of $\rho(F)$ or $m = 3$.

Clearly, the removal of robots must be done very carefully so as to guarantee that $C(R)$ does not change (hence, each time the moving robot must be non-critical). Moreover, if $\rho(F)$ is even and hence only two robots must remain in $C(R)$, then it is possible that T_4 must terminate with three robots on $C(R)$ instead on two (it is possible that each of the three remaining robots is critical). In this case, task T_6 is required before the removal of the last robot from $C(R)$, that is two antipodal robots must be created on $C(R)$ as otherwise the smallest enclosing circle of the robots would change with respect to the initial one.

For this task, again Procedure $\mathtt{GoToC^T}$ is used. According to move m_4, it is performed by the non-critical robot in $\partial C(R)$ of minimal view. In this way, the moving robot will reach C^T by also ensuring that the new configuration still guarantees that $\rho(R)$ divides $\rho(F)$. It is worth to remark that in case the moving robot is stopped by the adversary before reaching the parking circle, then task T_2 is applied again to make Ann empty (in other words, T_2 collaborates with T_4 to correctly transfer robots from $C(R)$ to C^T).

Concerning the running example, Figure 8 (left side) shows the configuration belonging to task T_4. This membership can be verified as follows. As analyzed for tasks T_1 and T_2 we have the same values for variables $\mathtt{w}$, $\mathtt{g}$, $\mathtt{p}$, $\mathtt{d}_1$, $\mathtt{d}_2$, so the configuration is not in T_7, T_8, T_9, T_{10}, and T_{11}. Variables $\mathtt{t}$ and $\mathtt{f}$ are both false, so the configuration is not in T_6 nor in T_5. Since the precondition $\mathtt{pre}_4 = \mathtt{a} \wedge \neg\mathtt{c} \wedge \mathtt{m}$ holds (in fact, here $\mathtt{a} = \mathtt{true}$, $\mathtt{c} = \mathtt{false}$, and $\mathtt{m} = \mathtt{true}$), then the predicate P_4 holds and hence the current configuration belongs to T_4.

Figure 8 (right side) shows the stationary configuration obtained after two consecutive applications of task T_4. Since this configuration contains three robots on $C(R)$ and $\rho(F) = 4$, then it must be processed by T_6 in order to guarantee two antipodal robots on $C(R)$ before leaving two robots on $C(R)$.

D. TASK T_6

This task is performed when there are exactly three robots on $C(R)$, 3 does not divide $\rho(F)$, and $\rho(F)$ is even. In such a case, one of the three robots, chosen so as to not modify $C(R)$, rotates until it becomes antipodal with respect to one of the other two robots. Once this happens, variable $\mathtt{m}$ becomes false since a regular 2-gon is created on $C(R)$.

Consider the running example of Figure 9 (left side). This configuration belongs to T_6. In fact, as analyzed in previous tasks we have the same values for variables $\mathtt{w}$, $\mathtt{g}$, $\mathtt{p}$, $\mathtt{d}_1$, $\mathtt{d}_2$, so the configuration is not in T_7, T_8, T_9, T_{10}, and T_{11}. Instead, now $\mathtt{a} = \mathtt{true}$ (i.e., there are no robots in Ann), $\mathtt{c} = \mathtt{false}$ (i.e., there are no robots in the interior of C^B), $\mathtt{m} = \mathtt{true}$ (i.e., there are no regular 2-gons in $C(R)$), and $\mathtt{t} = \mathtt{true}$ (i.e., $|\partial C(R)| = 3$ and 2 is a divisor of $\rho(F)$). Hence the predicate defining T_6 is true.

The three robots on $C(R)$ form a triangle with angles $\alpha_1 \geq \alpha_2 \geq \alpha_3$ where r_1, r_2 and r_3 are the three corresponding robots. The move planned for this task (cf. move m_6) rotates r_2 along $C(R)$ so as to obtain a configuration with two antipodal robots on $C(R)$. Once this happens (and, as usual, it may require multiple LCM cycles), the configuration belongs to T_3 as there is a regular 2-gon on $C(R)$, with 2 being a divisor of $\rho(F)$ but with a third robot that must be moved from $C(R)$ toward C^T. Such a movement initiated by T_3 might be continued via task T_2 if the robot does not conclude its movement within one LCM cycle.

E. TASK T_3

Together with task T_4, this task is meant to manage the cases in which there are too many robots on $C(R)$ with respect to $\rho(F)$. In particular, task T_3 is specialized to manage the case in which $\mathcal{M}(C(R)) \neq \emptyset$.

The move planned for this task is m_3 and it carefully moves robots from $C(R)$ toward the parking circle C^T by means of Procedure $\mathtt{GoToC^T}$. According to its specification, we observe that it considers two cases: (1) if $\partial C(R) \setminus \mathcal{M}'(C(R)) \neq \emptyset$ then all robots of minimal view in $\partial C(R) \setminus \mathcal{M}'(C(R))$ are moved, otherwise (2) all robots on $C(R)$ of minimal view are moved. Notice that it is possible that even though R might be symmetric, its symmetricity is (or becomes) smaller than $\rho(F)$. However, by Lemma III.5 we are ensured that $\rho(R)$ remains a divisor of $\rho(F)$ as long as T_3 is applied. Moreover, even in the possible case where $\rho(R) > 1$, due to the ASYNC model not all robots belonging to a same regular m-gon (say M) are necessarily active, and hence after some LCM cycles some of such robots may be in Ann while some other may still stay on $C(R)$. Any robot in Ann is then moved by T_2, and once T_2 has completely removed robots from Ann, then the remaining robots of M left on $C(R)$ are later processed again by T_3 since they result to be in $\partial C(R) \setminus \mathcal{M}'(C(R))$.

It is worth to remark that, as soon as a robot leaves $C(R)$, variable $\mathtt{a}$ becomes false, and task T_2 might be invoked.

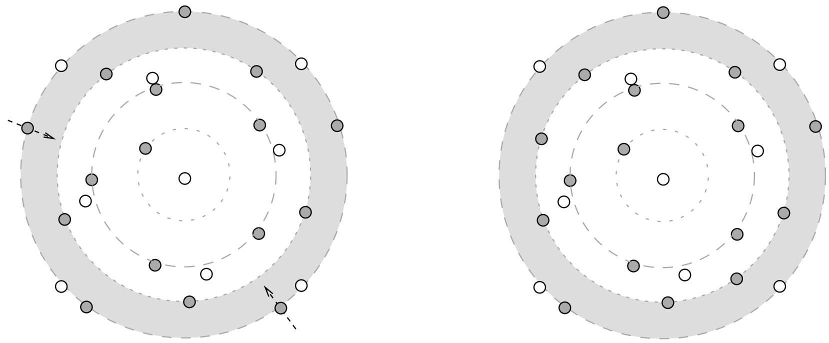


FIGURE 8: Task T_4: Case $\mathcal{M}(C(R)) = \emptyset$, removing robots from $C(R)$ until exactly m robots remain, with m being the minimal prime factor of $\rho(F)$ or $m = 3$. The configuration on the left side is obtained from Figure 7 after all robots in Ann reached the parking circle C^T.

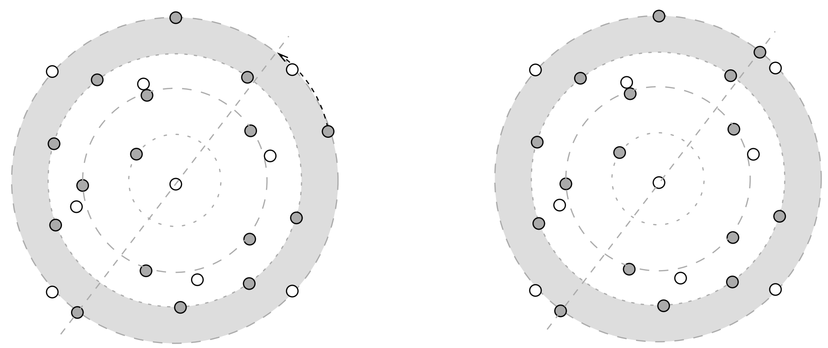


FIGURE 9: Task T_6: Create two antipodal robots on $C(R)$.

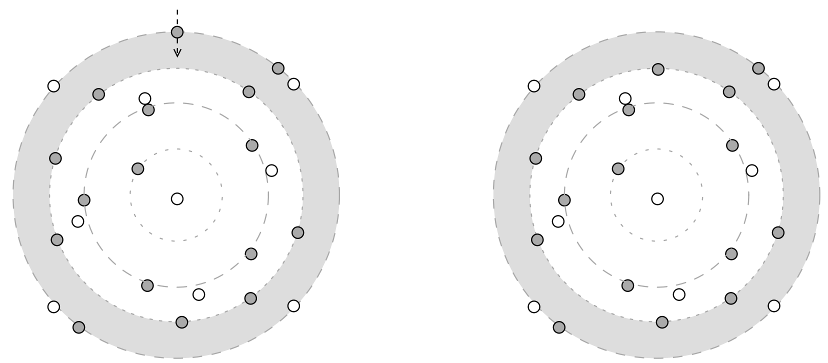


FIGURE 10: Task T_3: Case $\mathcal{M}(C(R)) \neq \emptyset$, removing robots from $C(R)$ until exactly one maximal regular m-gon of $\mathcal{M}$ remains.

Consider the running example of Figure 10 (left side). This configuration belong to T_3. In fact, as analyzed in previous tasks we have the same values for variables $\mathtt{w}$, $\mathtt{g}$, $\mathtt{p}$, $\mathtt{d}_1$, $\mathtt{d}_2$, so the configuration is not in T_7, T_8, T_9, T_{10}, and T_{11}. Variable $\mathtt{m} = \mathtt{false}$ (there is one regular 2-gon in $C(R)$ and $\rho(F)$ is even), so the configuration is not in T_6 or T_4; variable $\mathtt{f} = \mathtt{false}$ ($|\partial C(R)| = 3$ and $\rho(F) = 4$), hence it does not belong T_5. In conclusion, since precondition $\mathtt{pre}_3 = \mathtt{a} \wedge \neg\mathtt{c} = \mathtt{true}$, then the configuration belongs to T_3.

Move m_3, possibly interleaved by move m_2, will lead to obtain the configuration shown in Figure 10 (right side). In this configuration the problem RS is solved, and hence the subsequent sub-problem PPF can be addressed by performing the planned task T_8.

F. TASK T_8

This task is responsible for solving the PPF sub-problem. In particular, it moves all robots that are inside or on C^T toward the targets computed with respect to the embedding of the *modified pattern* F'. As described in Section IV-A (cf. description of PPF), pattern F' differs from F only for those possible targets on $C(F)$ different from the m ones already matched by the resolution of sub-problem RS (i.e., the embedding of F on R and hence the embedding of F' on R are well-defined, cf. description of RS). Such additional points on $C(F)$, if any, are instead radially projected to C^T in F'. In our strategy, task T_8 is designed to solve the pattern formation problem with respect to F'.

Concerning the running example, Figure 11 shows how each robot views the embedding of F' in the current configuration. It is worth to note that, during this task, (1) no robots on $C(R)$ move, and (2) no robots are moved out of C^T (i.e., no robot enters in Ann); this implies that the embedding of F' remains the same during the whole task T_8.

To solve PPF, at any time, each robot inside C^T must determine (1) whether it is already on its target or not (i.e., whether it is *matched* or not), (2) if it is not matched, which is its target, and (3) whether it is its turn to move or not. To this aim, and to formally define Procedure `Distmin` that is used to solve task T_8, we need some further definitions and properties (cf. Figure 12).

Let P be a multiset of points and let $p, q \in P$. We denote by C_p and C_q the circles centered in $c(P)$ and with radii $d(c(P), p)$ and $d(c(P), q)$, respectively. Points p' and q' correspond to $C_q \cap hline(c(P), p)$ and $C_p \cap hline(c(P), q)$, respectively (cf. Figure 12.(a)). Symbol $AS(p, q)$ is used to denote the *annulus sector* given by the area enclosed by circles C_p and C_q, and by segments $[p, p']$ and $[q', q]$, subtending $\sphericalangle(p, c(R), q)$ (cf. Figure 12.(b)). Notice that when $\sphericalangle(p, c(R), q) = \pi$, by definition $AS(p, q)$ corresponds to the annulus sector spanned by $hline(c(P), p)$ to overlap $hline(c(P), q)$ by means of a clockwise rotation. We say that $AS(p, q)$ is *degenerate* when it reduces to a point (i.e., when $p = q$) or to a segment/arc (i.e., when p and q lie on the same ray/circle).

Definition V.2 (Sectorial path and sectorial distance)**.** *Let P be a multiset of points in the plane. Given $p, q \in P$, the sectorial path between p and q is given by either the arc $\widehat{pq'}$ composed with the segment $[q', q]$, or the segment $[p, p']$ composed with the arc $\widehat{p'q}$ (cf. Figure 12.(a)). The sectorial distance between p and q is denoted by $dist(p, q)$ and if $\delta(C(P)) = 0$ then $dist(p, q) = 0$, else $dist(p, q) = |d(p, c(P)) - d(q, c(P))|/\delta(C(P)) + \min\{\sphericalangle(p, c(P), q), \sphericalangle(q, c(P), p)\}/\pi$.*

Informally, the sectorial distance is a sort of Manhattan distance where moving between two points is constrained by rotating along concentric circles centered at $c(P)$ and moving along rays starting from $c(P)$. It is easy to verify that function $dist()$ is in fact a distance function.

Property V.3. *Let P be a multiset of points in the plane, and let $p, q \in P$. For each point $s \in AS(p, q)$ it follows that $dist(p, q) = dist(p, s) + dist(s, q)$.*

According to this property, the sectorial distance implies the existence of infinitely many shortest paths (composed of one or more sectorial paths) connecting two distinct points (cf. Figure 12.(c)).

The above notation and definitions will be applied to what was before informally called a "sector". The following definition formalizes such a concept.

Definition V.4 (Sector)**.** *Let ℓ and ℓ' be two consecutive (clockwise) robot-rays. A sector S is the area confined by ℓ, ℓ', and C^T. Concerning the boundary, ℓ belongs to S, ℓ' does not belong to S, the portion on C^T delimiting S belongs to S, and $c(R)$ does not belong to S. $Sector(R)$ denotes the set containing all the sectors of a configuration R.*

We now exploit the sectorial distance to determine the trajectories used by robots to move toward the targets.

Definition V.5 (Safe trajectory)**.** *Given a configuration R and a sector $S \in Sector(R)$, a robot $r \in Rob(S)$ is said to admit a safe trajectory toward a target point $t \in S \cup c(R)$ if there exists a shortest path between r and t according to $dist()$ that does not pass through any other robot.*

The next statements (see Lemma V.6 and Proposition V.7) will play a central role for the definition of `Distmin`.

Lemma V.6. *Given a configuration R and a sector $S \in Sector(R)$, let $r \in Rob(S)$ and $t \in S$ be a target point. If $AS(r, t)$ is not degenerate, then r admits a safe trajectory toward t.*

Proof. The claim simply follows from Proposition V.3 that implies the existence of infinitely many shortest paths between r and t, and by observing that R is finite. □

Property V.7. *For each sector S, the sub-configuration given by $\partial C(R) \cup Rob(S)$ is asymmetric.*

The above statements can be combined as follows: the former ensures that when a robot r moves toward a target t and $AS(r, t)$ is not degenerate, then r admits a safe trajectory toward t; the latter says that inside a sector S it is always possible to elect a leader $r \in Rob(S)$. By combining them we get that inside a sector S we can always elect a robot r to move toward a target t, and if $AS(r, t)$ is not degenerate then r can move along a shortest path without creating collisions. Given a sector S, the following additional notation allow us to formalize such an approach:

- $R^m(S) = Rob(S) \cap F'$ denotes the matched robots;
- $F^m(S) = F' \cap R^m(S)$ denotes the matched targets;
- $R^{\neg m}(S) = Rob(S) \setminus R^m(S)$ denotes the unmatched robots;
- $F^{\neg m}(S) = (F' \cap S) \setminus F^m(S)$ denotes the unmatched targets;
- $R^{safe}(S) = \{r \in R^{\neg m}(S) : \exists$ a safe trajectory from r to t, $t \in F^{\neg m}(S)\}$ denotes the subset of $R^{\neg m}(S)$ containing only robots having a safe trajectory toward at least one target in $F^{\neg m}(S)$;

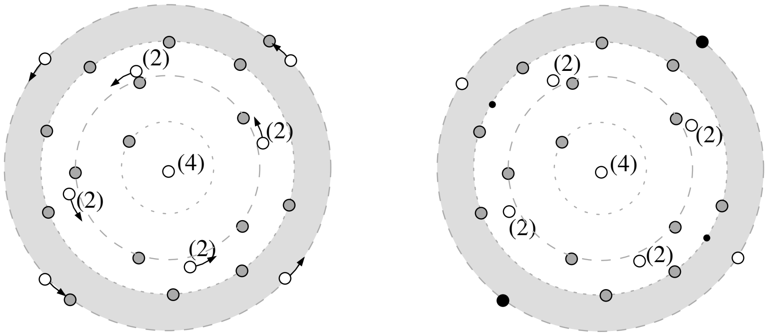


FIGURE 11: Task T_8. Make a partial pattern formation: embedding of F and F'. The light-gray corona is the Ann; gray circles represent robots; white circles represent points of F; On the left, arrows represent how F must be rotated according to the embedding defined in Section IV-A. On the right, black circles represent robots matched with points of F after the embedding; finally, the two black dots on C^T represent points of F' obtained as radial projections of unmatched points of F on $C(R)$.

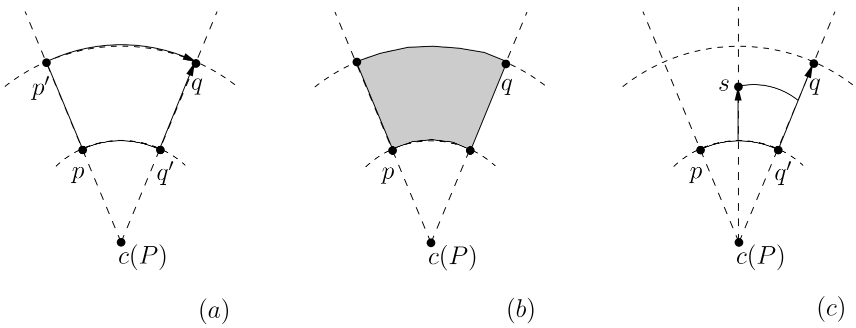


FIGURE 12: A representation of: (a) the sectorial paths between points p and q; (b) the annulus sector $AS(p, q)$ (the gray region); (c) two shortest paths from p to q (one passing through s and composed by two sectorial paths).

If in S both $R^{\neg m}(S) \neq \emptyset$ and $F^{\neg m}(S) \neq \emptyset$ then:

- $r^*(S)$ denotes the unmatched robot in S that has to move toward an unmatched target still in S. If $R^{safe}(S) \neq \emptyset$ then $r^*(S)$ is the robot of minimum view satisfying $\arg\min_{r \in R^{safe}(S)} \{dist(r, t) : t \in F^{\neg m}(S)\}$ else $r^*(S)$ is selected from $R^{\neg m}(S)$ according to the minimum view (cf. Proposition V.7).

Consider now the case in which there are more robots than targets within a sector S. Our approach will move one robot at a time in S (always identified as $r^*(S)$) toward a target in $F^{\neg m}(S)$ until all targets become matched. At that time, we will get $R^{\neg m}(S) \neq \emptyset$ and $F^{\neg m}(S) = \emptyset$. Then, our strategy will move the remaining robots in $R^{\neg m}(S)$ toward points on the robot-ray belonging to S', where S' is the next sector with respect to S according to the clockwise direction. We then extend the previous notation as follows:

- $R^{safe}(S, S') = \{r \in R^{\neg m}(S) : r$ is lying on a circle $C^i_\downarrow$ and it can rotate along $C^i_\downarrow$ until reaching S' without collisions$\}$ denotes the set containing any robot that can reach S' by means of a simple rotation along the circle $C^i_\downarrow$ where it lies;
- $r^*(S, S')$ denotes the unmatched robot in S that has to move toward S'. If $R^{safe}(S, S') \neq \emptyset$ then $r^*(S) = \arg\min_{r \in R^{safe}(S,S')} \{dist(r, t) : t \in S'\}$ else $r^*(S)$ is selected from $R^{\neg m}(S)$ according to the minimum view.

Procedure `Distmin` is given in Algorithm 2. Its description can be found in the corresponding correctness proof provided in Lemma V.8. Figure 13 provides a partial illustration of how `Distmin` determines the pairs robot-target within one sector of the running example.

Lemma V.8. *Given a configuration R belonging to $T_8 \cap (\mathcal{I} \setminus \mathcal{U}(F))$, by repeatedly applying Procedure* `Distmin` *the pattern F' can be formed.*

Proof. According to Proposition V.7, two robots with the same view cannot belong to a same sector $S \in Sector(R)$. Hence, all moves allowed by Procedure `Distmin` involve at most one robot per sector as ties are always broken by means of the minimum view.

Lines 1-3 consider the cases when the current multiplicity in the center $c(R)$ is less than that required in $c(F)$. Notice that $\rho(F) > 1$ by hypothesis, and this implies that in $c(F)$ there is a number of points which is multiple of $\rho(F)$. Since $\rho(R)$ divides $\rho(F)$, then the number of robots in each circle $C^i_\downarrow(R)$ divides $\rho(F)$, and hence the number of robots to be moved toward the center is always correctly determined by

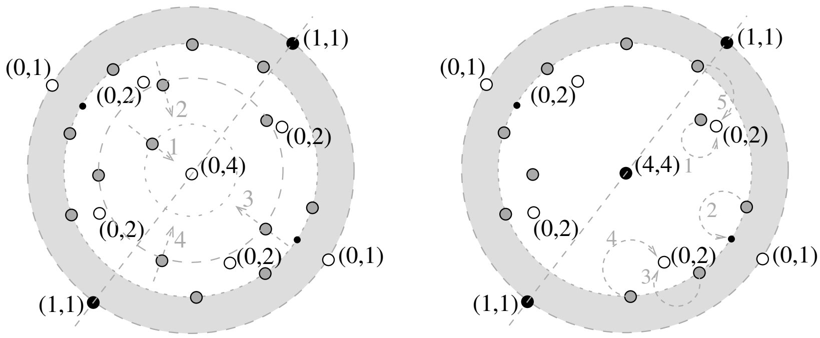


FIGURE 13: Task T_8. Any pair of integers close to points of F represents multiplicities of robots and of targets, respectively. (left) Preliminary phase, the right multiplicity is formed on $c(F)$ (cf. Lines 1-3 of Algorithm 2). The numbers close to the arrows show the order in which robots move. (right) Order of robots' movements toward targets within one sector. Notice that the gray arrows only show robot-target pairs and not trajectories: we recall that Algorithm 2 uses sectorial paths as robots' trajectories.

Algorithm 2 `Distmin`

1: **if** $mult(c(R), R) < mult(c(F), F)$ **then**
2: **if** $d(r, c(R))$ is minimum among all robots in R, and r is of minimum view in case of ties **then**
3: r moves toward $c(R)$
4: **else**
5: **if** $\exists$ sector S s.t. $R^{\neg m}(S) \neq \emptyset$ and $F^{\neg m}(S) \neq \emptyset$ **then**
6: **if** $R^{safe}(S) \neq \emptyset$ **then**
7: $r^*(S)$ moves toward its target $f \in F^{\neg m}(S)$ along a safe trajectory
8: **else**
9: **if** $r^*(S)$ and its target f belong to a circle $C^i_\downarrow(R)$ **then**
10: $r^*(S)$ moves radially at half distance from $C^{i-1}_\downarrow(R)$ if this exists or from $c(R)$
11: **else**
12: $r^*(S)$ rotates clockwise at half distance from the closest robot-ray or from the closest robot if there is one on the way
13: **else**
14: **if** $\exists$ sector S s.t. $R^{\neg m}(S) \neq \emptyset$ **then**
15: Let S' be the next sector in clockwise order;
16: **if** $R^{safe}(S, S') \neq \emptyset$ **then**
17: $r^*(S, S')$ rotates toward the robot-ray of S'
18: **else**
19: Let $C^i_\downarrow(R)$ be the circle to which $r^*(S, S')$ belongs to
20: $r^*(S, S')$ moves radially at half distance from $C^{i-1}_\downarrow(R)$ if this exists or from $c(R)$
21: **else**
22: Let r be the robot on $c(R)$: r radially moves along the segment connecting $c(R)$ with the unique point left in $F^{\neg m}(S)$ for some sector S, until distance $\delta(C^B)$;

the procedure: this is $\rho(R)$ which divides $\rho(F)$ that in turn divides the number of robots in $c(F)$.

Lines 5-22 consider the cases when the multiplicity in the center (if any) is already correctly formed. In particular, lines 5-12 are executed when there exists a sector S in which there are both unmatched robots and unmatched targets. According to our definitions, the robot $r^*(S)$ elected to move follows a safe trajectory if it exists. Once this robot starts moving, it will be moved until reaching its target, possibly within multiple LCM cycles. In fact, (1) robots admitting safe trajectories move before robots not admitting safe trajectories, and (2) the moves along safe trajectories assure to decrease the distances to the target; hence, in case of multiple LCM cycles, the moving robot $r^*(S)$, for each sector S, will be again chosen to reach its target. In case there is not a safe trajectory from $r^*(S)$ to the target, then the robot is slightly deviated (see moves at Lines 10 and 12) to avoid collisions. Then, by Lemma V.6, the deviated robots admit safe trajectories and will be chosen again by the algorithm to be moved.

Once each sector contains only unmatched robots or only unmatched targets, then unmatched robots in any sector S are moved toward a point on the boundary of the next sector S' in clockwise order (cf. Lines 14-20). As before, robots moved are first those elected that admit a safe trajectory toward the next (clockwise) sector, and then the remaining ones (which are deviated as before in order to avoid collisions). Notice that, as soon as the moved robot reaches the boundary, it enters into the next sector S'. As a consequence, the procedure processes this robot when it will be elected in S' to be moved either toward an unmatched target in the same sector, or toward the boundary of the successive (clockwise) sector.

The last line (Line-22) consider the cases when a robot must be moved from the center $c(R)$ whereas any other robot is matched. This case is processed at the end because, by definition, the center $c(R)$ does not belong to any sector. The robot is moved toward the last unmatched target in F' until reaching the circle C^B (by definition, along the trajectory there are no targets and hence no robots). Regardless whether it is stopped or not by the adversary, once it becomes active again, it will be processed as an unmatched robot by Lines 5-12. $\square$

G. TASK T_9

This task is devoted to finalize the pattern formation. It is characterized by the precondition $\mathtt{pre}_9 = \neg\mathtt{m} \wedge \mathtt{p}$, which means: there is a subset of $m \geq 2$ robots on $C(R)$ that form a regular m-gon, with m divisor of $\rho(F)$; the unmatched robots with respect to F are only those in Ann or on C^T; F can be obtained by radial movements of the unmatched robots toward $C(R)$.

Move m_9 makes such robots moving radially toward $C(R)$. As described in Section IV-A (cf. description of Fin), while robots move from C^T to $C(R)$, the common reference system might be lost as soon as some robots reaches $C(R)$. However, robots can always detect whether the configuration obtained by a radial projection of all robots in $Ann \cup C^T$ to $C(R)$ produces F or not as both Ann and C^T can be determined just on the basis of F. This is the way to establish the value of variable $\mathtt{p}$. Trivially, once all robots finish their movements, $\mathtt{w}$ becomes true, that is F is formed. Figure 14 provides an illustration of this task when it is applied to the running example.

H. TASK T_{11}

This is actually not a real task. It is identified by variable $\mathtt{w}$ which means F is formed, hence robot must not move anymore. It guarantees the obtained configuration does not change anymore.

I. TASK T_5

This task is complementary with respect to T_3 and T_4 as it is invoked when the number m of robots on $C(R)$ is too small with respect to $\rho(F)$, that is m is smaller than the minimal prime factor of $\rho(F)$. In this case, the configuration is necessarily asymmetric and, consequently, one robot per time is moved from $C^2_{\downarrow}(R)$ toward $C^1_{\downarrow}(R) = C(R)$ by means of move m_5. Robots are moved toward $C(R)$ avoiding forbidden points for $C(R)$. These forbidden points are similar to those introduced in the description of Task T_2: a point of $C(R)$ is forbidden if it may form a regular n-gon along with the points occupied by some robots already located on $C(R)$. Again, avoiding forbidden points ensures that when a robot reaches $C(R)$ all robots in such a circle are non-equivalent; this helps to ensure that no unsolvable configurations are created.

An example of application of m_5 can be seen in Figure 15. There $\rho(R) = 1$ whereas $\rho(F) = 5$. Moreover, $|\partial C(R)| = 2$ is smaller than the minimal prime factor of $\rho(F)$, which is five. So, $\mathtt{f} = \mathtt{true}$, whereas $\mathtt{c} = \mathtt{false}$. The configuration is then in T_5 as it can be easily checked: $\mathtt{w}$, $\mathtt{g}$, $\mathtt{p}$ are false, that is the configuration does not belong to T_{11}, T_{10}, T_9, respectively; $\mathtt{d}_1$ and $\mathtt{d}_2$ are true, hence R is not in T_8 nor in T_7; $\mathtt{t} = \mathtt{false}$ since 2 is not a divisor of $\rho(F)$, hence R is not in T_6.

Once three robots, one per time, are moved to $C(R)$ by means of m_5, the configuration in Figure 16, right side, is obtained. It belongs to T_7 as a regular 5-gon must be formed on $C(R)$ since $\mathtt{d}_1$ and $\mathtt{d}_2$ are now false as well as $\mathtt{u}$.

J. TASK T_7

This task is meant to create a regular m-gon on $C(R)$. It is a sort of generalization of T_6 as it is used when $m = |\partial C(R)|$ is the minimal prime factor of $\rho(F)$, that is $\mathtt{d}_1$ and $\mathtt{d}_2$ are both false. By means of move m_7 the m robots on $C(R)$ are opportunely rotated so as to obtain a regular m-gon. Once this happens, $\mathtt{m}$ becomes false and $\mathtt{u}$ becomes true. Actually, Procedure `CircleForm()` applies the same movements of the algorithm proposed in [17] where the problem was to uniformly distribute robots along a ring. The only difference is that here the ring is the circumference of $C(R)$, hence to guarantee the correctness of the algorithm we need to guarantee that $C(R)$ does never change.

Algorithm 3 `CircleForm`(α)

1: Let r', r and r'' be three consecutive (clockwise) robots on $C(R)$;
2: Let p be the antipodal point of r';
3: Let q be the point on $C(R)$ preceding r'' wrt the clockwise direction such that $\sphericalangle(q, c(R), r'') = \alpha$;
4: **if** $\sphericalangle(r, c(R), r'') > \alpha$ **then**
5: $\quad$ r rotates clockwise toward the closest point among p and q;

Given a configuration R, with $|R| \geq 3$, let $\alpha = 2\pi/|\partial C(R)|$, and let r', r and r'' be three consecutive (clockwise) robots on $C(R)$. The following lemma can be stated.

Lemma V.9. *Let p and q be the two points calculated by a robot r when running algorithm* `CircleForm`(α). *If r has to move, it will reach q, within a finite number of LCM cycles.*

Proof. If p is not in between r and q then the statement clearly holds as all moving robots follow the clockwise direction, and hence within different LCM cycles the target to reach either is unchanged or it is further (clockwise) than q with respect to the starting position of r, that is r reaches (and possibly overpasses) q.

When p is in between r and q then r must stop at p, and eventually r reaches p. In this case, since $|R| \geq 3$, necessarily $\sphericalangle(r', c(R), r) > \alpha$, that is, also robot r' must move.

Consider the points q' and q'' on $C(R)$ antipodal to q and r'', respectively. When r' moves, it cannot overpass q'', however, by construction, once r has reached p, then q' is met by r' before reaching q''. It follows that as soon as r' reaches q' then r is free to reach q. $\square$

By combining the result of Lemma V.9 with the correctness proof of the Circle Formation algorithm given in [17], the following corollary holds.

Corollary V.10. *Let R be a configuration belonging to $T_7 \cap \mathcal{I}_{\mathbb{A}}$ with m robots on $C(R)$. By repeatedly applying Algorithm* `CircleForm`*, configuration R is transformed into a configuration R' having a regular m-gon on $C(R)$.*

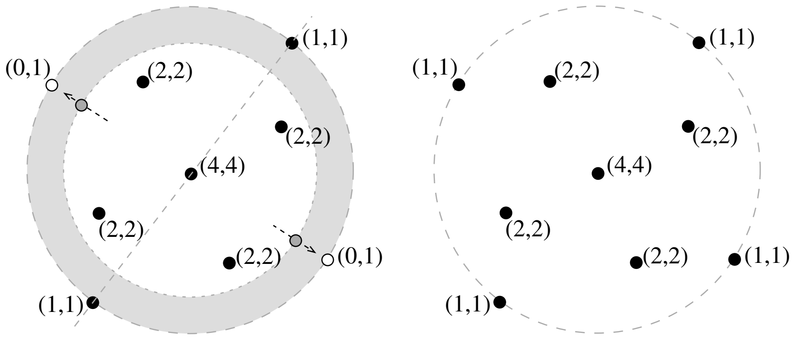


FIGURE 14: Task T_9: finalize the pattern formation.

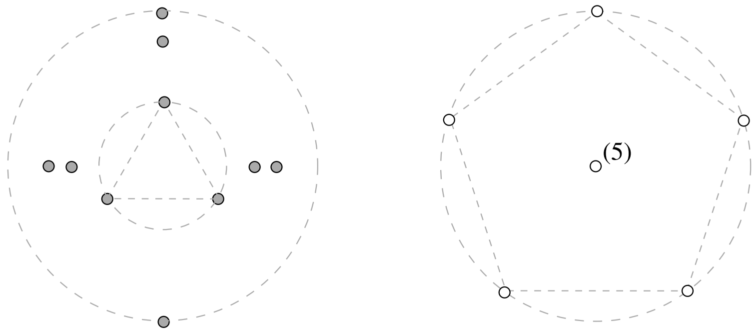


FIGURE 15: The input for the PF problem that we use as secondary running example. Notice that the initial configuration R is composed of 10 robots and $rho(R) = 1$, while the pattern F has symmetricity $\rho(F) = 5$ (numbers close to points refer to multiplicities).

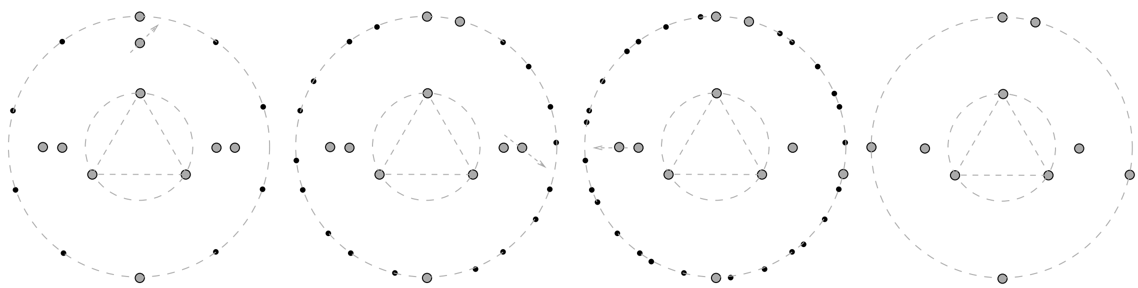


FIGURE 16: Task T_5 applied to the input specified in Figure 15: bring robots to $C(R)$ until $|\partial C(R)|$ divides $\rho(F)$. Gray circles represent robots and small black dots represent forbidden points for $C(R)$.

Proof. The proof simply follows by observing that algorithm `CircleForm` operates the same movements of those in [17] but with the further constraint to not changing $C(R)$. However, Lemma V.9, proves that eventually each moving robot will reach the destination imposed in [17]. It means that within multiple (but finite) LCM cycles, each moving robot behaves like in [17]. □

Considering the running example of Figure 17 (left side), the robots on $C(R)$ are opportunely rotated in the clockwise direction so as to obtain a configuration with a regular pentagon on $C(R)$. In particular, the only robot that will never move in the specific configuration is the top-most one since the angle it forms in $c(R)$ with the clockwise neighbor is smaller than $\frac{2\pi}{5}$. All other robots, will rotate eventually. Once configuration in Figure 17 (right side) is obtained, it means $\mathtt{u} = \mathtt{true}$. Predicates $\mathtt{g}$, $\mathtt{a}$ and $\mathtt{d}_2$ did not change their values, whereas variables $\mathtt{w}$ and $\mathtt{p}$ are clearly false. Hence the configuration cannot belong to T_9, T_{10} and T_{11}. Since $\neg \mathtt{d}_2 \Rightarrow \neg \mathtt{d}_1$, then the configuration belongs to T_8.

K. TASK T_{10}

This is actually not a real task. It solves PF by exploiting other algorithms (namely `Gathering` from [10] and `Leader` from [6]) when F is composed of one point with multiplicity $|R|$, that is $\rho(F) = |R|$, or when $\rho(F) = 1$, respectively. Notice that P_{10} depends only on F and not on the current

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3089081, IEEE Access

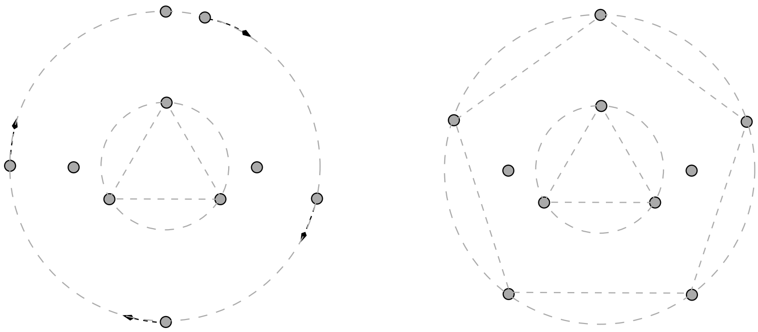


FIGURE 17: Task T_7. Create a regular m-gon on $C(R)$.

configuration. This implies that once one algorithm among Gathering and Leader starts, it will be invoked to process the configuration until the pattern is formed.

VI. CORRECTNESS

In this section, we prove the correctness of the provided algorithm. According to the methodology proposed in [8], it is realized by proving that each property in Table 4 holds. It is worth noting that properties $H_{3'}$ and $H_{3''}$ are desirable but not necessary to prove the correctness of the algorithm.

Concerning property H_1, since the tasks' predicates $P_1, P_2, \ldots, P_{11}$ used by the algorithm have been defined as suggested by Equation 1, it holds according to Remark 1.

Since properties H_2, H_3, $H_{3'}$, $H_{3''}$ and H_4 (the last limited to self-loops only) must be proved for each transition / move, then in the following we provide a specific lemma for each task. It is worth pointing out that, according to Remark IV.7, if one of such lemmas analyzes a task - say T_i - and we have already proved that all the transitions toward T_i are stationary or almost-stationary or robust, then during the analysis of T_i we can basically ignore possible pending moves. A final theorem (cf. Theorem VI.12) will make use of all these lemmas and will also prove the remaining part of property H_4 concerning cycles that are not self-loops. As last remark, we remind that properties $H_{3'}$ and $H_{3''}$ are desirable but not necessary to prove the correctness of the algorithm. As we are going to see, in a few cases we cannot guarantee them.

Lemma VI.1. *Let R be a stationary configuration in T_{10}. From R the algorithm eventually leads to a stationary configuration belonging to T_{11}.*

Proof. Since g holds, we have two cases: either $\rho(F) = 1$ or F contains only one element with multiplicity $|R|$. In the first case move m_{10} consists in calling the Leader() algorithm given in [6]. In the second case, move m_{10} consists in applying the algorithm Gathering() given in [10]. Since the predicate only depends on F, its value never changes then one of the two algorithms can be applied until forming pattern F. Concerning the correctness of the algorithms we refer the reader to the proofs given in [10] and [6], respectively. □

Remark VI.2. *As* g *only depends on F and not on the current configuration, from now on we can always consider variable* g *as false since the movements of robots cannot change its value. It also follows that no transitions can lead to T_{10} apart for self-loops.*

Lemma VI.3. *Let R be a stationary configuration in T_9. From R the algorithm eventually leads to a stationary configuration belonging to T_{11}.*

Proof. Move m_9 aims to finalize the pattern formation by performing only radial movements of robots from $C^T \cup Ann$ to $C(R)$.

- H_2: During this task, since move m_9 does not remove any robot from $C(R)$, then $C(R)$ does not change and $\neg$m remains true. Moreover, since the movement is radial and by the fact that the computation of C^T depends only on F, p remains true during all the movements. Similarly, w remains false until the last robot reaches $C(R)$. This means that it is always possible to solve PF when $\neg$m$\wedge$p holds. It is enough to radially move all robots from $C^T \cup Ann$ to $C(R)$ (which is exactly what move m_9 does). Hence, independently on the activation of the robots, the incurred configurations until F is formed are all solvable, that is none of them belongs to $\mathcal{U}(F)$.
- H_3: as observed, during the move $\neg$m, p and $\neg$w remain true, then no other tasks can start. If robots are stopped during their movement by the adversary, the configuration remains in T_9. This defines a self-loop in T_9. If all robots involved by move m_9 reach their target on $C(R)$ then w becomes true and the configuration is in T_{11}.
- $H_{3'}$: If the configuration remains in T_9 after applying move m_9, the set of robots involved by the move as well as their trajectories do not change, hence the self-loop of T_9 is almost-stationary. Once all the robots in $C^T \cup Ann$ reach $C(R)$ (that is F is formed and the configuration is in T_{11}) the configuration is stationary.
- $H_{3''}$: actually two robots on the same ray can potentially collide, but this is not a problem as at their destination

TABLE 4: Properties underlying the correctness

H_1	=	*for each configuration in $\mathcal{I}_{\mathbb{A}}$ at least one predicate P_i is true and for each $i \neq j$, $T_i \cap T_j = \emptyset$;*
H_2	=	*configurations in $\mathcal{U}(F)$ are not generated by $\mathbb{A}$, i.e. $\mathcal{I}_{\mathbb{A}} \cap \mathcal{U}(F) = \emptyset$ - this means that given R and F as input, each generated configuration $R(t)$, $t > 0$, must ensure that $\rho(R(t))$ divides $\rho(F)$;*
H_3	=	*for each class T_i, the classes reachable from T_i by means of a transition are exactly those represented in the transition graph G* (i.e., the transition graph is correct);
$H_{3'}$	=	*each transition not leading to T_{11} is stationary, almost-stationary, or robust, while each transition leading to T_{11} is stationary;*
$H_{3''}$	=	*the algorithm is collision-free;*
H_4	=	*possible cycles in the transition graph G (including self-loops but excluding the self-loop in T_{11}) must be performed a finite number of times.*

there must be a multiplicity, as p holds.[6]

H_4: when the self-loop is traversed, the overall distance of the robots involved by move m_9 to $C(R)$ is decreased. Then, eventually, it becomes zero and all such robots will be on $C(R)$.

□

Lemma VI.4. *Let R be a stationary configuration in T_8. From R the algorithm eventually leads to a solvable and stationary configuration belonging to T_9 or T_{11}.*

Proof. The aim of the task is to form pattern F' rather than F. This is done so as the embedding of F on R is maintained thanks to the k-gon on $C(R)$ (cf. description of sub-problems RS and PPF of Section IV-A). Note that $\neg \mathtt{d}_1 \wedge \mathtt{u} \Rightarrow \neg \mathtt{m}$, hence p must be false as otherwise the configuration would be in T_9.

H_2: during the movements, as robots in $C(R)$ remain unchanged (and so $C(R)$ itself), $\rho(R)$ can be at most $|\partial C(R)|$ or a divisor of it. Being $|\partial C(R)|$ a divisor of $\rho(F)$ (since $\neg \mathtt{m}$ holds), then no unsolvable configurations with respect to the symmetricity (cf. Theorem III.3) can be generated. Moreover, by Lemma V.8, if move m_8 leads to create a multiplicity, this is on a point corresponding to a multiplicity in F', and also its size would not be greater than that specified by F'. Hence, no unsolvable configurations are created on this respect as well.

H_3: during the movements, m_8 does not change the values of w, m, and $\mathtt{d}_1$, that are `false`, and those of a and u, that are `true`, as no robots are moved neither toward nor from $C(R) \cup Ann$. Moreover, by definition p remains false until the last robot reaches its destination, that is once F' is formed. The configuration is then always in T_8 until F' is formed. As soon as the last robot reaches its destination, the configuration satisfies p. Hence, if F' is different from F (in case there are robots on C^T), then the configuration is in T_9, otherwise the configuration is in T_{11}.

$H_{3'}$: except for the robots moved by `Distmin`, no other robot is moved (the only possible ones are those on $\partial C(R)$, not affected by m_8), then, when p holds the configuration is stationary. Whereas, if the configuration remains in T_8 after applying move m_8 and it is non-stationary, the set of robots involved by the move does not change but their trajectories could. This may happen when robots deviate to avoid collisions. Hence, the self-loop in T_8 is robust.

$H_{3''}$: by Lemma V.8, procedure `Distmin` avoids collisions.

H_4: if a moving robot is stopped by the adversary during its movement, the configuration remains in T_8 and the robot will be moved again. By Lemma V.8, the total distance of the robots from their target decreases. Hence, the self-loop of T_8 can be traversed only a finite number of times.

□

The next lemmas refer to the RS subproblem, that is to tasks T_1, T_2, …, T_7. All those tasks operates on configurations in $\mathcal{I} \setminus \mathcal{U}(F)$, that is solvable configurations without multiplicities, and as we are going to show each of them generates a configuration in $\mathcal{I} \setminus \mathcal{U}(F)$. Non-initial configurations are instead managed only by tasks T_8, T_9, T_{10}, T_{11} and, as shown in the above lemmas, they never generate configurations in T_1, T_2, …, T_7.

Lemma VI.5. *Let R be a stationary configuration in $T_7 \cap (\mathcal{I} \setminus \mathcal{U}(F))$. From R the algorithm eventually leads to a stationary configuration in $\mathcal{I} \setminus \mathcal{U}(F)$ belonging to T_8, T_9 or T_{11}.*

Proof. Let $k = |\partial C(R)|$ be the minimal prime factor of $\rho(F)$. Then $\neg \mathtt{d}_2$ holds and this implies that $\neg \mathtt{d}_1$ holds too. The k robots on $C(R)$ are rotated by m_7 which applies Procedure `CircleForm` so as to obtain a regular k-gon

[6] We remind that property $H_{3''}$ - as well as $H_{3'}$ - are desirable but not necessary. As we are going to prove, the current case is actually the only one where property $H_{3''}$ might be violated.

without affecting $C(R)$. Once this happens, m becomes false and u becomes true.

H_2: as $k = |\partial C(R)|$ is the minimal prime factor of $\rho(F)$, then k is prime. This implies either $\rho(R) = 1$ or $\rho(R) = k$. This last possibility can happen only at the end of this task when u becomes true, whereas $\rho(R) = 1$ for each generated configuration R during the task. Moreover, as Procedure `CircleForm` guarantees to not create multiplicities, then no unsolvable configurations can be generated.

H_3: the move only involves robots in $C(R)$ along $C(R)$, hence a, that is `true`, and $\mathtt{d}_2$, that is `false` do not change their values. Variable w can become true only once the k-gon is formed. Similarly $\neg \mathtt{m} \wedge \mathtt{p}$ and u remain false as long as the k-gon is not formed. Hence, if robots are stopped during their movements, the configuration remains in T_7. Once the k-gon is formed then m becomes `false` and u becomes `true`. Since $\neg \mathtt{d}_1$ holds, this implies that the configuration can be in T_8 (not in T_1, T_2, T_3, T_4, T_5, T_6), in T_9, or in T_{11} according to possible changes of the values of p and w.

$H_{3'}$: at the end of the task the configuration is clearly stationary as the only robots allowed to move are those on $C(R)$ and they do not move once u holds. If the configuration remains in T_7 after applying move m_7, the trajectory of a moving robot might be prolonged but always along the circumference of $C(R)$. Hence, the self-loop in T_7 is almost-stationary.

$H_{3''}$: in Procedure `CircleForm` no collisions are possible because the target of a move is always between the moving robot and the next (clockwise) robot on $C(R)$.

H_4: the correctness of Procedure `CircleForm` provided in Corollary V.10 guarantees the property.

□

Lemma VI.6. *Let R be a stationary configuration in $T_6 \cap (\mathcal{I} \setminus \mathcal{U}(F))$. From R the algorithm eventually leads to a stationary configuration in $\mathcal{I} \setminus \mathcal{U}(F)$ belonging to T_3 or T_9.*

Proof. There are exactly three robots on $C(R)$ (as $\mathtt{t} = \mathtt{true}$) and $\mathtt{w} = \mathtt{false}$. Note that $\rho(R) = 1$, otherwise, if $\rho(R) = 3$ (and then 3 is a divisor of $\rho(F)$) the configuration would not be in T_6 (it would be in T_8, because in this case $\neg \mathtt{d}_1 \wedge \mathtt{u}$ holds). Moreover $\rho(F)$ must be even as t holds. By referring to the description of move m_6 note that $\alpha_1 \neq 90°$ as otherwise r_2 and r_3 are antipodal, against m. Moreover, $\alpha_1 < 90°$ as otherwise the three robots would lie in half $C(R)$ hence defining a different smallest enclosing circle. Being $\rho(R) = 1$, the configuration is asymmetric and hence robot r_2 can always be selected and moved toward its target without modifying $C(R)$.

The configuration can start with an equilateral triangle on $C(R)$ (when three is not a divisor of $\rho(F)$), but as soon as r_2 moves, u is false and remains false until the end of the task.

H_2: since during this task $\rho(R) = 1$ and no multiplicities are created, no unsolvable configurations can be generated.

H_3: during the movement (i.e., before reaching the target), the variables involved in $\mathtt{pre}_6$ do not change their values. Hence the configuration cannot be in T_9 because of m. It cannot be in T_8 because of u. It cannot be in T_7 because of $\mathtt{t} \Rightarrow \mathtt{d}_2$. Then the configuration remains in T_6 until the moving robot reaches the target. At that point, m becomes false. If p is also true then the configuration is in T_9. By the same considerations as above, the obtained configuration cannot be in T_8 nor in T_7. It is not in T_6 nor in T_4 because of m. It is not in T_5 because of f. Hence, it is in T_3 since $\mathtt{pre}_3$ holds.

$H_{3'}$: the transitions to the tasks following T_6 are obviously stationary being r_2 the only moving robot. Whereas the self-loop is almost-stationary as the same robot along the same trajectory is moved at any time.

$H_{3''}$: by the definition of move m_6 no collision can be generated by r_2.

H_4: the possible self-loops of this task will end as the total distance of the robot from its target decreases.

□

Lemma VI.7. *Let R be a stationary configuration in $T_5 \cap (\mathcal{I} \setminus \mathcal{U}(F))$. From R the algorithm eventually leads to a stationary configuration in $\mathcal{I} \setminus \mathcal{U}(F)$ belonging to T_2 or T_7.*

Proof. At the beginning the configuration is necessarily asymmetric, that is $\rho(R) = 1$, because the number of robots on $C(R)$ is less than the minimal prime factor of $\rho(F)$, being $\mathtt{f} = \mathtt{true}$. Hence one robot per time is moved from $C^1_\downarrow(R)$ toward $C(R)$ by means of move m_5. In general, the movements are radial toward $C(R)$. Deviations are applied if the move may cause a collision on $C(R)$ or may potentially make the configuration symmetric. As alternative target we may consider the closest middle point in the clockwise direction between two consecutive forbidden points. In any case, $C(R)$ remains unchanged.

H_2: since during this task $\rho(R) = 1$ is guaranteed by avoiding forbidden points for $C(R)$, hence avoiding also to create multiplicities, no unsolvable configurations can be generated.

H_3: during the movement of the robot $\mathtt{c} = \mathtt{false}$, whereas both f and m are `true`; variable $\mathtt{w} = \mathtt{false}$ and variables both $\mathtt{d}_1$ and $\mathtt{d}_2$ are `true`. Moreover $\mathtt{t} = \mathtt{false}$ since 2 is not a divisor of $\rho(F)$. Then the configuration cannot be in any task from T_6 to T_{11}, so any configuration generated during the movement remains in T_5. Once the last robot reaches $C(R)$, variable f becomes false. The obtained configuration cannot belong to T_{11} because of variables w, It cannot belong to T_9 and T_8 because of m and u, respectively, as moving robots avoided forbidden points for $C(R)$. If $\mathtt{a} = \mathtt{true}$, it belongs to T_7 since both $\mathtt{d}_2$ and u are

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3089081, IEEE Access

false, otherwise it belongs to T_2 since it cannot belong to T_3, T_4, and T_6 being $\mathtt{a} = \mathtt{false}$.

$H_{3'}$: the transitions to the tasks following T_5 are obviously stationary because there is only one moving robot per time. Whereas the self-loop is robust as the same robot will be moved but its target may change because of deviations to avoid forbidden points for $C(R)$.

$H_{3''}$: by the definition of move m_5 there is no robot between the moving robot and its target, then no collision can be generated.

H_4: the possible self-loops of this task will end as the total distance of the robots from $C(R)$ decreases.

□

Lemma VI.8. *Let R be a stationary configuration in $T_4 \cap (\mathcal{I} \setminus \mathcal{U}(F))$. From R the algorithm eventually leads to a stationary configuration in $\mathcal{I} \setminus \mathcal{U}(F)$ belonging to T_6, T_7, or to a robust configuration in $\mathcal{I} \setminus \mathcal{U}(F)$ belonging to T_2.*

Proof. In this task $\mathtt{pre}_4 = \mathtt{a} \wedge \neg\mathtt{c} \wedge \mathtt{m}$ holds. Since $\rho(R)$ divides $\rho(F)$ by hypothesis, $\mathtt{c} = \mathtt{false}$ and $\mathtt{m} = \mathtt{true}$ imply that the current configuration R is asymmetric. Moreover, according to the way predicates are defined, all preconditions concerning tasks T_5, T_6, $\ldots$, T_{11} are false. In particular, this implies the following properties:

- being $\mathtt{c} = \mathtt{false}$, from $\mathtt{pre}_5 = \mathtt{false}$ we derive $\mathtt{f} = \mathtt{false}$: this means that on $C(R)$ there is a number of robots greater than or equal to the minimal prime factor of $\rho(F)$.
- being $\mathtt{a} = \mathtt{true}$, from $\mathtt{pre}_7 = \mathtt{false}$ and $\mathtt{pre}_8 = \mathtt{false}$ we derive that at least one variable among $\mathtt{d}_1$ and $\mathtt{d}_2$ must be true. This means that on $C(R)$ there is a number of robots which is not equal to the minimal prime factor of $\rho(F)$.

By combining the previous properties, we know that on $C(R)$ there is a number of robots greater than the minimal prime factor of $\rho(F)$.

According to move m_4, the algorithm removes one robot at a time from $C(R)$ (without affecting $C(R)$ by opportunely removing non-critical robots) until exactly p robots remain, where p is the minimal prime factor of $\rho(F)$.

H_2: according to Procedure $\mathtt{GoToC^T}$, the robot r on $C(R)$ of minimal view is straightly moved toward a suitable point on C^T. By similar arguments applied in the proof of Lemma V.1, such a movement maintains the configuration asymmetric, that is its symmetricity equals one. Moreover, no multiplicities are created and hence no unsolvable configurations are generated.

H_3: as soon as r starts moving, $\mathtt{a}$ becomes false. We can distinguish two cases: either r reaches its target on C^T or it stops before.
When r reaches its target on C^T, each variable referring to $C(R)$ can be potentially influenced. Among those, certainly $\mathtt{d}_1$ and $\mathtt{d}_2$ can change; $\mathtt{f}$ cannot change and hence it remains false; $\mathtt{t}$ and $\mathtt{u}$ can change; $\mathtt{m} = \mathtt{true}$ and does not change; $\mathtt{w}$ cannot change. Consequently, no configurations in T_{11} nor in T_9 can be generated because of $\mathtt{m}$. Concerning T_8, notice that the following implication holds $\neg\mathtt{d}_1 \wedge \mathtt{u} \Rightarrow \neg\mathtt{m}$. Hence, since $\mathtt{m} = \mathtt{true}$ in R and it does not change its value, then no configurations in T_8 can be generated (P_8 requires $\neg\mathtt{d}_1 \wedge \mathtt{u} = \mathtt{true}$). If $\mathtt{d}_2$ becomes false, then task T_7 must be applied so as to evenly distribute robots on $C(R)$, hence making variable $\mathtt{u}$ true. This is due to the fact that $\mathtt{m}$ is false along the whole task. If $\mathtt{t}$ becomes true, then task T_6 must be applied as 3 would not be the minimal prime factor of $\rho(F)$ and $\mathtt{m} = \mathtt{true}$, that is there are no antipodal robots on $C(R)$. T_5 cannot be reached as $\mathtt{f}$ cannot change and hence it remains false. If nothing changes, still task T_4 is applied.
When r does not reach its target on C^T (i.e., it is stopped by the adversary inside Ann), $\mathtt{a}$ becomes false. It can be easily observed that in this case only task T_2 can be reached.

$H_{3'}$: the transitions to tasks T_6 and T_7 as well as the self-loop are obviously stationary because there is only one moving robot per time which has to reach its target. Whereas the transition to T_2 is robust as the same robot will be moved by m_2 but its target may change because of deviations to avoid forbidden points for C^T.

$H_{3''}$: collisions cannot occur according to Procedure $\mathtt{GoToC^T}$.

H_4: the repeated application of m_4 eventually ends as the number of robots in $\partial C(R)$ decreases opportunely.

□

Lemma VI.9. *Let R be a configuration in $T_3 \cap (\mathcal{I} \setminus \mathcal{U}(F))$. From R the algorithm eventually leads to a stationary configuration in $\mathcal{I} \setminus \mathcal{U}(F)$ belonging to T_8 or to a configuration in $\mathcal{I} \setminus \mathcal{U}(F)$ belonging to T_2.*

Proof. In this task $\mathtt{a} \wedge \neg\mathtt{c}$ holds and all preconditions concerning tasks T_4, T_5, $\ldots$, T_{11} are false. This means that from $\mathtt{pre}_3 \wedge \neg\mathtt{pre}_4$ it follows $\mathtt{m} = \mathtt{false}$. That is, on $C(R)$ there exists a maximal set of k robots regularly disposed, such that k divides $\rho(F)$. On $C(R)$ there must be more than k robots as otherwise being $\mathtt{m}$ true, $\neg\mathtt{d}_1 \wedge \mathtt{u}$ would be true as well and the configuration is instead in T_8. The aim of the move is to keep on $C(R)$ only k robots forming a regular k-gon (hence $C(R)$ is unchanged) and this is realized by means of Procedure $\mathtt{GoToC^T}$ that moves robots from $C(R)$ to C^T. According to move m_3, at most $\rho(R)$ robots per time can move.

H_2: according to m_3, the robots on $C(R)$ that should move are those of minimum view chosen among the set $\partial C(R) \setminus \mathcal{M}'$ if this is not empty, otherwise all robots on $C(R)$ of minimal view are chosen. The selected robots are straightly moved toward suitable points on C^T. As the move is basically the same applied in T_2 but involving robots from $C(R)$, similar arguments of the proof of Lemma V.1 guarantee to maintain the symmetricity of the configuration equal to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3089081, IEEE Access

a divisor of k along all the movement. Since k divides $\rho(F)$ and since no multiplicities are created, then no configuration in $\mathcal{U}(F)$ can be generated.

H_3: as soon as robots from $C(R)$ start moving, $\mathtt{a}$ becomes false. We can distinguish three cases: 1) all the active robots involved by move m_3 reach their targets on C^T; 2) some of them do not reach their target but all of them start moving; 3) some of them have performed the `Look` phase but did not start moving yet.
In case 1, if no variable changes its value, still task T_3 is applied. Otherwise, being the targets of the moving robots on C^T, then $\mathtt{w} = \mathtt{false}$. Moreover, similarly to what is shown in the proof of Lemma V.1, $\mathtt{p}$ remains false as well because of the limit imposed by angle α established when calling $\mathtt{GoToC^T}$. Differently from m_2 now robots start moving from $C(R)$ which potentially may affect the definition of angle α. However, since in m_3 only robots with the same minimum view can move concurrently, then they could not have been consecutive on $C(R)$ when T_3 started. This would in fact imply that all robots on $C(R)$ were equivalent, i.e. $\mathtt{u}$ was true. Since $\mathtt{m}$ was false, then also $\mathtt{d}_1$ would have been true, but then the configuration was in T_8 rather than in T_3.
Hence, the configuration is not in T_9 nor in T_{11}. If $\mathtt{d}_1$ becomes false, then the configuration might belong to T_8 if $\mathtt{u}$ is true. Whereas if $\mathtt{u}$ is false, it does not belong to T_8 nor to T_7 because $\mathtt{d}_1 \Rightarrow \mathtt{d}_2$. Variable $\mathtt{m}$ cannot change its value and it is false, that is the configuration cannot belong to T_6 nor to T_4. It does not belong to T_5 because of $\mathtt{f}$.
In case 2, some robots are still inside Ann, hence $\mathtt{a}$ becomes false and task T_2 is invoked.
In case 3, some robots might be still inside Ann in which case task T_2 is invoked. Whereas if Ann is empty then task T_3 is still applied because more than k robots are on $C(R)$, that is $\neg\mathtt{d}_1 \wedge \mathtt{u}$ is false.

$H_{3'}$: the reached configuration is stationary if all robots reach C^T (i.e. the configuration belongs to T_8). Otherwise there might be robots on $C(R)$ or in Ann concerning pending moves that will reach a suitable target on C^T, possibly computed from a different task and/or from a different configuration. By Lemma V.1, we have that the transition to T_2 or even the self-loop are unclassified. This is due to the fact that when such transitions occur, there might be robots that have decided to move while they wouldn't have moved from the current configuration, or they would have moved with respect to a different trajectory.

$H_{3''}$: collisions cannot occur according to Procedure $\mathtt{GoToC^T}$.

H_4: the repeated application of m_3 eventually ends as the number of robots in $\partial C(R)$ decreases until leaving a single regular k-gon.

□

Lemma VI.10. *Let R be a configuration in $T_2 \cap (\mathcal{I} \setminus \mathcal{U}(F))$. From R the algorithm eventually leads to a stationary configuration in $\mathcal{I} \setminus \mathcal{U}(F)$ belonging to T_4, T_6, T_7, T_8, or to a configuration in $\mathcal{I} \setminus \mathcal{U}(F)$ belonging to T_3.*

Proof. In this task $\mathtt{pre}_2 = \neg\mathtt{c}$ holds and, consequently, all preconditions concerning tasks T_3, T_4, …, T_{11} are false. We recall that task T_2 is responsible for the correct removal of the robots from Ann toward C^T in a configuration R. Hence $C(R)$ cannot change. Notice that in R, and during all the movements of all robots in Ann, variable $\mathtt{a} = \mathtt{false}$. Moreover, there might be a number of robots equal to $\rho(R)$ that can move concurrently according to m_2 (this may occur when the processed configuration is symmetric). In particular, all robots in Ann closest to $c(R)$ and of minimal view move according to the trajectory computed by Procedure $\mathtt{GoToC^T}$. Note that at beginning of task T_2 the configuration could be non-stationary if the previous performed task is T_3.

H_2: if the configuration R is stationary, by Lemma V.1 no configuration in $\mathcal{U}(F)$ is generated. If the configuration R is non-stationary then the transition that led to R was robust as generated from task T_3 by calling the same Procedure $\mathtt{GoToC^T}$. By similar arguments provided in the proof of Lemma V.1, it is possible to show that unsolvable configurations cannot be generated.

H_3: when all the moving robots reach their target, the configuration can be in T_2 again if there were more robots in Ann than the moved ones (e.g., when there are circles $C^i_\downarrow$ with different index i inside Ann). The configuration remains in T_2 as long as $Ann \neq \emptyset$. Once this occurs, all the robots from Ann have reached C^T, and the resulting configuration R' cannot be in T_1 as $\mathtt{c} = \mathtt{false}$, in T_9 as $\mathtt{p}$ remains false by the computed targets of $\mathtt{GoToC^T}$, in T_{11} as $\mathtt{w}$ remains false. In contrast, R' could be in any class T_3, T_4, T_6, T_7, T_8, depending on the status of the variables.

$H_{3'}$: the transition to T_3 might be unclassified if R was originally generated from T_3 itself by means of an unclassified transition. Otherwise, and for any other task different from T_2, the obtained configuration is stationary as variable $\mathtt{a}$ changes its value only when all the robots in Ann reach C^T. The self-loop is unclassified as the set of robots involved by m_2 might change as well as their trajectories. However, Lemma V.1 ensures to make Ann empty eventually.

$H_{3''}$: Lemma V.1 guarantees that any configuration obtained while performing task T_2 has no multiplicities. This implies that move m_2 is collision-free.

H_4: if a robot does not reach its target because of the adversary, then the configuration remains in T_2, since no variable changes its value and Ann is not empty ($\mathtt{a}$ remains false). However the moving robot decreases its distance to C^T, so task T_2 can be performed a finite number of times.

□

Lemma VI.11. *Let R be a stationary configuration in $T_1 \cap (\mathcal{I} \setminus \mathcal{U}(F))$. From R the algorithm eventually leads to a stationary configuration in $\mathcal{I} \setminus \mathcal{U}(F)$ belonging to T_2, T_3, T_4, T_5 or T_6.*

Proof. In this task $\mathtt{c} = \mathtt{true}$, which means there is exactly one robot r inside C^B that must be moved. Robot r is moved toward any point at distance $\delta(C^B)$ from $c(R)$. Hence $C(R)$ cannot change. If the robot does not reach its target, move m_1 is repeatedly applied to r until a point on C^B is reached by the robot. Then, if r does not occupy $c(r)$ its trajectory is radial.

- H_2: since $\mathtt{c} = \mathtt{true}$, a single robot is in $int(C^B)$ and then the configuration admits symmetricity equal to one along all the movement of r, that is no unsolvable configurations are generated.
- H_3: when r reaches its target (possibly after applying move m_1 many times) all the variables remain unchanged except $\mathtt{c}$ that becomes false. In particular, $\mathtt{w} = \mathtt{false}$ as the moving robot remains confined on C^B, that is it has not reached a possible target point of F, regardless the embedding; $\neg\mathtt{m} \wedge \mathtt{p}$ remains false as neither robots on $C(R)$ nor robots in Ann moved and r has not reached a possible target point of F; $\mathtt{a}$ remains unchanged as robots in Ann are not moved; $\mathtt{d}_1$, $\mathtt{d}_2$ and $\mathtt{u}$ remain unchanged as robots on $C(R)$ are not moved. We can then conclude that the final configuration can be only in T_2, T_3, T_4, T_5, or T_6.
- $H_{3'}$: the reached configuration is stationary as the only moving robot is r and no other robot moves as all the variables remain unchanged during the movement. The self-loop is instead almost-stationary as the moving robot will be moved along the same trajectory until reaching C^B.
- $H_{3''}$: collisions cannot occur being r the only robot inside C^B.
- H_4: the repeated application of m_1 eventually ends as the distance of r to its target reduces.

□

We are now ready to state the correctness of the algorithm.

Theorem VI.12 (Correctness)**.** *Let R be an initial configuration of* ASYNC *robots with chirality, and F be any pattern (possibly with multiplicities) with $|F| = |R|$. Then, there exists an algorithm able to solve the Pattern Formation problem if and only if $\rho(R)$ divides $\rho(F)$.*

Proof. ($\Longrightarrow$) This is the case in which $\rho(R)$ does not divide $\rho(F)$. By Theorem III.3, F is not formable from R.

($\Longleftarrow$) To this aim, it is sufficient to show that the provided algorithm fulfills all properties $H_1, \ldots, H_4$. Concerning property H_1, we have already pointed out at the beginning of this section that the tasks' predicates $P_1, P_2, \ldots, P_{11}$ used by the algorithm have been defined as suggested by Equation 1; then, according to Remark IV.1, H_1 holds. By Lemmas VI.1-VI.11 we have that both H_2 (i.e., no unsolvable configurations are created) and H_3 (i.e., the transition graph is exactly that represented in Figure 4) are true. In order to conclude the proof, we need to prove property H_4. By Lemmas VI.1-VI.11 we have that self-loops are executed a finite number of times. We can then focus on the simple cycles contained in the transition graph shown in Figure 4, that are: (T_2, T_3), (T_2, T_4), (T_2, T_6, T_3), (T_2, T_4, T_6, T_3).

Considering node T_2, which belongs to all such simple cycles, we now show it can be entered a limited number of times. In particular, concerning the nodes involved in the simple cycles, T_2 can be reached from T_3 and T_4 by means of moves m_3 and m_4, respectively. Actually, both moves decrease $\partial C(R)$ of at least one robot. Since none of the involved tasks in the cycles increases $\partial C(R)$, then any cycle involving T_2 can occur a finite number of times. □

VII. CONCLUSION

We considered one of the most fundamental problems in the context of distributed computing by mobile robots, that is Pattern Formation with chirality. We provided a full characterization, ultimately showing that asynchronous robots are as powerful as synchronous ones with respect to the feasibility of the studied problem. This answers to an old-standing question about the solvability of the studied problem by means of asynchronous robots. Moreover, our result highlights the higher difficulties in designing a resolution strategy in the context of ASYNC robots with respect to FSYNC (or SSYNC) ones. However, we have exploited a recent methodology in order to approach the problem so that the correctness proof results to be well-structured and less prone to faulty arguments as it happened in previous attempts.

As main open question, it is still not known whether PF without assuming chirality is solvable, even by FSYNC robots.

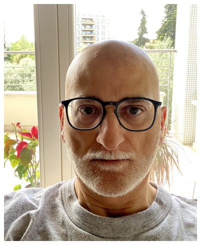
Another interesting direction of investigation concerns whether there exists a separation problem for robots moving in the Euclidean plane between SSYNC and ASYNC, that is a problem solvable by SSYNC robots but not by ASYNC ones, within the same setting considered here. Concerning the separation between FSYNC and SSYNC, it is sufficient to consider the *rendez-vous* problem, that is the gathering problem when only two robots are considered. For robots moving on graphs, by [16] it is known that such a separation exists but the considered problem does not preserve the same properties in the context of robots moving in the Euclidean plane.

REFERENCES

[1] Subhash Bhagat, Sruti Gan Chaudhuri, and Krishnendu Mukhopadhyaya. Formation of general position by asynchronous mobile robots under one-axis agreement. In Proc. 10th Int.'l WS on Algorithms and Computation (WALCOM), volume 9627 of LNCS, pages 80–91. Springer, 2016.

[2] Marjorie Bournat, Swan Dubois, and Franck Petit. Computability of perpetual exploration in highly dynamic rings. In Kisung Lee and Ling Liu,

editors, 37th IEEE International Conference on Distributed Computing Systems, ICDCS 2017, Atlanta, GA, USA, June 5-8, 2017, pages 794–804. IEEE Computer Society, 2017.
[3] Serafino Cicerone, Alessia Di Fonso, Gabriele Di Stefano, and Alfredo Navarra. Arbitrary pattern formation on infinite regular tessellation graphs. In Proc. 22nd Int.'l Conf. on Distributed Computing and Networking (ICDCN), page 56–65, New York, NY, USA, 2021. ACM.
[4] Serafino Cicerone, Gabriele Di Stefano, Leszek Gasieniec, Tomasz Jurdzinski, Alfredo Navarra, Tomasz Radzik, and Grzegorz Stachowiak. Fair hitting sequence problem: Scheduling activities with varied frequency requirements. In Algorithms and Complexity - 11th International Conference, CIAC, volume 11485 of LNCS, pages 174–186. Springer, 2019.
[5] Serafino Cicerone, Gabriele Di Stefano, and Alfredo Navarra. "Semi-Asynchronous": a new scheduler for robot based computing systems. In Proc. 38th IEEE Int.'l Conf. on Distributed Computing Systems, (ICDCS), pages 176–187. IEEE, 2018.
[6] Serafino Cicerone, Gabriele Di Stefano, and Alfredo Navarra. Asynchronous arbitrary pattern formation: the effects of a rigorous approach. Distributed Computing, 32(2):91–132, 2019.
[7] Serafino Cicerone, Gabriele Di Stefano, and Alfredo Navarra. Embedded pattern formation by asynchronous robots without chirality. Distributed Computing, 32(4):291–315, 2019.
[8] Serafino Cicerone, Gabriele Di Stefano, and Alfredo Navarra. A methodology to design distributed algorithms for mobile entities: the pattern formation problem as case study. CoRR, abs/2010.12463, 2020.
[9] Serafino Cicerone, Gabriele Di Stefano, and Alfredo Navarra. "Semi-Asynchronous": a new scheduler in distributed computing. IEEE Access, 9, 2021.
[10] Mark Cieliebak, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Distributed computing by mobile robots: Gathering. SIAM J. on Computing, 41(4):829–879, 2012.
[11] Mark Cieliebak and Giuseppe Prencipe. Gathering autonomous mobile robots. In Proc. of the 9th Int.'l Colloquium on Structural Information and Communication Complexity (SIROCCO), volume 13, pages 57–72. Carleton Scientific, 2002.
[12] Jurek Czyzowicz, Leszek Gasieniec, Adrian Kosowski, Evangelos Kranakis, Danny Krizanc, and Najmeh Taleb. When patrolmen become corrupted: Monitoring a graph using faulty mobile robots. Algorithmica, 79(3):925–940, 2017.
[13] Gianlorenzo D'Angelo, Mattia D'Emidio, Shantanu Das, Alfredo Navarra, and Giuseppe Prencipe. Asynchronous silent programmable matter achieves leader election and compaction. IEEE Access, 8:207619–207634, 2020.
[14] Shantanu Das, Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Masafumi Yamashita. Autonomous mobile robots with lights. Theor. Comput. Sci., 609:171–184, 2016.
[15] Shantanu Das, Paola Flocchini, Nicola Santoro, and Masafumi Yamashita. Forming sequences of geometric patterns with oblivious mobile robots. Distributed Computing, 28(2):131–145, 2015.
[16] Mattia D'Emidio, Gabriele Di Stefano, Daniele Frigioni, and Alfredo Navarra. Characterizing the computational power of mobile robots on graphs and implications for the euclidean plane. Inf. Comput., 263:57–74, 2018.
[17] Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Self-deployment of mobile sensors on a ring. Theor. Comput. Sci., 402(1):67–80, 2008.
[18] Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Peter Widmayer. Arbitrary pattern formation by asynchronous, anonymous, oblivious robots. Theor. Comput. Sci., 407(1-3):412–447, 2008.
[19] Nao Fujinaga, Yukiko Yamauchi, Hirotaka Ono, Shuji Kijima, and Masafumi Yamashita. Pattern formation by oblivious asynchronous mobile robots. SIAM J. Computing, 44(3):740–785, 2015.
[20] Nao Fujinaga, Yukiko Yamauchi, Hirotaka Ono, Shuji Kijima, and Masafumi Yamashita. Erratum: Pattern formation by oblivious asynchronous mobile robots. http://tcs.inf.kyushu-u.ac.jp/~yamauchi/manuscripts/E-FYOKY15.pdf, 2017.
[21] Leszek Gasieniec, Ralf Klasing, Russell A. Martin, Alfredo Navarra, and Xiaohui Zhang. Fast periodic graph exploration with constant memory. J. Comput. Syst. Sci., 74(5):808–822, 2008.
[22] Swapnil Ghike and Krishnendu Mukhopadhyaya. A distributed algorithm for pattern formation by autonomous robots, with no agreement on coordinate compass. In Proc. 6th Int.'l Conf. on Distributed Computing and Internet Technology, (ICDCIT), volume 5966 of LNCS, pages 157–169. Springer, 2010.
[23] Akitoshi Kawamura and Yusuke Kobayashi. Fence patrolling by mobile agents with distinct speeds. Distributed Computing, 28(2):147–154, 2015.
[24] Nimrod Megiddo. Linear-time algorithms for linear programming in R^3 and related problems. SIAM J. Comput., 12(4):759–776, 1983.
[25] Ichiro Suzuki and Masafumi Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. SIAM J. Comput., 28(4):1347–1363, 1999.
[26] Emo Welzl. Smallest enclosing disks (balls and ellipsoids). In Results and New Trends in Computer Science, pages 359–370. Springer-Verlag, 1991.
[27] Masafumi Yamashita and Ichiro Suzuki. Characterizing geometric patterns formable by oblivious anonymous mobile robots. Theor. Comput. Sci., 411(26-28):2433–2453, 2010.
[28] Yukiko Yamauchi, Taichi Uehara, Shuji Kijima, and Masafumi Yamashita. Plane formation by synchronous mobile robots in the three-dimensional euclidean space. J. ACM, 64(3):16:1–16:43, 2017.



SERAFINO CICERONE is an Associate Professor at the Department of Information Engineering, Computer Science and Mathematics of the University of L'Aquila. He graduated in Computer Science from the University of L'Aquila in 1993 and the PhD from the University of Rome "La Sapienza" in 1998. In general, his research interests revolve around the specification, design, verification and implementation of efficient algorithms. Specific areas of interest include distributed algorithms, combinatorial optimization, algorithm engineering, algorithmic graph theory, spatial and geometric data.

GABRIELE DI STEFANO is a Full Professor at the Department of Information Engineering, Computer Science and Mathematics of the University of L'Aquila.
He received his Ph.D. at University "La Sapienza" of Rome in 1992. His current research interests include network algorithms, combinatorial optimization, algorithmic graph theory, distributed computing. He is (co-)author of more than 120 publications in journals and international conferences. He had key-participations in several EU funded projects. Among them: MILORD (AIM 2024), COLUMBUS (IST 2001-38314), AMORE (HPRN-CT-1999-00104), ARRIVAL (IST FP6-021235-2), and GEOSAFE (H2020-691161).

ALFREDO NAVARRA is Associate Professor since 2015 at the Mathematics and Computer Science Dept, University of Perugia, Italy. He received the PhD in Computer Science in 2004 from "Sapienza" University of Rome, Italy. Before joining the University of Perugia, he has been with various international research institutes like the INRIA of Sophia Antipolis, France; the Dept of Computer Science atthe Univ. of L'Aquila, Italy; the LaBRI, Univ. of Bordeaux, France. His research interests include algorithms, computational complexity, distributed computing and networking.