

Uniform Circle Formation for Fully, Semi-, and Asynchronous Opaque Robots with Lights^{*}

Caterina Feletti, Carlo Mereghetti[†], Beatrice Palano and Priscilla Raucci

Università degli Studi di Milano, Italy

Abstract

We study the `Uniform Circle Formation` (UCF) problem, which asks a swarm of mobile agents, arbitrarily positioned onto the plane, to arrange on the vertices of a regular polygon. Each agent, customarily called robot, runs a distributed algorithm by executing a sequence of look-compute-move cycles. The robot swarm may adhere to three synchronization modes: fully synchronous, semi-synchronous, and asynchronous. Our robots are assumed to be punctiform, anonymous, and indistinguishable by their appearance; they do not store past actions or system snapshots, and they have neither a coordinate system nor chirality in common. Moreover, we consider *opaque* robots, i.e. they may have obstructed visibility due to collinearities. To cope with these strong limitations, we consider *luminous* robots, that is, they are equipped with a persistent light assuming different colors. This latter peculiarity represents the only way robots have to communicate. For all three synchronization modes, we solve the UCF problem with a constant number of colors. Concerning the running time, our solutions use a constant number of cycles (epochs) for fully synchronous (semi-synchronous) robots, and linearly many epochs in the worst-case for asynchronous robots.

Keywords

Autonomous mobile robots, Opaque robots, Luminous robots, Pattern formation

1. Introduction

A well consolidated trend in the literature on distributed computing studies models and algorithms for agent-based computing systems, having great relevance in several real-world applications. In these systems, a swarm of mobile computing entities, called *robots*, has to cooperate to solve a given problem. Robots act under several assumptions on their capabilities and on a particular scenario. Of great importance are models where robots are autonomous, i.e. they work without a central control, and operate through a sequence of *look-compute-move* cycles in which each robot: (i) takes the snapshot of the system (*look*), (ii) executes a deterministic algorithm (*compute*), and (iii) travels to the computed destination, if any (*move*) [1, 2, 3, 4].

Different modeling assumptions are considered, affecting the computational power of robots. For instance, robots may have distinct identifiers (yielding the ability of distinguishing one

Proceedings of the 23rd Italian Conference on Theoretical Computer Science, Rome, Italy, September 7-9, 2022

^{*} Partially supported by UniMI through the *Seal of Excellence (SoE) SEED 2020*, under the project *Self-Organizing Photonic Quantum Links (S-O PhoQuLis)*.

[†] Corresponding author.

✉ carlo.mereghetti@unimi.it (C. Mereghetti); beatrice.palano@unimi.it (B. Palano); priscilla.raucci@unimi.it (P. Raucci)

ORCID 0000-0002-7778-7257 (C. Mereghetti); 0000-0003-3948-4658 (B. Palano)

 © 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

robot from another) or, on the contrary, they may be anonymous. They may have a finite but persistent memory, which is preserved from one look-compute-move cycle to the next one. If no such memory exists, robots are said to be oblivious. A “compromise” between memory and obliviousness is given by luminous robots, featuring a persistent light assuming different colors as a means of communication as well. Another step towards realistic models is to work with no point-like (punctiform model) but fat robots, where all robots are supposed to be solid discs with a certain radius. Moreover, robots can be transparent, enabling a complete visibility of the system, or opaque. Depending on the nature of the problem, robots can move either on the Euclidean plane, or on a graph which can either be known in advance or not. For robot activation policy, three models are proposed in the literature: *fully synchronous*, where all robots execute their cycle synchronously, *semi-synchronous*, where a subset of robots executes its cycle synchronously whereas the others remain idle, and *asynchronous*, where each robot acts asynchronously.

Several research efforts focus on very basic classes of geometric pattern formation problems to be solved within such distributed environments [5, 6, 7, 8, 9, 10, 11, 12, 13, 14]. In the `Uniform Circle Formation` problem, we ask robots to move to vertices of a common regular polygon whose number of vertices — known or not in advance — is exactly the number of robots in the system. This problem received a lot of attention from the literature (see, e.g., [15] for a survey) for both theoretical and practical reasons. From a practical point of view, a regular layout may present several advantages for a distributed system. E.g., for a network of mobile agents, it may be convenient to regularly displace them to facilitate communications, visibility and computations. Every agent is equidistant from its neighbors and has the same view of the system: this guarantees a fair communication, where there are no evident differences in the energy spent in sending messages. Moreover, this uniform pattern allows to implement distributed algorithms which guarantee a fair load balancing among the agents.

In [16], an algorithm for `Uniform Circle Formation` is designed, which asymptotically converges to a regular polygon. In [17], a semi-synchronous solution is given, starting from particular robot configurations. For asynchronous systems, the problem is solved in [18] for robots which are punctiform and transparent, and in [19] for fat robots with limited visibility and agreeing on common origin and axes.

So far, swarms of *transparent* robots have been mainly considered. In more realistic models, robots are assumed to be *opaque* so that they can obstruct visibility in case of collinearity. In this realm, the first natural problem to be tackled is `Complete Visibility`, where robots are required to displace on the plane so that each robot is visible to all others. For the `Complete Visibility` problem, opaqueness is a serious problem. Thus, additional robot capabilities might be considered. Solutions in the literature are proposed for *luminous robots* (see, e.g., [20, 21, 22, 23, 1, 2]), i.e., robots with persistent lights assuming different colors. In [24], a $\mathcal{O}(\log N)$ time and $\mathcal{O}(1)$ colors asynchronous algorithm is designed for `Complete Visibility`. In [25] ([26]), the problem is solved by a $\mathcal{O}(1)$ time and colors semi-synchronous (asynchronous) algorithm. In [27], a fault-tolerant algorithm for `Complete Visibility` is exhibited.

In [28], we focused on the `Uniform Circle Formation` (UCF) problem which, as recalled above, consists of displacing robots on the vertices of a common regular polygon. In particular, we exhibited an algorithm solving UCF for a swarm of opaque robots with lights *only in the fully synchronous* model, featuring six cycles and a constant number of colors. In the present paper,

instead, we propose algorithms to solve UCF by opaque luminous robots in the *fully*, *semi*-, and *asynchronous* settings. In particular, the algorithm for the fully synchronous case improves the one in [28] in that it uses only three instead of six cycles and a constant number of colors. The semi-synchronous case is solved within constant time (epochs) and a constant number of colors. Finally, we adapt our algorithmic designs in order to obtain a solution for the asynchronous model, working in worst-case linear time (epochs) and with a constant number of colors.

Due to page limit, some material and proofs have been omitted.

2. Preliminaries

We overview the distributed system we shall be dealing with (see, e.g., [3, 4, 23] for details), and formally state the Uniform Circle Formation (UCF) problem.

Robot swarms. Consider a finite set (swarm) of punctiform computational agents, called robots, which forms a distributed system located in the plane $\mathbb{R}^2$. These robots are: (i) *anonymous and indistinguishable*: they do not share any own identifier, (ii) *autonomous*: there is no central coordinator, (iii) *homogeneous*: they execute the same deterministic algorithm, (iv) *oblivious*: they do not remember any data about previous actions, (v) *mobile*: they can freely move on the plane, provided they never collide, (vi) *rigid*: they cannot be stopped before reaching the computed destination (i.e., no adversary can stop robot movement).

The robots are equipped with sensory capabilities to spot the positions of other robots. Moreover, they are able to compute in finite time and infinite precision any algebraic function of points in the plane. Also, we assume the following limitations on robots: (i) they do not know how many they are, (ii) they are disoriented: no agreement among the individual coordinate systems, nor on unit distance and chirality (roughly speaking, agreement on clockwise direction), (iii) *they are opaque (not transparent)*: collinearity causes obstructed visibility. Indeed, these latter three inabilities introduce complications in algorithm design. E.g., due to opacity (iii) and lack of knowledge of the number of robots (i), each robot may not be able to know whether or not some robots are hidden at any given time. Clearly, this lack of the knowledge of the number of robots makes the system easily scalable. Moreover, the disorientation (ii) might cause robot collisions which may compromise system integrity. To deal with these adversities, *we equip robots with a light displaying a certain number of different colors they can communicate through* (see, e.g., [1, 2, 20, 21, 22, 23]). We remark that such a light is the only means robots have to exchange information.

When taking the system *snapshot* at any given time, a robot r basically collects the coordinates —*according to its own coordinate system*— and the light color for any visible robot (itself included). So, r operates in *look-compute-move* cycles, each executed in a single and atomic instant of time and consisting of the three phases:

- **Look:** r takes the instantaneous snapshot of the system.
- **Compute:** r runs a deterministic algorithm which, by having the snapshot as sole input, computes the destination point of r and the (possibly) new color for the light of r .
- **Move:** r sets its new color and moves straight towards the destination point computed above, without being stopped (rigidity assumption).

Different models of robot activation and synchronization are studied. In the *fully synchronous* model, time is logically subdivided into global *rounds*. All robots are activated at every round occurring at each time t . Since *look-compute-move* cycles are executed atomically, all robots terminate their cycle by the next round. The *semi-synchronous* model coincides with the fully synchronous model, except that not all robots are necessarily activated at every round. However, every robot is activated infinitely often (*fair condition*). In the *asynchronous* model, the robots are activated independently and each robot executes its *look-compute-move* cycle within an unpredictable but finite amount of time. There does not exist a common notion of time. Also, notice that the configuration perceived by a robot during its look phase — by which the robot computes its final destination — may significantly change before the robot starts moving.

The UCF problem. We are now ready to present the UCF problem to be solved on fully, semi-, and asynchronous swarms of opaque robots with lights. Let a fully, semi-, or asynchronous swarm of n robots be in any given initial valid configuration C_0 (i.e. where all the robots occupy a distinct point of the plane). The UCF problem asks the swarm to move from C_0 to a valid terminal configuration in which robots form a regular n -gon.

3. Some Notions and Results

Throughout the paper, we will be working with swarms of at least 5 robots. This is due to some technicalities that can be dealt with, but that will be omitted here.

Let a set of n robots sit on their *smallest enclosing circle* (SEC). Two distinct robots p, q on the SEC delimit two arcs $\widehat{pq}$ (clockwise and counterclockwise). We say that p and q are *adjacent* whenever there is at least one of the two arcs $\widehat{pq}$ upon which no other robot sits. Clearly, in a robot swarm forming a regular n -gon, each pair p, q of adjacent robots forms an angle of $2\pi/n$ with the center O of the SEC, formally $\widehat{pOq} = 2\pi/n$.

Definition 1. For any given $2 \leq k \leq n$, let $(r_1, r_2, \dots, r_k)$ be a k -tuple of distinct robots satisfying $\widehat{r_i O r_{i+1}} = 2\pi/n$ for every $1 \leq i < k$. Then, $(r_1, r_2, \dots, r_k)$ is said to be a *regular k -tuple*. For odd k (even $k > 2$), the central robot $r_{\lceil \frac{k}{2} \rceil}$ (the central robots $r_{\frac{k}{2}}, r_{\frac{k}{2}+1}$) is the *pivot* (are the pivots) of the regular k -tuple.

Our strategies to move the robots on the vertices of a regular n -gon (inscribed in the SEC) start by setting particular regular $\{3, 4, 5\}$ -tuples. These tuples will not move for the whole computation, while the other robots will move to form the regular n -gon. In particular, the pivots in such regular tuples will be crucial to set the *main diameter* of the SEC (see Definition 2). A key result for our algorithms is contained in the following

Theorem 1. Let n be the number of robots lying on the SEC.

- For odd n , there is a diameter passing through a robot and dividing the SEC into two halves, each having $(n-1)/2$ robots.
- For even n , there is a diameter dividing the SEC into two halves such that: (i) either the diameter passes through a robot and the half-SECs have $n/2$ and $(n/2)-1$ robots, or (ii) the diameter passes through two (opposite) robots and the half-SECs have $(n/2)-1$ robots each.

Another central role in our algorithms will be played by the notion of a main diameter. We distinguish between three cases, depending on the type of symmetry possessed by the robot swarm configuration:

Definition 2.

- **Asymmetry.** A diameter settled according to Theorem 1 will be called the main diameter. In this case, the main diameter has at least one robot at one endpoint, whose light will assume color pivot.
- **Symmetry with exactly one axis.** We call main diameter the diameter laying on the symmetry axis. In this case, the main diameter does not necessarily pass through a robot with light of color pivot.
- **Rotational symmetry with two sectors.** We call main diameter the diameter joining two opposite robots whose lights have color pivot.

Let d be a main diameter by Definition 2. Let b be one of the nearest robots to d but not laying on d , and let $\ell(b, d)$ be the distance from b to d . Moreover, let ν be one of the closest but not belonging to d vertices of the regular polygon that has to be formed by our robot swarm.

Definition 3. Let d' and d'' be the two opposite chords parallel to a main diameter d , at distance $\frac{\min\{\ell(b, d), \ell(\nu, d)\}}{2}$ from d . We call d' and d'' the safe diameters of d .

For rotational symmetries with more than two sectors, instead of safe diameters we will be considering safe chords:

Definition 4. Let (a_1, p_1, a_2) and (a_3, p_2, a_4) be two adjacent regular 3-tuples (i.e., without any other 3-tuple on the arcs $\widehat{a_1 a_3}$ or $\widehat{a_2 a_4}$), and suppose that p_1 and p_2 are not the two endpoints of a same diameter. The chord joining p_1 and p_2 is called a safe chord.

As we will see, a key operation of our algorithms is to set at most three robots on three points per each half-SEC, such points being actually three vertices of the regular polygon. The robots reaching such vertices will assume color *angle* or *pivot*. In the following theorem, we show that this can be done in a single cycle, provided robots share a common clockwise direction in the half-SEC they sit on:

Definition 5. Two or more robots are said to be oriented whenever they agree on a common clockwise direction.

For a group of oriented robots, it is always possible to unambiguously spot robots to move within a cycle (e.g., the last ones according to the common orientation). This is crucial for the following

Theorem 2. Given three oriented robots and three points on the same half-SEC, it is always possible for the robots to reach these points in a single cycle and avoiding collisions.

4. The Algorithm for the Fully Synchronous Case

In a first phase, all robots gain *complete visibility* of the swarm [25, 26], and then move onto their SEC maintaining the knowledge of the exact number n of robots in the system. The robots work out this phase by using a set of colors different from the colors used later on. This ensures robots are always aware of the ongoing phase.

Let C_H be the swarm configuration at this point; without loss of generality, we assume robots lights having the same color. Clearly, all robots are again vertices of a convex hull. The resulting dynamic of the algorithm depends on the type of this convex hull, in particular on its degree of symmetry. First of all, a special case occurs whenever the convex hull is *perfect* [29]. In this case, all robots lie onto the edges of the associated regular n -gon (called *supporting polygon*, SP), two robots per alternate edges. The goal of our algorithm for this particular configuration is to slide robots along the edges of the SP, until they reach the vertices of the SP. Given a perfect convex hull, the SP is unique and computable in a single round: each robot takes its snapshot (*look*), checks whether the system configuration forms a perfect convex hull and computes the SP (*compute*), eventually slides along the edge until it reaches the correct vertex (*move*). Notice that two robots on the same edge head in opposite directions, and therefore no collision occurs. We remark that a *biangular configuration*¹ is a special case of perfect convex hull. Also in [18], the perfect convex hull and biangular configuration are dealt with as special cases at the beginning of their algorithm.

Let us show how our algorithm works.

4.1. Cycle 1: pivots selection and angle setting

From C_H , we start setting the regular tuples which will be the reference points for the movements of the other robots. Once settled, a regular tuple will not move anymore, thus fixing once and for all for the whole robot swarm the angle $2\pi/n$ of the regular n -gon to be formed. As observed above, the dynamic of the algorithm depends on the degree of symmetry of C_H . We distinguish between three cases: *asymmetry*, *symmetry with exactly one axis*, and *rotational symmetry*.

Asymmetry. Consider the general case of asymmetry, i.e., no symmetry axis exists in C_H . However, there exists at least one diameter passing through a robot and dividing the SEC into two halves, upon which robots distribute according to Theorem 1. The robot swarm must agree on one of such diameters, say D , which by Definition 2 will be the *main diameter*. To find D , robots agree on starting from a commonly designated robot x and following a common orientation on the SEC. To perform this task, we need

Lemma 1. *Starting from the configuration C_H and in case of asymmetry, all robots on the SEC are able to unambiguously agree on a robot x .*

So, let x be the robot chosen by the swarm according to Lemma 1 (i.e., x has the lexicograph-

¹A set of $n \geq 2$ robots forms a biangular configuration if robots lie on a circle C centered in O , and two non zero angles α, β exist such that for every pair r and p of robots consecutive on C , we have $\widehat{rOp} \in \{\alpha, \beta\}$ and α and β alternate clockwise [29].

ically smallest angle-string²). By the same reasoning, the swarm can also agree on a second robot y with the lexicographically second-smallest angle-string and such that x and y are not the endpoints of the same diameter. (Note that such a y can always be found since we are in an asymmetric configuration.) This allows us to have a *starting* robot (i.e., x) and an *orientation* on the SEC (i.e., the direction from x to y), by which the main diameter D search can be carried on: each robot starts checking whether the diameter through x satisfies Theorem 1; if not, it tries with the next robot along the orientation settled by y . Clearly, this process makes the whole swarm to converge on the claimed main diameter D . Let p be the position of the robot that determines D . Such a position represents a vertex of the regular polygon to be formed, so the light of the robot on position p assumes color *pivot*. Let us now distinguish between having an odd or even number n of robots in the swarm. Here, we discuss the latter case.

EVEN n . By Theorem 1, one or two robots may sit at the endpoints of D . In both cases, the robot at position p has color *pivot*. Moreover, in the second case, the opposite robot on D assumes color *angle*. Let us now continue discussing the first situation, since the second may be easily derived. By Theorem 1, one of the half-SEC, say H^+ , has one more robot than the other, say H^- . So, a robot in H^+ needs to move to the empty endpoint a_3 (opposite to p) of D and assume color *angle*. At the same time, two other positions a_1 and a_2 (regular polygon vertices) must be reached by two robots. These two positions correspond to the polygon vertices at the immediate left and right of p . The three robots sat at positions a_1, p, a_2 form the *regular 3-tuple*.

So, globally, three robots must be moved: two from H^+ and one from H^- , so that both the half-SECs contain the same number of robots at the end of this cycle. To determine which robots in H^+ and H^- must move at the vertices of the polygon, we use the orientation provided by x and y : for instance, we choose the last robots by such an orientation. Then, Theorem 2 ensures that the three vertices will be safely reached by the robots. Summing up, each robot r performs:

- **Look-Compute:** r unambiguously spots the pivot position p on the main diameter. Furthermore, it computes the positions a_1, a_2, a_3 and the robots heading to these positions.
- **Move:** If r is in the pivot position p , then it does not move and sets its color as *pivot*. If r is one of the robots heading to positions a_1, a_2, a_3 , it moves there and sets its color as *angle*. Otherwise, r does nothing.

Symmetry with exactly one axis³. Let l be the axis of symmetry in C_H . By Definition 2, the diameter of the SEC lying on l will be the *main diameter*. We consider three cases: odd n , even n with two axis pivots, and even n with no axis pivot. Here, we discuss the odd case.

ODD n . Suppose that l passes through a robot, which will be the pivot p , and splits the opposite edge of the polygon. So, l divides the SEC into two symmetric halves, each with $(n-1)/2$ robots (p excluded). Now, we aim to create the regular 3-tuple around p , as well as setting two robots at the endpoints of the polygon edge opposite to p . Therefore, two robots in each half-SEC must be moved. Even in this case, Theorem 2 ensures no crossing trajectories. However, notice that to apply Theorem 2 we need an orientation which in this case can be trivially settled in each half-SEC as being the direction from the pivot to the other endpoint of

²Let $r_0, \dots, r_{n-1}$ be a listing of the n robots on the SEC so that they are consecutive, i.e., r_i and r_{i+1} are adjacent for every $0 \leq i < n-1$. Let $\alpha_i = \widehat{r_i O r_{(i+1) \bmod n}}$. The corresponding angle-string is $\alpha_0 \cdots \alpha_{n-1}$.

³Two or more symmetry axes yield a rotational symmetry, considered in the next case.

the diameter (namely, the main diameter) laying on the axis l . To determine which robots in each half-SEC must move at the vertices of the polygon, we can choose, e.g., the last robots according to such an orientation. As in the asymmetric case discussed above, we call a_1 and a_2 the positions of the two polygon vertices around p , while a_3 and a_4 are the positions of the two vertices of the polygon edge opposite to p . Summing up, each robot r performs:

- **Look-Compute:** r computes the axis of symmetry l (and hence the main diameter) and the positions a_1, a_2, a_3, a_4 of the polygon vertices.
- **Move:** If r is on l , it sets its color as *pivot* and stays put. If r is one of the robots heading to positions a_1, a_2, a_3, a_4 , it gets there and sets its color as *angle*. Otherwise, r does nothing.

Rotational symmetry. Let $r_0, \dots, r_{n-1}$ be the sequence of robots consecutive on the SEC, starting from an arbitrary robot. If the related angle-string $\alpha_0 \cdots \alpha_{n-1}$ can be factored into k identical substrings up to rotation, then the convex hull on the SEC can be divided into k identical sectors, each being the $\frac{2\pi}{k}$ -rotation of the previous one. Let $P_i = \{r_j \mid j \equiv i \pmod{n/k}\}$ be the class of symmetry which contains n/k robots sharing the same position in the k different sectors. By using angle-strings, we can unambiguously choose one or two classes of symmetry. For the sake of simplicity, in what follows we assume a single⁴ *main class of symmetry* P . The robots in P will be the *pivots*. Note that, in case of rotational symmetry with two sectors, the diameter joining the two pivots will be the *main diameter*, by Definition 2. For more than two sectors, the chords joining pivots in two consecutive sectors will be the *safe chords*, by Definition 4. We now show how to set k *regular 3-tuples* in this k -angular configuration. Each robot r performs:

- **Look-Compute:** r computes the main class of symmetry $P = \{p_1, \dots, p_k\}$ and the positions $\{a_{j_1}, a_{j_2} \mid j \in \{1, \dots, k\}\}$ of the vertices which will be the nearest to elements in P in the regular n -gon. For a rotational symmetry with two (more than two) sectors, determining P amounts to settle the main diameter (the safe chords) as well.
- **Move:** If r belongs to P , it sets its color as *pivot*. If r is the nearest robot to some a_{i_1} or a_{i_2} , it sets its color as *angle* and moves to its nearest destination point a_{i_1} or a_{i_2} . If two robots share the same distance from their destination, we can unambiguously choose one robot (as before) by considering the distance from the pivot. Otherwise, r does nothing.

4.2. Cycle 2: rappelling down on the safe diameters or safe chords

Let us distinguish among the following cases:

One or two regular 3-tuples, or 4-tuples. Let us consider the cases of asymmetry, symmetry with one axis, and rotational symmetry with two sectors. Let d be their main diameter, which is uniquely determined as observed above. Let d' and d'' be their safe diameters. In this cycle, each robot r performs:

- **Look-Compute:** If r is *pivot* or *angle*, it does nothing. Otherwise, r computes: the safe diameters d' and d'' , its destination point t on the SEC, the point $t_\perp$, which is the projection of t on the safe diameter nearest to r .

⁴The case of two classes of symmetry follows by generalization.

- **Move:** r sets its color as *internal* and travels to $t_{\perp}$.

Three or more regular 3-tuples. Let us consider the case of $k \geq 3$ regular 3-tuples. It is easy to see that this kind of configuration is a rotational symmetry. In this cycle, each robot r performs:

- **Look-Compute:** If r is *pivot* or *angle*, it does nothing. Otherwise, r computes: the nearest safe chord c , its destination point t on the SEC, the point $t_{\perp}$, which is the projection of t on c .
- **Move:** r sets its color as *internal* and travels to $t_{\perp}$.

Our algorithm yields safe robot movements, as stated by the following

Lemma 2. *The rappelling in Cycle 2 of robots on safe diameters and chords yields collision-free trajectories.*

4.3. Cycle 3: reaching the SEC

At the end of the previous cycle, each robot which is not *pivot* or *angle* colored lies on a safe diameter or on a safe chord. Our strategy was to move the robot to a position where enough information from the system configuration is available to compute its final destination on the SEC. So, in this cycle, each robot r performs:

- **Look-Compute:** If r is *pivot* or *angle* colored, it does nothing. Otherwise, r is within the SEC on a safe chord or a safe diameter, and sees at least three robots on the SEC which are *pivot* or *angle* colored. Thus, r can reconstruct the SEC. So, r computes: the original SEC upon which it has to travel, the safe chord or safe diameter it currently lies on, its destination arc H , and the projection point t of r on H .
- **Move:** r sets its color as *sec* and travels to t .

5. The Algorithm for the Semi-Synchronous Case

Let us outline our algorithm to solve UCF on the semi-synchronous model, pointing out main differences with the fully synchronous algorithm presented in the previous section. Again, we let C_H be the swarm configuration where all robots sit onto their SEC, thus knowing the exact number n of robots in the system. We assume robots lights having the same color. The dynamic of the algorithm depends on the degree of symmetry of C_H .

It is worth remarking that, for the semi-synchronous model, an appropriate way to measure time is addressed by the notion of an *epoch*. By epoch, we mean an amount of time within which each robot will be activated at least once. This must occur by the fairness condition on the system, pointed out in Section 2.

Asymmetry. In the **first epoch**, with the same dynamic as in the fully synchronous case, we set the color *pivot* to the robot through which the main diameter passes, dividing the swarm into two subs-swarms of the same cardinality.

In the **second epoch**, we set the color *angle* to robots, but now we need two different colors *angle* and *angle_o* to fix a common orientation. For odd (even) n , we set one (two) regular 3-tuple(s) of type $(angle, pivot, angle_o)$, plus two robots of colors *angle_o* and *angle* for odd n . As in the fully synchronous setting, asymmetry always enables to provide a common orientation for the whole SEC.

In the **third epoch**, robots move to the main diameter. By the common orientation, the robots on a half-SEC can choose one of the two radiuses giving the main diameter. Then, they move onto such a radius as follows. An activated robot r first checks whether it sits on an n -gon vertex. If so and it is not already colored as *pivot*, *angle*, *angle_o*, then r assumes color *sec* and ends its task. Otherwise, r moves onto the radius in a position which univocally determines the polygon vertex on the SEC to be reached in the next epoch without collisions. At the beginning of the **fourth epoch**, all robots are on their own radius, except those on the SEC with light color *pivot*, *angle*, *angle_o*, *sec*. Each robot on the radiuses sees at least three lighted robots on the SEC. So, it computes its target vertex on the SEC by its current position.

Symmetry with exactly one axis. While in the fully synchronous case, *pivot* and *angle* colors are both turned on in the first cycle, here we spend the **first epoch** to determine only those robots acting as pivot, and light them with new colors *pivot_s1*, *pivot_s2*, *pivot_s3*. Determining these robots takes place as in the fully synchronous case, while the new colors must be used to signal along the whole evolution the fact that the system starts in a configuration with a single symmetry axis. This latter fact turns out to be crucial since, due to semi-synchronous dynamics, at a certain instant it may be the case that the initial symmetry is lost. We set *pivot* lights *pivot_s1*, *pivot_s2*, *pivot_s3* depending on the odd or even number of robots and on the number of robots (zero, one or two) laying on the symmetry axis at the beginning of this epoch. Precisely: **case a** – *pivot_s1* is used for an odd number of robots and one robot on the symmetry axis, **case b** – *pivot_s2* is used for an even number of robots and no pivot on the symmetry axis, **case c** – *pivot_s3* is used for an even number of robots and two pivots on the symmetry axis. So, in this **first epoch**, for **cases a** and **c**, the robots on the symmetry axis immediately assume color *pivot_s1* and *pivot_s3*, respectively, and stay put. For **case b**, the robots nearest to the polygon vertices around the symmetry axis move to such positions and assume color *pivot_s2* (the case of robots equally distant from these vertices can be dealt with, e.g., by considering their distance from the symmetry axis).

Now, the **second epoch** comes: we move some robots on polygon vertices, and light them up with *angle* color. As in the fully synchronous case, robots can agree on an *upper* and a *lower* direction. So, in **case a**, a *regular 3-tuple* $(angle, pivot_s1, angle)$ is formed in the upper part of the SEC, and a *regular 4-tuple* $(angle, angle, angle, angle)$ in the lower part. For **case b**, a *regular 4-tuple* $(angle, pivot_s2, pivot_s2, angle)$ is formed in the upper part of the SEC, and a *regular 2-tuple* $(angle, angle)$ in the lower part. For **case c**, a *regular 3-tuple* $(angle, pivot_s3, angle)$ is formed in the upper part of the SEC, and a *regular 5-tuple* $(angle, angle, pivot_s3, angle, angle)$ in the lower part. We stress that in all these cases, no more than three robots per each half-SEC move to reach their final destinations. Hence, by Theorem 2, no crossing trajectory exists.

For the following epochs, we note that, given the type of symmetry we are considering, the whole swarm cannot share a common sense of orientation. So, in the **third epoch**, along which robots move into the SEC, they cannot choose a radius to drop on, as in the asymmetric case. Instead, as in the fully synchronous case, they can set safe diameters and move onto them on

the orthogonal projections of polygon vertices. At this point, in the **fourth epoch**, robots get back to the SEC by leaving perpendicularly from safe diameters. It is worth mentioning that during this latter epoch, in all three **cases a, b, and c** at least three robots with lights on are always visible, so that the SEC can be safely reconstructed.

Rotational symmetry. In the **first epoch**, we determine only those robots acting as pivot and light them with the new color *pivot_r*. This new color is used to signal along the whole evolution the fact that the system starts in a rotational symmetry configuration. Let $h > 2$ be the number of rotational symmetry sectors (the case $h = 2$ can be managed by adapting the asymmetric case technique). The following epochs basically take place as in the fully synchronous case. During the **second epoch**, we set h regular 3-tuples (*angle*, *pivot_r*, *angle*); in the **third epoch**, robots drop down to the safe chords; in the **fourth epoch**, all robots get back to the SEC at polygon vertex positions.

6. The Algorithm for the Asynchronous Case

In an asynchronous setting, severe problems may arise whenever robots are activated while other robots are moving to their destinations. In fact, the awoken robots would take snapshots which most likely will not be useful to exactly reconstruct some fundamental aspects of the system (e.g., SEC, main diameters, safe diameters and chords), crucial for their correct motion. This is basically due to the fact that moving robots can obscure other robots. Moreover, even if this is not the case, we cannot generally compute the final destination of a moving robot. Clearly, this could prevent an activated robot from targeting its correct final destination, e.g., when reaching angle and pivot positions, or right positions on main diameter, safe diameters and chords. To overcome the problem of moving robots, new light colors of type *moving* are lighted up as soon as a robot starts moving. Thus, we can use our semi-synchronous algorithm plus the following features, depending on the phases of the algorithm:

- In the first phase, pivots and then angles must be set, according to the same precedence logic as in the semi-synchronous mode. More precisely, if an activated robot does not see such points and does not see moving robots (i.e., robots with some *moving* light colors), it establishes whether or not it has to move towards a pivot or angle position. In the affirmative, its light assumes color *moving_pivot_s2*, or *moving_angle*, or *moving_angle_o* depending on its final destination, and the robot moves there. Once re-activated, a robot with color *moving_x* simply changes its color into $x \in \{pivot_s2, angle, angle_o\}$, and stands still. Otherwise, if the robot does not have to reach a pivot or angle position, or it sees some robots with color light *moving*, then it stays put.
- The second phase starts after setting angles and pivots. Robots enter the SEC on diameters and chords only if they see no other robot with color *moving_internal*. Activated robots that sit on n -gon vertices and are not colored by *pivot*, *angle*, *angle_o*, assume color *sec* and stay put. When entering the SEC, a robot assumes color *moving_internal*. Once re-activated, a robot colored *moving_internal* simply switches its color into *internal* and stays put.
- The third phase starts whenever on the SEC there are only robots with lights on (with colors *pivot*, *pivot_r*, *pivot_s1*, *pivot_s2*, *pivot_s3*, *angle*, *angle_o*, *sec*), and no robot is

moving (i.e., has a light color of type *moving*). In this phase, robots get back to SEC from their positions on chords and diameters. A robot travels back only when it sees three light colors among *pivot*, *angle*, *angle_o*, so that it can reconstruct the SEC. If this is the case, it determines its n -agon vertex position and moves there while assuming the color *sec*. Otherwise, it stays put. We notice that, along this phase, our algorithm takes full advantage of parallelization since the trajectory of moving robots cannot collide, as explained in the fully and semi-synchronous cases.

7. Conclusions and Research Outlooks

In this paper, we designed algorithms solving the Uniform Circle Formation (UCF) problem on opaque luminous robot swarms. Our first algorithm solves the problem for fully synchronous swarms of robots, by using a constant number of look-compute-move cycles as well as a constant number of colors. In particular, with respect to our result in [28], here we reduce the number of cycles from six to three. In addition, we also study the semi-synchronous and asynchronous cases. For the former setting, we propose an algorithm featuring a constant number of epochs; for the latter a linear amount of epochs is required in the worst case. In both settings, we use a constant number of colors.

Among many possible future researches, we feel it interesting to pinpoint connections with *Formal Language Theory*. In the literature, and in the present paper as well, some interesting formal language aspects show up and, in our opinion, deserve further and more systematic investigations. For instance, as observed, recognizing certain types of symmetries in the robot swarm displacement reduces to verifying certain properties enjoyed by angle-strings. This is equivalent to accepting certain languages, such as the palindrome language, mirror language, copy language, etc.. Therefore, well established results on the hardness of language acceptance could carry over to distributed system investigation, stating the possibility or not to solving certain problems or the minimal amount of computational resources agents must possess to correctly operate.

In particular, considering the realm of luminous robots, the communication system provided by a constant number of colors is easily seen to be modeled by a finite state automaton. This observation might suggest, e.g., that minimizing the number of colors could be related to minimizing the number of states in finite state automata. More generally, finite state automata can be examined by so many points of view: *descriptive complexity* [30, 31, 32, 33, 34], studying the size (number of states) of automata, *descriptive complexity* [35], studying automata representation by logic frameworks, *quantum computing* [36, 37, 38], studying the impact of the quantum paradigm on finite state machines size reduction. All these and other viewpoints might bring interesting insights and new tools and research problems in distributed system investigation.

Acknowledgments

The authors wish to thank the anonymous referees for their very helpful comments and remarks.

References

- [1] M. D’Emidio, D. Frigioni, A. Navarra, Synchronous robots vs asynchronous lights-enhanced robots on graphs, *Electronic Notes in Theoretical Computer Science* 322 (2016) 169–180.
- [2] M. D’Emidio, D. Frigioni, A. Navarra, Characterizing the computational power of anonymous mobile robots, in: 36th IEEE International Conference on Distributed Computing Systems (ICDCS), IEEE, 2016, pp. 293–302.
- [3] P. Flocchini, G. Prencipe, N. Santoro, Distributed computing by oblivious mobile robots, *Synthesis Lectures on Distributed Computing Theory* 3 (2012) 1–185.
- [4] P. Flocchini, G. Prencipe, N. Santoro, Distributed Computing by Mobile Entities. Current Research in Moving and Computing, volume 11340 of *LNCS*, Springer, 2019.
- [5] R. Adhikary, M. K. Kundu, B. Sau, Circle formation by asynchronous opaque robots on infinite grid, *Computer Science* 22 (2021).
- [6] K. Bose, M. K. Kundu, R. Adhikary, B. Sau, Arbitrary pattern formation by asynchronous opaque robots with lights, *Theoretical Computer Science* 849 (2021) 138–158.
- [7] K. Bolla, T. Kovacs, G. Fazekas, Gathering of fat robots with limited visibility and without global navigation, in: Int. Symposium on Evolutionary Computation/Swarm Intelligence and Differential Evolution (EC/SIDE), volume 7269 of *LNCS*, Springer, 2012, pp. 30–38.
- [8] J. Czyzowicz, L. Gasieniec, A. Pelc, Gathering few fat mobile robots in the plane, *Theoretical Computer Science* 410 (2009) 481–499.
- [9] S. Das, R. Focardi, F. L. Luccio, E. Markou, M. Squarcina, Gathering of robots in a ring with mobile faults, *Theoretical Computer Science* 764 (2019) 42–60.
- [10] S. Datta, A. Dutta, S. Gan Chaudhuri, K. Mukhopadhyaya, Circle formation by asynchronous transparent fat robots, in: 9th International Conference on Distributed Computing and Internet Technology (ICDCIT), volume 7753 of *LNCS*, Springer, 2013, pp. 195–207.
- [11] P. Flocchini, G. Prencipe, N. Santoro, P. Widmayer, Gathering of asynchronous robots with limited visibility, *Theoretical Computer Science* 337 (2005) 147–168.
- [12] K. Sugihara, I. Suzuki, Distributed algorithms for formation of geometric patterns with many mobile robots, *Journal of Robotic Systems* 13 (1996) 127–139.
- [13] I. Suzuki, M. Yamashita, Distributed anonymous mobile robots: Formation of geometric patterns, *SIAM Journal on Computing* 28 (1999) 1347–1363.
- [14] M. Yamashita, I. Suzuki, Characterizing geometric patterns formable by oblivious anonymous mobile robots, *Theoretical Computer Science* 411 (2010) 2433–2453.
- [15] G. Viglietta, Uniform circle formation, in: *Distr. Comp. by Mobile Entities. Current Research in Moving and Computing*, volume 11340 of *LNCS*, Springer, 2019, pp. 83–108.
- [16] X. Défago, A. Konagaya, Circle formation for oblivious anonymous mobile robots with no common sense of orientation, in: 2nd ACM International Workshop on Principles of Mobile Computing (POMC), 2002, pp. 97–104.
- [17] Y. Dieudonné, F. Petit, Squaring the circle with weak mobile robots, in: 19th Int. Symp. on Algorithms and Computation (ISAAC), volume 5369 of *LNCS*, Springer, 2008, pp. 354–365.
- [18] P. Flocchini, G. Prencipe, N. Santoro, G. Viglietta, Distributed computing by mobile robots: uniform circle formation, *Distributed Computing* 30 (2017) 413–457.
- [19] M. Mondal, S. Gan Chaudhuri, Uniform circle formation by swarm robots under limited visibility, in: 16th International Conference on Distributed Computing and Internet

Technology (ICDCIT), volume 11969 of *LNCS*, Springer, 2020, pp. 420–428.

- [20] K. Buchin, P. Flocchini, I. Kostitsyna, T. Peters, N. Santoro, K. Wada, Autonomous mobile robots: Refining the computational landscape, in: 2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS), IEEE, 2021, pp. 576–585.
- [21] S. Das, P. Flocchini, G. Prencipe, N. Santoro, M. Yamashita, The power of lights: synchronizing asynchronous robots using visible bits, in: 32nd IEEE International Conference on Distributed Computing Systems (ICDCS), IEEE, 2012, pp. 506–515.
- [22] G. A. D. Luna, P. Flocchini, S. Gan Chaudhuri, N. Santoro, G. Viglietta, Robots with lights: Overcoming obstructed visibility without colliding, in: 16th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS), volume 8756 of *LNCS*, Springer, 2014, pp. 150–164.
- [23] S. Das, P. Flocchini, G. Prencipe, N. Santoro, M. Yamashita, Autonomous mobile robots with lights, *Theoretical Computer Science* 609 (2016) 171–184.
- [24] G. Sharma, R. Vaidyanathan, J. L. Trahan, C. Busch, S. Rai, $O(\log n)$ -time complete visibility for asynchronous robots with lights, in: 2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS), IEEE, 2017, pp. 513–522.
- [25] G. Sharma, R. Vaidyanathan, J. L. Trahan, C. Busch, S. Rai, Complete visibility for robots with lights in $O(1)$ time, in: 18th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS), volume 10083 of *LNCS*, Springer, 2016, pp. 327–345.
- [26] G. Sharma, R. Vaidyanathan, J. L. Trahan, Constant-time complete visibility for asynchronous robots with lights, in: 19th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS), volume 10616 of *LNCS*, Springer, 2017, pp. 265–281.
- [27] A. Aljohani, G. Sharma, Complete visibility for mobile robots with lights tolerating faults, *International Journal of Networking and Computing* 8 (2018) 32–52.
- [28] C. Feletti, C. Mereghetti, B. Palano, Uniform circle formation for swarms of opaque robots with lights, in: 20th International Symposium on Stabilizing, Safety, and Security of Distributed Systems (SSS), volume 11201 of *LNCS*, Springer, 2018, pp. 317–332.
- [29] Y. Dieudonné, F. Petit, Swing words to make circle formation quiescent, in: 14th International Colloquium on Structural Information and Communication Complexity (SIROCCO), volume 4474 of *LNCS*, Springer, 2007, pp. 166–179.
- [30] A. Malcher, C. Mereghetti, B. Palano, Descriptive complexity of two-way pushdown automata with restricted head reversals, *Theoretical Computer Science* 449 (2012) 119–133.
- [31] V. Geffert, Z. Bednářová, C. Mereghetti, B. Palano, Boolean language operations on nondeterministic automata with a pushdown of constant height, in: 8th Int. Computer Science Symposium in Russia (CSR), volume 7913 of *LNCS*, Springer, 2013, pp. 100–111.
- [32] S. Jakobi, K. Meckel, C. Mereghetti, B. Palano, Queue automata of constant length, in: 15th International Workshop on Descriptive Complexity of Formal Systems (DCFS), volume 8031 of *LNCS*, Springer, 2013, pp. 124–135.
- [33] Z. Bednářová, V. Geffert, C. Mereghetti, B. Palano, Boolean language operations on nondeterministic automata with a pushdown of constant height, *Journal of Computer and System Science* 90 (2017) 99–114.
- [34] M. Kutrib, A. Malcher, C. Mereghetti, B. Palano, Descriptive complexity of iterated uniform finite-state transducers, in: 21st International Workshop on Descriptive Complexity of Formal Systems (DCFS), volume 11612 of *LNCS*, Springer, 2019, pp. 223–234.

- [35] C. Choffrut, A. Malcher, C. Mereghetti, B. Palano, First-order logics: some characterizations and closure properties, *Acta Informatica* 49 (2012) 225–248.
- [36] C. Mereghetti, B. Palano, Quantum automata for some multiperiodic languages, *Theoretical Computer Science* 387 (2007) 177–186.
- [37] M. P. Bianchi, C. Mereghetti, B. Palano, Quantum finite automata: Advances on Bertoni’s ideas, *Theoretical Computer Science* 664 (2017) 39–53.
- [38] A. Kumar, D. A. de Jesus Pacheco, K. Kaushik, J. J.P.C. Rodrigues, Futuristic view of the Internet of quantum drones: Review, challenges and research agenda, *Vehicular Communications* 36 (2022) 100487.