$\mathcal{O}(\log n)$-Time Uniform Circle Formation for Asynchronous Opaque Luminous Robots

Caterina Feletti
Dipartimento di Informatica, Università degli Studi di Milano, Italy

Carlo Mereghetti
Dipartimento di Informatica, Università degli Studi di Milano, Italy

Beatrice Palano
Dipartimento di Informatica, Università degli Studi di Milano, Italy

Abstract

We study the `Uniform Circle Formation` (`UCF`) problem for a distributed system of n robots which are required to displace on the vertices of a regular n-gon. We consider a well-studied model of autonomous, anonymous, mobile robots that act on the plane through Look-Compute-Move cycles. Moreover, robots are unaware of the cardinality of the system, they are punctiform, completely disoriented, opaque, and luminous. Collisions among robots are not tolerated.

In the literature, the `UCF` problem has been solved for such a model by a deterministic algorithm in the asynchronous mode, using a constant amount of light colors and $\mathcal{O}(n)$ epochs in the worst case. In this paper, we provide an improved algorithm for solving the `UCF` problem for asynchronous robots, which uses $\mathcal{O}(\log n)$ epochs still maintaining a constant amount of colors.

2012 ACM Subject Classification Theory of computation → Distributed algorithms; Computing methodologies → Mobile agents; Computing methodologies → Robotic planning

Keywords and phrases Autonomous mobile robots, Opaque robots, Luminous robots, Pattern formation

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2023.5

1 Introduction

One of the most studied classes of agent models examines systems (swarms) of autonomous and computational entities, called *robots*, which have to collaborate to solve a given problem without any central control [18, 19]. They operate through an infinite sequence of *look-compute-move* cycles, following one of the three activation and synchronization modes: the *fully synchronous* mode (FSYNCH), where time is divided in atomic *rounds* and all the robots execute a cycle synchronously in each round, the *semi-synchronous* mode (SSYNCH), which differs from the FSYNCH just for the fact that, in each round, a subset of robots executes the cycle synchronously whereas the others remain idle, and the *asynchronous* mode (ASYNCH), where there is no global clock and each robot acts asynchronously. For the SSYNCH and ASYNCH modes, time complexity is computed considering the number of *epochs* (rather than the number of rounds), where an epoch is a minimal time frame within which each robot is activated at least once.

For the sake of generality, most of the works in this field consider systems of robots with very limited features: they are *anonymous* and *indistinguishable*, they are *oblivious* (i.e. they do not have any persistent memory), they are *silent* (i.e. they cannot send messages), and they are *disoriented* (i.e. can have different local coordinate systems). In the SSYNCH and ASYNCH modes, robots do not have any information about the activation scheduling (e.g. which robots are activated or idle, or when an epoch starts). Another common feature is the robot *unawareness* of the cardinality of the system: whilst it can restrict the computational power of the system, such an unawareness condition makes the system scalable and open to

robot insertions or removals. Such minimalistic and unnatural models have been introduced in order to investigate the minimal sets of robot abilities required to solve a specific problem. At the same time, other feature assumptions are considered for exploring more realistic models: instead of the classic *punctiform* model, where robots are treated as points in space, *fat* models [2, 9, 5, 22] regard robots as solid discs. Also, instead of "more naive" models of *transparent* robots (enabling complete visibility of the system), and/or *incorporeal* robots (so that robots can occupy the same position without colliding), more recent models assume robots to be *opaque* [1, 3, 17] and acting within environments where collisions are not tolerated [15, 17, 23]. Mostly, models assume robots move in discrete spaces (grids or graphs) or in continuous spaces (typically in the Euclidean plane). Communication among agents has proven to be essential to solve some problems [4, 8, 11], especially in the ASYNCH mode where the robot motions typically must be "traffic lights coordinated". For this reason, literature has considered models of *luminous* robots, which are equipped with a light, and which can communicate and/or store some information through the colors the light can assume.

Pattern formation problems [3, 30, 31, 33] are of great importance and interest for autonomous swarms of robots; such problems require mobile entities to move on the given space and form a target pattern. The `Circle Formation` problem (requiring the swarm to achieve a *circle configuration*, i.e. a configuration where all robots lay on the same circle[1]) has been broadly investigated in the last two decades for various models [1, 9, 10, 14]. Thereafter, several algorithms have been designed to solve the "uniform" variant. The `Uniform Circle Formation` (`UCF`) problem [10, 12, 15, 17, 20, 26, 32] requires robots to arrange on the vertices of a regular n-gon, n being the cardinality of the swarm. The well-known geometric properties of a regular polygon (e.g. agreement on both origin and unit distance, equally-spaced distribution) make the `UCF` solution a fundamental algorithmic primitive for a distributed mobile system. Generally, an algorithm for `UCF` is decomposed into two steps [10, 15, 16, 17]: firstly, a `Circle Formation` solution is provided, then the algorithm solves the `Uniform Transformation` sub-problem [10] which asks robots from a circle configuration to equally distribute on the circle.

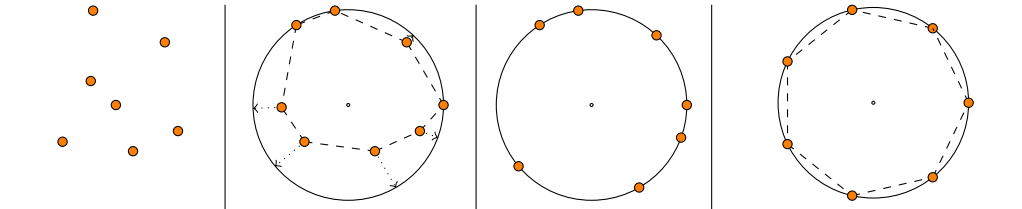
The `UCF` problem has been investigated and solved under different combinations of system features in order to point out the minimal sets of assumptions for its solution. The algorithms presented in [10] make oblivious and disoriented robots form a circle configuration and then converge toward an even distribution. The same robot model is investigated in [12] where an exact algorithm solves `UCF` for n robots, n being prime. In [25] and [26], the authors solve the problem in a swarm of fat robots, but (partially) oriented. In [20], the authors provide a solution for disoriented, and non-rigid robots, in the ASYNCH mode, avoiding collisions. In [15, 16, 17] the `UCF` problem is solved for disoriented, and opaque robots without collisions. With such strong constraints, the introduction of lights as means of communication and persistent memory turns out to be crucial. In particular, in [17] the authors provide three algorithms using $\mathcal{O}(1)$ light colors and solving the `UCF` problem in the three different synchronization modes: the ASYNCH algorithm requires $\mathcal{O}(n)$ epochs, while the synchronous algorithms solve the problem in constant time. Their FSYNCH solution improves the algorithm presented in [15] for the same model and synchronization mode. We emphasize that solving the `UCF` problem for such an opaque model but *without* lights is still an open question.

Related works and contributions. We consider the same model and the same problem as in [17]. Namely, we want to solve the `UCF` problem by opaque and luminous robots, avoiding collisions. In [17], the authors use two preliminary steps to displace the robots on the same

[1] We will always consider non-degenerate circles.

circle, upon which the regular n-gon will be formed. In particular, *(i)* they first use the algorithms in [28] (for the FSYNCH and SSYNCH modes) and [27] (for the ASYNCH mode) to displace the robots on the vertices of a convex hull, providing complete visibility to the swarm. Then, *(ii)* robots are easily made to move radially to reach their *smallest enclosing circle* (SEC). After these preliminary steps, *(iii)* authors provide original strategies to solve the `Uniform Transformation` sub-problem. Table 1 depicts the configurations obtained by executing steps *(i-iii)* in order to attain the regular polygon.

Table 1 Preliminary steps of the algorithm before reaching the Uniform Circle Configuration.



Regarding time complexity, steps *(i-ii)* need a constant number of colors and running time (rounds and epochs), in all the three synchronization modes. Concerning the original algorithms in [17] for step *(iii)*, they solve the `Uniform Transformation` sub-problem with $\mathcal{O}(1)$ colors and $\mathcal{O}(1)$ time in the FSYNCH and SSYNCH modes. On the other hand, the algorithm for step *(iii)* in the ASYNCH mode needs *linearly* many epochs in the worst case.

In this work, we present an improved solution for the `Uniform Transformation` sub-problem (i.e. step *(iii)*) in the ASYNCH mode, using a number of epochs which is *logarithmic* in the number of robots. As a corollary, this provides a $\mathcal{O}(\log n)$-time algorithm for solving the `UCF` problem from an arbitrary configuration. For the sake of conciseness, the details of steps *(i-ii)* are not repeated here: this choice allows us to focus on the novel results, without affecting the integrity and the relevance of the whole work.

Moreover, a key tool in our constructions and analysis is represented by the formalization of some geometric properties through *circular strings* [7, 21, 24, 29]. Similar approaches have been used in the realm of robot swarms [6, 12, 13]: as a matter of fact, such a formalization represents a natural and convenient tool to analyze the system configurations reached by robots. Specifically, circular strings have been used to provide robots with orientation and sorting, and to solve the *Leader Election* primitive. Moreover, specific patterns of strings have been defined to formalize particular geometric patterns: in [12], the authors solve `UCF` for asymmetric configurations by using the *Lyndon words* to elect a leader robot among the swarm. In [13], the authors introduced the *Swing words* to formalize specific configurations and solve the `UCF` problem. In this paper, we present a general discussion, providing an exhaustive and rigorous formalization of any arbitrary circle configuration through circular strings and an appropriate formal method to check the correctness of the results. As a matter of fact, several theorems and lemmas are presented (see Appendix B.2) in order to prove our strategies and outline the significant properties of robot configurations.

2 Model

We consider a system of n autonomous computational mobile robots which are *punctiform*, *anonymous*, *indistinguishable*, *homogeneous*, and *completely disoriented* (no agreement on the local coordinate system, unit distance, and chirality), operating in the Euclidean plane. They

do not know how many they are. They are *opaque*, so in the case of collinearity between three robots, the endpoint robots cannot see each other. However, they are *luminous*, i.e. they are equipped with a *persistent light* that can assume one among a constant number of colors. We say that two robots r and s *mutually see* each other if *(i)* $r = s$ or *(ii)* there is no other robot on the segment $\overline{rs}$. If r sees s, r senses only the position (in its own local coordinate system) and the color of s. Robots are given no other means to store or communicate information to the swarm. They form a distributed system where each robot executes the same *deterministic algorithm* through a sequence of *look-compute-move* cycles, in which each activated robot r takes the instantaneous snapshot (positions and lights) of the visible part of the system (*look*), executes the algorithm and computes the next position τ and light color c (*compute*), updates its light with the color c and moves straight towards τ (*move*[2]). Light color is maintained until the robot updates it in subsequent cycles. We assume that the model is *rigid*, i.e. no adversary can stop robot movement. Our robots operate in the ASYNCH mode: robots are activated independently of each other, and each *look-compute-move* cycle lasts an unpredictable but finite amount of time. Robots do not know which other robots are active at any given time. We assume the *fairness condition*: for any time t and for any robot r, there exists a time $t' > t$ such that r is activated. Our model does not tolerate either *multiplicity* (i.e. no robot can occupy the same location of another robot at the same time) or *overlapping trajectories* (robots r and s have overlapping trajectories if *(i)* r is moving from a to a', *(ii)* s is moving from b to b', and *(iii)* the segments $\overline{aa'}$ and $\overline{bb'}$ have points in common). We refer to both multiplicity and overlapping trajectories as *collisions*: hence, we aim to design algorithms that avoid collisions among robots.

3 Some notions

Given a regular n-gon, its *base angle* is the angle $\frac{2\pi}{n}$. Let $r_0, \ldots, r_{m-1}$ be m distinct robots laying ordered on the SEC according to a fixed orientation. We say that they are *consecutive* if they appear in sequence by traveling along the SEC only once. We say they are *adjacent* if, for every $0 \leq i < m-1$, no other robot lays on the arc $\overset{\frown}{r_i r_{i+1}}$.
Given a listing $r_0, \ldots, r_{n-1}$ of n adjacent robots on the SEC, centered in O, we define the sequence $\alpha_0 \cdots \alpha_{n-1}$ as the corresponding *angle-string*, where $\alpha_i = \widehat{r_i O r_{(i+1) \mod n}}$. A group of robots on an arc is *oriented* if it agrees on a common clockwise direction. Unless otherwise stated, given two points a and b on the SEC, not laying on the two different endpoints of the same diameter, we refer to the arc $\overset{\frown}{ab}$ as the *minor arc* joining a and b on the SEC.

4 Parallelism with circular strings

Once all the robots are on their SEC, we can study the geometric properties of such a disposition by considering the circular strings formed by the angle-strings in that configuration. From this viewpoint, spotting special geometric properties amounts to analyzing some properties on the circular strings.

Circular strings. Let $x = x_0 \cdots x_{n-1}$ be a *linear string* on an alphabet Σ (from now on, we simply call it *string*). We denote by $|x|$ the *length* of x, by ϵ the *empty string*, and by $x^R = x_{n-1} \cdots x_0$ the reverse string of x. A *factor* of x is ϵ or any string $x_i x_{i+1} \cdots x_j$ such that $0 \leq i \leq j \leq n-1$. We define the *k-shift* of x the string $\sigma_k(x) = x_k \cdots x_{n-1} x_0 \cdots x_{k-1}$,

[2] If τ is the current position of r, we say that r executes a *null movement*.

where $0 \leq k \leq n-1$. A string x is a *palindrome* if $x = x^R$. A *mirrored* string is a palindrome with an even length. A string x is a *power string* whenever there exists a factor y of x, such that $y \neq \epsilon$, $y \neq x$ and $x = y^k$ for an integer k. On the contrary, x is a *base string* if it is not a power string.

Given a linear string $x = x_0 \cdots x_{n-1}$, the corresponding *circular string* is the multiset $\sigma(x) = \{\sigma_0(x), \ldots, \sigma_{n-1}(x)\}$. A circular string is *minimal* if it contains only base strings. Given a string x on the alphabet Σ, we say that it is *symmetric* if *(i)* x is a palindrome, or *(ii)* $x = ayby^R$ where $y \in \Sigma^*, a, b \in \Sigma$. A circular string is said to be *symmetric* if it contains a symmetric string.

Symmetries and circular strings. Given a configuration $\mathcal{C}$ where all the robots are on the SEC and given a fixed orientation (w.l.o.g. clockwise), let us consider the listing of adjacent robots $r_0, \ldots, r_{n-1}$ on the SEC with the corresponding angle-string $\alpha = \alpha_0 \cdots \alpha_{n-1}$. From each robot r_i, two angle-strings start: $\sigma_i(\alpha)$ (clockwise) and its reverse $\sigma_{n-i}(\alpha^R)$ (counterclockwise). We denote these relations with the following notation: $r_i \curvearrowright \sigma_i(\alpha)$ and $r_i \curvearrowleft \sigma_{n-i}(\alpha^R)$. We call *configuration strings* of $\mathcal{C}$ the circular string $\sigma_{\mathcal{C}} = \sigma(\alpha)$ and $\sigma_{\mathcal{C}}^R = \sigma(\alpha^R)$. According to the properties of $\sigma_{\mathcal{C}}$ (clearly the same holds for $\sigma_{\mathcal{C}}^R$), the configuration $\mathcal{C}$ comes in three different cases of *geometric* symmetry:

- in case of asymmetry, $\sigma_{\mathcal{C}}$ is minimal and not symmetric;
- in case of symmetry with one axis, $\sigma_{\mathcal{C}}$ is symmetric. In particular, three sub-cases exist according to the number of *axis robots* (i.e. robots laying on the endpoints of the symmetry axis):
  - one axis robot (odd n), $\sigma_{\mathcal{C}}$ contains a palindrome in the form xax^R;
  - two axis robots (even n), $\sigma_{\mathcal{C}}$ contains two mirrored strings in the form xx^R and x^Rx;
  - zero axis robots (even n), $\sigma_{\mathcal{C}}$ contains two symmetric strings in the form $axbx^R$ and bx^Rax.

  In all the above three sub-cases, no other symmetric strings are contained in $\sigma_{\mathcal{C}}$;
- in case of rotational symmetry, all the strings in $\sigma_{\mathcal{C}}$ are power strings in the form x^k where x is a base string.

5 The algorithm

Let us consider a configuration $\mathcal{C}$ where all the n robots are on the same smallest enclosing circle (SEC). We explain the algorithm for the `Uniform Trasformation` sub-problem, by displacing robots on a regular n-gon inscribed on the SEC, in the ASYNCH mode. Throughout the explanation of the algorithm, we will always be using the terminology "the SEC" by meaning the *original* SEC in the configuration $\mathcal{C}$.

Regular tuple. Let $\mathcal{P}$ be the target regular n-gon, which is uniquely determined by $\mathcal{C}$ according to our algorithm. Our algorithm consists of two initial tasks where triples or 4-tuples of robots will move to some vertices of $\mathcal{P}$ and set their lights with special colors[3] (*pivot* and *angle*) to communicate "we are the reference robots". In particular, we call *regular tuple* the tuple of consecutive robots in the form *(angle, pivot, angle)* or *(angle, pivot, pivot, angle)*, laying on adjacent vertices of $\mathcal{P}$. Accordingly, it holds that, if $(r_1, \ldots, r_k)$ is a regular tuple on the SEC centered in O (with $k \in \{3, 4\}$), then $\widehat{r_i O r_{i+1}} = \frac{2\pi}{n}$ for $1 \leq i < k$.

[3] We use the *italic* notation to denote the exact color of a robot (e.g. *guard, guard_c, pivot* ...), whereas we indicate the role of a robot by the normal text (e.g. guard, pivot ...). Multiple colors can be used for the same role (e.g. a guard robot can assume the color *guard, guard_c, moving_guard* ...).

Splitting chord. Let τ, τ' be two regular tuples such that a, a' (resp. b, b') are the points where the angle robots of τ (resp. τ') lay on, such that a, a', b, b' are consecutive. Let us consider the two disjoint arcs $\overset{\frown}{ab'}$ and $\overset{\frown}{ba'}$. If in at least one of the two arcs, no regular tuple is formed (or must be formed), we say that τ and τ' are adjacent. Given two adjacent regular tuples τ, τ' defined as above, let m_a (resp. m_b) be the middle point on the minor arc aa' (resp. bb'). We call the chord $m_a\bar{m}_b$ as *splitting chord* of τ, τ'. Given a splitting chord ρ on the SEC, we call *performance arc* of ρ, denoted as $\overset{\frown}{\rho}$, the minor arc cut by ρ. We call *performance area* of ρ the region of the plane φ_ρ such that *(i)* it contains the performance arc $\overset{\frown}{\rho}$ and *(ii)* it is delimited by ρ and the two distinct straight lines, perpendicular to ρ, each one passing through one of the two endpoints of ρ. Figure 1 shows the explained elements. Note that, if ρ is a diameter, it defines two opposite performance arcs/areas.

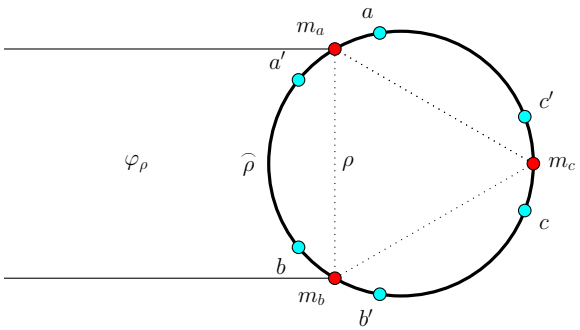


Figure 1 A splitting chord ρ, its performance arc $\overset{\frown}{\rho}$, and its performance area φ_ρ. The configuration has three splitting chords, defined by three regular triples in the form *(angle, pivot, angle)*.

Summary of the algorithm. We list the main steps and strategies used in our algorithm.

- At first, we split the SEC into performance arcs of equal length and robot cardinality, such that all the robots on each arc (or on each half-arc) are oriented. In this part, we use circular strings to formalize the geometric properties of the configurations.
- To fix the performance arcs, some robots per arc are elected and made to move in specific positions on the SEC, around each arc endpoint. These reference robots will form the regular tuples which will be used by a robot r to *(i)* reconstruct the original SEC where the regular n-gon $\mathcal{P}$ must be formed, *(ii)* recompute its base angle and so the global number of robots in the system, *(iii)* determine the performance area where r has to move, and *(iv)* detect the group of robots (on the same performance area of r) with which r has to collaborate.
- Performance areas partition the swarm into groups of robots. Each group will act independently of the others, in its own original performance area. This allows the next steps to be executed in parallel by each group.
- Within each performance area, robots equally distribute themselves on the performance arc by executing the same routine. This routine includes a loop that iterates a sequence of nine tasks, where each task is fulfilled within a constant number of epochs. Since the sequence of tasks is iterated an amount of time which is *logarithmic* in the number of robots acting in the performance area, we obtain an exponential decrease in the running time, when compared to the existing algorithms for `UCF`. Such a routine can be adapted into a stand-alone algorithm that uniformly arranges robots along a given arc.

- The time-complexity improvement from $\mathcal{O}(n)$ (see [17]) to $\mathcal{O}(\log n)$ is given by the "pairing technique" used in the task loop: robots pair themselves and use their mutual distance to encode all the needed information to reach their target destinations.

Asynchronism. The main issue with the ASYNCH mode with opaque and collision-intolerant robots is the management of moving robots which can *(i)* hide or *(ii)* be hidden by other robots. To cope with the issue *(i)*, our algorithm makes a robot color itself as *moving_x* before starting moving, where x will be the color it has to set once stopped and reactivated. A task of our algorithm is said to be *fully parallelized* if robots can safely execute their move phase (update their color or position) even if they see other *moving* robots. Instead, given a region ω on $\mathbb{R}^2$, we say that a task for a subset of robots S is ω *-synchronized* if it requires any robot $r \in S$ to skip its move phase if r sees a *moving_* colored robot in ω, different from itself. The region of synchronization of a robot can be the whole $\mathbb{R}^2$ plane or its performance area, according to the task to be executed. For the issue *(ii)*, our algorithm guarantees no collinearity with moving robots creating ambiguous snapshots. Let us show the algorithm task by task. For the sake of brevity, our explanation will omit the detail of the *moving_* colors, assuming their need and logic as understood.

5.1 SEC splitting

In the first task, some robots have to set their lights to fix the splitting chords. Moreover, at most one robot is required to travel, in a particular configuration. Starting from the configuration $\mathcal{C}$, our strategy is to select the splitting chords which split the SEC into performance arcs with the same length and with the same number of robots (possibly with just one robot of difference). The method to select such arcs depends on the geometric symmetry degree of $\mathcal{C}$.

Asymmetry. In this case, every robot can elect the lexicographically smallest string y over $\sigma_{\mathcal{C}} \cup \sigma_{\mathcal{C}}^R$ and so the robot p from which y starts (i.e. either $p \curvearrowright y$ or $p \curvearrowleft y$), such that the diameter passing through p splits the SEC into two halves with the same robot cardinality (except for just one exceeding robot at most). Note that such a diameter always exists (see Theorem 1 in [17]). In this case, the selection of y is always taken unambiguously (see Lemma 5 in Appendix B.2). The diameter passing through p is elected as the splitting chord for $\mathcal{C}$.

In this task, p sets its light as *pivot* and fixes the splitting chord. If n is even, on the other endpoint e of the splitting chord, a robot is elected to move to e where it will assume the color *pivot_a* once reactivated. For this purpose, we elect the robot already on e, if it exists, otherwise we elect the closest robot to e belonging to the most populated half-SEC.

Symmetry with just one axis. In this case, robots elect the diameter on the axis of symmetry as the splitting chord. The elected splitting chord, say d, can pass through 0, 1, or 2 opposite robots. In this task, the robots laying on d begin pivots and set their light as *pivot*. If no robot lays on d, we have to color just two robots, symmetric to d, with the color *placeholder*, to univocally fix the splitting chord (in fact, given two distinct points on a circle, there exists a unique axis of symmetry for the two points which does not pass through them). To establish an unambiguous method to elect the symmetric placeholders, we can elect the farthest robots to d (in case of equal distance, we choose according to the lexicographic order of the half-SEC angle-string).

Rotational symmetry. In this case, all strings in $\sigma(\mathcal{C})$ are power strings. Let $\alpha = \alpha_0 \cdots \alpha_{n-1}$ be an angle-string in $\sigma(\mathcal{C})$, and let β be the base string such that $\beta^k = \alpha$ (the configuration $\mathcal{C}$ can be divided into $k > 1$ identical sectors, each being the $\frac{2\pi}{k}$-rotation of the previous one). We call $\sigma(\beta)$ and $\sigma(\beta^R)$ the *sector circular strings* of the configuration. We note that they are *minimal*. Consider the ordered list of the adjacent robots on the SEC $r_0, \ldots, r_{n-1}$ which forms α. Let $P_i = \{r_j \mid j \equiv i \mod \frac{n}{k}\}$ be the *class of symmetry* which contains k robots sharing the same position in the k different sectors. We can observe that, for each robot $r \in P_i$, $r \curvearrowright \sigma_i(\alpha)$ and $r \curvearrowleft \sigma_i(\alpha)^R$ hold true. Our strategy now selects one or two classes of symmetry among $P_0, \ldots, P_{\frac{n}{k}-1}$ according to a particular property of their associated angle-strings. Observe that

▶ **Observation 1.** *Since each angle-string $\sigma_i(\alpha)$ (resp. $\sigma_i(\alpha)^R$) is the k-power of $\sigma_i(\beta)$ (resp. $\sigma_i(\beta)^R$), we can analyze the properties of the angle-strings just taking into account their factors $\sigma_i(\beta)$ and $\sigma_i(\beta)^R$.*

We say that P_i *reads* $\sigma_i(\beta)$ and $\sigma_i(\beta)^R$ (formally $P_i \curvearrowright \sigma_i(\beta)$ and $P_i \curvearrowleft \sigma_i(\beta)^R$) since the robots in P_i are the starting points for these strings (in opposite directions).

According to this, our algorithm unambiguously chooses *just one* string, say $\tilde{\gamma}$, within $\sigma(\beta) \cup \sigma(\beta^R)$, which is read exactly from *one or two* classes of symmetry. In particular, let Γ_1 and Γ_2 be the sets of strings in $\sigma(\beta) \cup \sigma(\beta^R)$, such that Γ_1 (resp. Γ_2) contains the strings read by just one class (resp. 2 classes) of symmetry. In particular, we can have two scenarios, according to the size of Γ_1:

- if $\Gamma_1 \neq \emptyset$, we unambiguously select a string $\tilde{\gamma}$ from Γ_1 (e.g. the lexicographically smallest one) and the class of symmetry, say P_i, which reads $\tilde{\gamma}$;
- if $\Gamma_1 = \emptyset$, we properly[4] select a string $\tilde{\gamma}$ from Γ_2 and the two classes of symmetry, say P_i and P_j, which read $\tilde{\gamma}$.

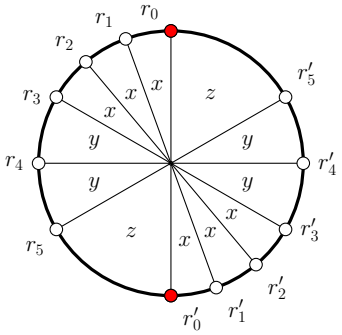
We call P_i and, if it exists, P_j *eligible classes of symmetry*. Note that we cannot have more than two classes of symmetry reading the same string in $\sigma(\beta)$ and $\sigma(\beta^R)$, otherwise the sectors would contain sub-sectors in turn, i.e. β would not be a base string (see Lemma 6 in Appendix B for the proof). Other properties about robot classes of symmetry (in particular about Γ_1, Γ_2, and so the presence of one or two eligible classes of symmetry) are explained in Appendix B.2. For example, note that, if $|\beta|$ is odd or $\sigma(\beta)$ contains a mirrored string, then Γ_1 contains at least a string, i.e. we can elect a unique eligible class (see Theorem 7 and Theorem 9 in Appendix B.2 for the proofs). Let us now show how the SEC is split in both scenarios (see Figures 2a and 2b).

One eligible class of symmetry: Each robot computes $\sigma(\beta)$ and $\sigma(\beta^R)$, and selects $\tilde{\gamma}$ and P_i as explained above. The k robots in P_i form the pivots (which won't move for the whole algorithm) by setting their color as *pivot*. The chords joining pivots belonging to adjacent sectors will be the splitting chords.

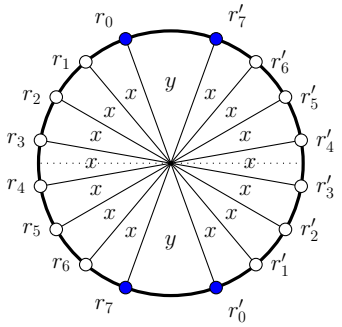
Two eligible classes of symmetry: Each robot computes $\sigma(\beta)$ and $\sigma(\beta^R)$ and selects properly $\tilde{\gamma}$, P_i, and P_j. The $2k$ robots in $P_i \cup P_j$ play the role of placeholders and set their color as *placeholder*. At the end of this task, there will be $2k$ placeholders, two for each sector. Moreover, each placeholder forms two different angle-strings with its adjacent[5] placeholders. Hence, it is important to share a common method to pair adjacent placeholders in k disjoint pairs; such pairs will be used in the next task to define the splitting chords and so delimit the k performance areas. In fact, the chords passing

[4] In order to guarantee complete visibility during regular tuple setting.

[5] Two placeholders p and r are adjacent if no other placeholder lay on the arc $\overset{\frown}{pr}$.



(a) Pivot coloring in a 2-rotation configuration with 6 classes of symmetry P_i, $i \in \{0, \dots, 5\}$. Here, $\tilde{\gamma} = x^3y^2z$ starts from just one eligible class $P_0 = \{r_0, r'_0\}$. The splitting chord is the diameter joining the two pivots.



(b) Placeholder coloring in a 2-rotation configuration with 8 classes of symmetry P_i, $i \in \{0, \dots, 7\}$. Here, $\tilde{\gamma} = x^7y$ starts from two eligible classes $P_0 = \{r_0, r'_0\}$ and $P_7 = \{r_7, r'_7\}$. The dotted diameter is elected as the splitting chord.

Figure 2 SEC splitting in the rotation case.

through the middle points of the selected pairs of placeholders will be the splitting chords. Between the two different ways to pair adjacent placeholders, robots can always choose a proper and unambiguous method to form the pairs (and so the splitting chords) so that, in the next task, the regular tuples can be formed safely. Note that for each pair of adjacent placeholders, an axis of symmetry passes between them (see Theorem 10 in Appendix B.2). Moreover, no robot lays on the endpoint of these axes of symmetry (see Theorem 11 in Appendix B.2).

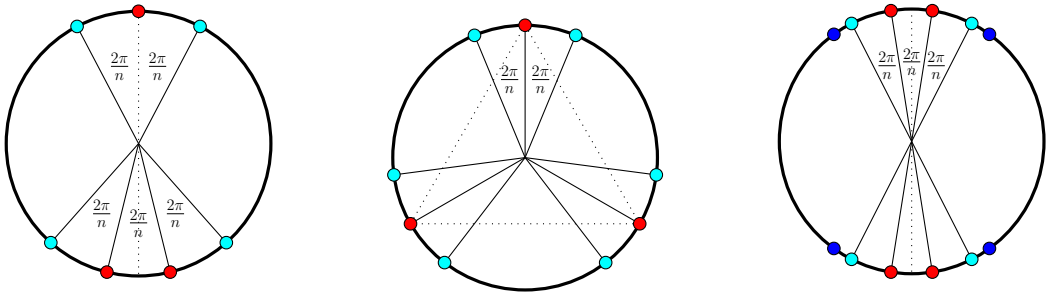
5.2 Regular tuple setting

This task requires some robots to change color and move for completing the regular tuples. The positions of these tuples are fixed by the presence of the pivots or the placeholders which have been set in the previous task. In particular, assume the current configuration presents k splitting chords $\rho_1, \dots, \rho_k$ defined by pivots and placeholders. From now on, no robot will cross the splitting chords: each robot remains in its original performance area.

Consider a splitting chord ρ, its two endpoints e_0, e_1, and its performance area φ_ρ. In this task, one or two robots per each endpoint e_i are elected on $\overset{\frown}{\rho}$, and move to complete the regular tuple around e_i in the form $(angle, pivot, angle)$ or $(angle, pivot, pivot, angle)$. The first case arises when the pivot already lays on e_i, so one robot on $\overset{\frown}{\rho}$ elects itself as an angle robot, moves in a position a on $\overset{\frown}{\rho}$ where it will set its light as *angle* for forming the base angle $\frac{2\pi}{n}$ with the pivot. Note that a second angle robot will do the same in the adjacent performance area, completing the regular triple centered on e_i. The second case arises when no robot lays on e_i, and ρ is held by *(i)* the pairs of placeholders or *(ii)* the pivot on the opposite endpoint e_{1-i} (cases of asymmetry and symmetry with odd n). In this case, two robots on $\overset{\frown}{\rho}$ elect themselves as pivot and angle, must set their light properly, and move to complete the 4-tuple centered on e_i. Figure 3 shows the regular tuple setting for different configuration types.

This task is φ_ρ -synchronized, i.e. a robot r in φ_ρ skips its move phase only if r *(i)* is not *moving_* colored and *(ii)* sees a *moving_* robot belonging to φ_ρ. In fact, other moving robots do not prevent r from executing its move phase. Note that there always exists an unambiguous strategy used by the swarm to *(i)* elect the robots which will become angle

robots and, possibly, pivots, *(ii)* make them move without colliding and always guaranteeing complete visibility to the swarm, and *(ii)* completing the k regular tuples in a constant number of epochs, potentially in parallel (see Theorem 2 in Appendix B). Eventually, the *pivot_a* robot (asymmetry case) turns its light into *pivot* aligning its color with all other pivots. Once all the regular tuples have been set, all pivots and angle robots will do nothing. Placeholders can turn off their lights since their role is no longer needed.



(a) Regular tuple setting in case of symmetry with one axis. The splitting chord lays on the axis of symmetry.

(b) Regular tuple setting in case of 3-rotation (with one eligible class). The splitting chords join the three pivots.

(c) Regular tuple setting in case of 2-rotation (with two eligible classes), around the pairs of placeholders.

Figure 3 Regular tuple setting with pivots (here red), angles (cyan), and, possibly, placeholders (blue).

5.3 Task loop

Let φ_ρ be a performance area cut by the splitting chord ρ. Let $\overset{\frown}{\rho}$ be its performance arc on the SEC, and let β_ρ be the angle-string related to the robots on $\overset{\frown}{\rho}$. If β_ρ is not a palindrome, we can establish an "upper" endpoint of $\overset{\frown}{\rho}$ according to the lexicographical orientation given by β_ρ. Otherwise, each half-arc of $\overset{\frown}{\rho}$ defines its nearest endpoint of ρ as its upper part.

Let m be the number of robots on $\overset{\frown}{\rho}$ which are not pivots or angle robots. We now show the sequence of tasks that has to be iterated until all the m robots have been moved to two particular lines external to the SEC, and parallel to ρ. As we will see, at each iteration half of the robots on $\overset{\frown}{\rho}$ leave the arc and move towards these lines, thus leading to logarithmic behavior for the whole algorithm. Note that no robot invades another performance area and the target polygon vertex of each robot is located on its performance arc.

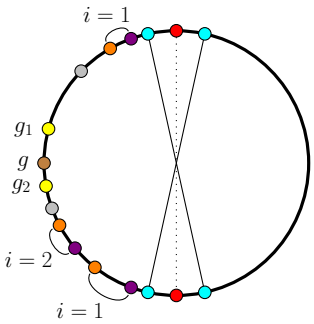
Let k be the number of performance areas on the SEC. The k groups of robots perform the same sequence of tasks independently and possibly in parallel. However, some critical tasks of the loop are φ_ρ-synchronized. Now, let us explain in detail each task of the loop (see Algorithm 1 in Appendix A for the pseudo-code of the whole task loop).

TBLG setting. In this φ_ρ -synchronized task[6], just 1 or 2 robots have to move on $\overset{\frown}{\rho}$. Let g be the middle point on $\overset{\frown}{\rho}$. We have to set 3 (if m is odd) or 2 (if m is even) guard robots around g, on the SEC, in this way:

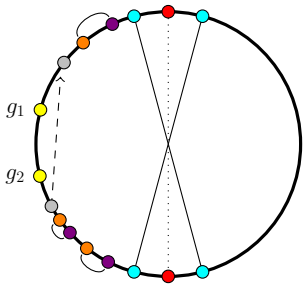
[6] The task name derives from the robot roles: top, bottom, loner, and guard.

- if m is odd, the robot closest to g (in case of two equally distant robots, the robot in the upper part of the arc is elected) moves to g and sets its color as *guard_c*[7]. We call this guard the central guard;
- the two robots closest to g (besides the one which has to head to g when m is odd) must set themselves with the *guard* color. If both are located in the same half-arc, the farthest from g must travel in the opposite half-arc, at a distance d from g which is a fraction of the minimum distance between any two robots on $\overset{\frown}{\rho}$ (this measure assures no collisions can occur). These two guards are called outer guards.

Note that no robots lay between two guards (except the central guard if m is odd). All the other robots in $\overset{\frown}{\rho}$, except for the pivots, the angles, and the guards, will color themselves according to the following roles. For each half-arc split by g, the robots pair themselves starting from the upper part of the half-arc (where the regular tuple lays): each pair is composed of the *top* robot and the *bottom* robot (in each top-bottom pair, the top robot is the closest robot to the related pivot). Each top-bottom pair must not have other robots between them. If, for example, there is an odd number of robots between the pivot and the related angle robot, the robot closest to the angle robot remains unpaired. The unpaired robots color themselves as *loner*. The final configuration is shown in Figure 4.



(a) TBLG setting with three guards. Indexing of the top-bottom pairs.



(b) TBLG disposition with two guards, and loner pairing.

Figure 4 Setting of top (here violet), bottom (here orange), loner (here lightgray), and guard robots. Top-bottom pairs are highlighted through arcs.

Loner pairing. In the previous task, at most three loners per half-arc of $\overset{\frown}{\rho}$ are created: one between the endpoint of $\overset{\frown}{\rho}$ and the pivot of the half-arc (if the pivot does not lay on ρ), one between the pivot and the angle robot, and the last one between the angle robot and its closest guard. In this task, if a half-arc presents more than one loner, the two uppermost loners (according to the upper part of each half-arc) pair themselves: the uppermost loner stays still and lights itself as *top*, while the second-uppermost one moves towards a position on the SEC in order be down with respect to the new top robot, at a relative distance from it (e.g. a fraction of the minimum distance between two robots on $\overset{\frown}{\rho}$). If just two loners remain (one per half-arc) and there is an unambiguous method to elect one of them, the elected loner will become a top robot while the second loner will approach it (using the same strategy as before) and become its bottom (see Figure 4b). This task aims to reduce the number of loners, maximizing the number of top-bottom pairs.

[7] If β_ρ is a mirrored string, then the guard is already on g.

Note that this task is φ_ρ-synchronized. At the end of this task, at most two loner robots (one per each half-arc of $\widehat{\rho}$) can exist.

Δ setting. In this φ_ρ -synchronized task, the outer guards g_1, g_2 have to approach symmetrically around g (the middle point of $\widehat{\rho}$). Let η be the minimum distance between two robots on $\widehat{\rho}$. If χ is the tangent line to the SEC passing through g, let μ be the length of the shortest projection of a segment $g\bar{g}_i$ on χ, for $i \in \{1, 2\}$. Let t be the position of the vertex of the target polygon $\mathcal{P}$ which is closest to g, but not laying on g. Let ζ be the length of the projection of the segment $\bar{gt}$ on χ. Then, each outer guard g_i moves to a position g'_i on the SEC such that g'_1 is symmetric to g'_2 with respect to g, and such that the distance between g'_1 and g'_2 is $\Delta = \min\{\eta, 2\mu, 2\zeta\}$. This measure assures *(i)* no guards turn away from g and risk colliding with other robots on the arc, and *(ii)* no guards will collide against other robots that will be moved out of the SEC in the next tasks. The measure Δ will be used as a shared value by the other robots on φ_ρ.

Pair approaching. In this fully parallelized task, every top robot r approaches its bottom robot s by traveling straight towards a point on the SEC so that the distance between r and s is δ, defined as

$$\delta = \frac{\Delta}{\langle n, m', d', i, t, c, z\rangle}$$

where

- Δ is fixed by the outer guards,
- n is the total number of robots (fixed by the base angle in the regular tuple),
- m' is the original number of robots on the arc (pivots, angles, and guards excluded),
- d' is the index of the current iteration of the loop (so, $d' = 1, \ldots, \lceil \log_2 m' \rceil$),
- i is the incremental index of the top-bottom pair starting from each endpoint of $\widehat{\rho}$ (see Figure 4a), such that the pairs closest to each endpoint have index 1 (so, $i = 1, \ldots, \lceil \frac{m'}{4} \rceil$),
- t is the number of top robots in the performance area,
- c is a flag in $\{0, 1\}$ such that if $c = 0$ then the target robot vertex is on the current half-arc where r already sits, otherwise the target vertex of r is on the other half-arc of $\widehat{\rho}$,
- z is a flag in $\{0, 1, 2\}$ such that, if $\widehat{\rho_{1/2}}$ is the half-arc of $\widehat{\rho}$ where the target vertex for r is located, then
  - $z = 0$ if the endpoints of ρ are either both covered by pivots or no pivot lays on the endpoints;
  - otherwise, $z = 1$ if the external endpoint of $\widehat{\rho_{1/2}}$ is covered by a pivot[8];
  - otherwise, $z = 2$ (i.e. if no pivot lays on the external endpoint of $\widehat{\rho_{1/2}}$).
- $\langle\ \rangle : \mathbb{N}^7 \to \mathbb{N}$ is the Cantor tuple function[9].

Note that each robot on the SEC has complete visibility of all the robots on or inside the SEC since top robots do not create collinearity with them during their movements. Note also that the setting of Δ guarantees that all top robots have to approach (and not turn away) the related bottoms to form the distance δ, avoiding collisions with other robots on

[8] The external endpoint of $\widehat{\rho_{1/2}}$ is the endpoint farthest from the guards.
[9] $\langle\ \rangle$ can be any isomorphism between $\mathbb{N}^7$ and $\mathbb{N}$.

the arc. Moreover, at the end of this task, the distance between each top and bottom robot of the same pair is always smaller than the distance between the top and the bottom robot belonging to different pairs.

Let us show how to compute d'. At each loop iteration, all the top robots are moved out of the SEC, along a specific line; namely half of the robots on the SEC leave the circle at each iteration. So each robot r on $\overset{\frown}{\rho}$ must compute the "degree of splitting" d in this way: it obtains n (from the base angle), the number k of performance areas (from the number of regular tuples), and so the number m' (where $m' < m < \frac{n}{k}$) of the original robots in its performance arc which are not pivots, angle robots or guards. Then, r counts the current q robots (pivot, angle, and guard robots excluded) in its performance arc, and computes $d = \lceil \log_2 m' \rceil - \lceil \log_2 q \rceil$, which corresponds to the index of the last iteration of the loop. Now, r knows it is starting the d'-th iteration of the loop, where $d' = d + 1$.

Top robots departure to h. Let R be the length of the radius of the SEC. Let h be the straight line parallel to the splitting chord ρ, external to the SEC, and at distance R from g (the middle point on $\overset{\frown}{\rho}$ between the outer guards). The woken top robots on $\overset{\frown}{\rho}$ can compute h, and move to this line perpendicularly, in a fully parallelized schema. Note in fact that moving robots do not obstruct other top robots from computing the next action: in fact, all the needed information to act properly is located on their performance arc.

Guard green light on. The guards can check if all the top robots have moved on h. If this is the case, the outer (resp. central) guards color themselves as *guard_green* (resp. *guard_c_green*) to tell robots on h they can proceed with the next step.

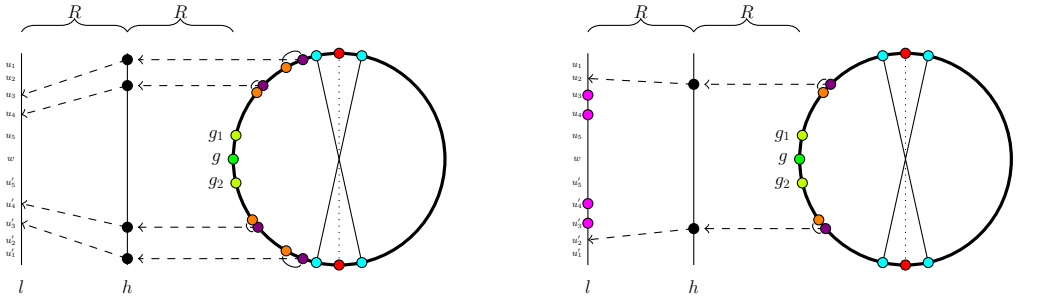
Top robots on l. Let r be a woken top robot on h. It can recompute the SEC (it can see at least two guards and at least a bottom robot), its original position on the SEC, and so the distance δ from its related bottom robot. Then, r decodes δ and recomputes the values of n, m', t, c, z, the current loop index d', and its pair index i.
Let l be the straight line parallel to h, external to the SEC, at distance R (resp. $2R$) from h (resp. the SEC). Let $\mathcal{P}$ be the target regular n-gon to be built on the SEC. Suppose m is even and let $v_1 \dots v_{\frac{m}{2}} v'_{\frac{m}{2}} \dots v'_1$ be the list of adjacent vertices of $\mathcal{P}$ on $\overset{\frown}{\rho}$ which are not already occupied by pivots and angle robots, such that they are equally distributed on the two distinct performance half-arcs. Note that the odd case is equivalent, with $v_1 \dots v_{\lfloor \frac{m}{2} \rfloor} w v'_{\lfloor \frac{m}{2} \rfloor} \dots v'_1$ as list of the vertices[10], where $w = v_{\lceil \frac{m}{2} \rceil} = v'_{\lceil \frac{m}{2} \rceil}$. Let us show how r can compute its vertex index. Thanks to the flag c, r can understand if its target vertex on $\mathcal{P}$ is on its original half-arc or on the other half-arc. This information is essential since two robots with the same index i can lay on the same half-arc: to avoid collisions, they have to know which half-arc the index refers to. Also the flag z is needed to compute the right position of the target vertex of r. Let $v_1 v_2 \dots v_{\frac{m}{2}}$ be the ordered list of the missing vertices on the target half-arc and let $u_1 u_2 \dots u_{\frac{m}{2}}$ be their projections on l. So, r chooses the vertex v_k where

$$k = \left\lfloor \frac{m}{2} \right\rfloor - \sum_{j=2}^{d'} \left\lfloor \frac{m}{2^j} \right\rfloor - \left\lceil \frac{t}{2} \right\rceil + i - 1$$

[10] As we will explain later, the vertex w is intended for the central guard.

and travels to the vertex projection u_k setting its light as *projection*. Note that this task is fully parallelized: in fact, r can compute l and δ even though all other top robots are moving toward l, since all the needed information is located "behind" r (i.e. on the SEC).

By following this schema, at each loop iteration, top robots dispose themselves on l symmetrically with respect to the central positions $u_{\frac{m}{2}}$ and $u'_{\frac{m}{2}}$, by covering the most internal free-robot projections and by leaving the two or three central positions which will be intended for the guards (see Figures 5a and 5b).



(a) Top robots travel from the SEC, to h, and then to l (after the guards have set the green light). Iteration $d' = 1$.

(b) Top robots departure from the SEC to h, and from h to l. Iteration $d' = 2$.

Figure 5 Top robots travelling from the SEC to h, and from h to l. The central positions u_5, w, u'_5 will be covered by the guards.

Gap fix. This task is executed just when on $\widehat{\rho}$ there is an odd number of bottom robots. In this case, after the previous task, there exists a projection, e.g., u_k on l where a robot lays, whereas its symmetric position u'_k is empty. From this configuration, a bottom robot can be easily elected[11] to cover the gap on l. Thus, it heads straight to u'_k and sets its color as *projection*. Note that in no other cases there exists a symmetry gap on l.

Guard green light off. The guard robots can check if no top robots still lay on h, and if, possibly, the symmetry gap on l has been covered. If so, the outer (resp. central) guard robots set their light as *guard_end* (resp. *guard_c_end*).

Loop. At this point, the algorithm re-executes the previous sequence of tasks $\mathcal{O}(\log_2 m)$ times, until all the robots on $\widehat{\rho}$ (which are not pivots, angle, or guard robots) displace themselves on the line l. Note that, in the TBLG setting, the guards do not change their positions after the first iteration, and the fixed Δ still remains the minimum distance between two robots on the same half-arc, before the *pair approaching* task.

5.4 Guards departure

At this point, the guards are the only robots on $\widehat{\rho}$ together with pivots and angle robots. Let R be the length of the radius of the SEC. Of course, the guards can compute it. In this task, each guard moves perpendicularly to l, and reaches a straight line f parallel to l and

[11] In fact, we can elect the closest robot to the guards that belongs to the half-arc with more bottom robots.

at distance R (resp. $3R$) from l (resp. from the SEC). They maintain their original colors *guard* and *guard_c* once on f. Note that their trajectories do not cross any *projection* robots on l thanks to how Δ has been chosen.

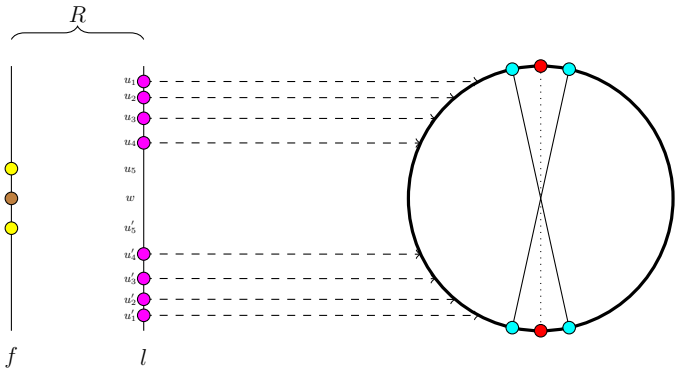


Figure 6 Robots migration from l to their target vertex on the SEC.

5.5 Back to the SEC

When all the *projection* robots on l see no more robots on the SEC (pivots and angle robots excluded), they can go back to the SEC traveling perpendicularly with respect to the line l they are placed on. Once on the SEC, they change their color in *sec* (see Figure 6).

Note that this task is fully parallelized since it is always possible for a robot r on l to reconstruct the original SEC since:

- r can see at least 3 robots on it, or
- if moving robots hide the SEC, r looks at the guards: it reconstructs f and l, and so the radius length R of the SEC and the position of its center.

5.6 Guards back to the SEC

Lastly, the guards reach the missing vertices in $\widehat{\rho}$, completing the section of the regular polygon. This task is φ_ρ -synchronized since moving guards can prevent the other guards on f to compute the needed information to act properly.

6 Conclusions

We have considered a system of n autonomous robots sharing the same features as [17]: punctiform, disoriented, indistinguishable, anonymous, opaque, and luminous. We have presented an algorithm for the UCF problem in the ASYNCH mode, which uses a constant number of colors and $\mathcal{O}(\log n)$ number of epochs. This work improves the previous solution in [17], which uses a linear number of epochs in the worst case. Note that the ASYNCH algorithm in [17] derives from an adaptation of a $\mathcal{O}(1)$-time algorithm for synchronized robots. Passing from synchronism to asynchronism, in fact, can require a fully-parallelizable algorithm to be executed within a one-by-one scheme, where each robot must wait until no other robot is moving before traveling. In this paper, we have designed an *ad-hoc* solution for asynchronous robots, where we have achieved a partial, but remarkable, parallelization degree. The natural investigation to pursue now concerns a possible fully-parallelizable algorithm (which uses a constant number of epochs) in the ASYNCH mode.

References

1 Ranendu Adhikary, Manash Kumar Kundu, and Buddhadeb Sau. Circle formation by asynchronous opaque robots on infinite grid. *Comput. Sci.*, 22(1), 2021. doi:10.7494/CSCI.2021.22.1.3840.

2 Kálmán Bolla, Tamás Kovács, and Gábor Fazekas. Gathering of fat robots with limited visibility and without global navigation. In *International Symposium on Swarm and Evolutionary Computation, SIDE 2012*, pages 30–38. Springer, 2012. doi:10.1007/978-3-642-29353-5_4.

3 Kaustav Bose, Manash Kumar Kundu, Ranendu Adhikary, and Buddhadeb Sau. Arbitrary pattern formation by asynchronous opaque robots with lights. *Theor. Comput. Sci.*, 849:138–158, 2021. doi:10.1016/J.TCS.2020.10.015.

4 Kevin Buchin, Paola Flocchini, Irina Kostitsyna, Tom Peters, Nicola Santoro, and Koichi Wada. Autonomous mobile robots: Refining the computational landscape. In *35th International Parallel and Distributed Processing Symposium Workshops, IPDPS Workshops 2021*, pages 576–585. IEEE, 2021. doi:10.1109/IPDPSW52791.2021.00091.

5 Sruti Gan Chaudhuri and Krishnendu Mukhopadhyaya. Leader election and gathering for asynchronous fat robots without common chirality. *J. Discrete Algorithms*, 33:171–192, 2015. doi:10.1016/J.JDA.2015.04.001.

6 Mark Cieliebak and Giuseppe Prencipe. Gathering autonomous mobile robots. In *9th International Colloquium on Structural Information and Communication Complexity, SIROCCO 2002*, pages 57–72. Carleton Scientific, 2002.

7 James D. Currie and D. Sean Fitzpatrick. Circular words avoiding patterns. In *6th International Conference on Developments in Language Theory, DLT 2002*, pages 319–325. Springer, 2002. doi:10.1007/3-540-45005-X_28.

8 Shantanu Das, Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Masafumi Yamashita. Autonomous mobile robots with lights. *Theor. Comput. Sci.*, 609:171–184, 2016. doi:10.1016/J.TCS.2015.09.018.

9 Suparno Datta, Ayan Dutta, Sruti Gan Chaudhuri, and Krishnendu Mukhopadhyaya. Circle formation by asynchronous transparent fat robots. In *9th International Conference on Distributed Computing and Internet Technology, ICDCIT 2013*, pages 195–207. Springer, 2013. doi:10.1007/978-3-642-36071-8_15.

10 Xavier Défago and Akihiko Konagaya. Circle formation for oblivious anonymous mobile robots with no common sense of orientation. In *Workshop on Principles of Mobile Computing, POMC 2002*, pages 97–104. ACM, 2002. doi:10.1145/584490.584509.

11 Mattia D'Emidio, Daniele Frigioni, and Alfredo Navarra. Characterizing the computational power of anonymous mobile robots. In *36th International Conference on Distributed Computing Systems, ICDCS 2016*, pages 293–302. IEEE Computer Society, 2016. doi:10.1109/ICDCS.2016.58.

12 Yoann Dieudonné and Franck Petit. Circle formation of weak robots and lyndon words. *Inf. Process. Lett.*, 101(4):156–162, 2007. doi:10.1016/J.IPL.2006.09.008.

13 Yoann Dieudonné and Franck Petit. Swing words to make circle formation quiescent. In *14th International Colloquium on Structural Information and Communication Complexity, SIROCCO 2007*, pages 166–179. Springer, 2007. doi:10.1007/978-3-540-72951-8_14.

14 Yoann Dieudonné and Franck Petit. Squaring the circle with weak mobile robots. In *19th International Symposium on Algorithms and Computation, ISAAC 2008*, pages 354–365. Springer, 2008. doi:10.1007/978-3-540-92182-0_33.

15 Caterina Feletti, Carlo Mereghetti, and Beatrice Palano. Uniform circle formation for swarms of opaque robots with lights. In *20th International Symposium on Stabilization, Safety, and Security of Distributed Systems, SSS 2018*, pages 317–332. Springer, 2018. doi:10.1007/978-3-030-03232-6_21.

16 Caterina Feletti, Carlo Mereghetti, and Beatrice Palano. Uniform circle formation for fully, semi-, and asynchronous opaque robots with lights. *Applied Sciences*, 13(13), 2023. doi:10.3390/app13137991.

17 Caterina Feletti, Carlo Mereghetti, Beatrice Palano, and Priscilla Raucci. Uniform circle formation for fully semi-, and asynchronous opaque robots with lights. In *23rd Italian Conference on Theoretical Computer Science, ICTCS 2022*, pages 207–221. CEUR-WS.org, 2022. URL: https://ceur-ws.org/Vol-3284/8511.pdf.

18 Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. *Distributed Computing by Oblivious Mobile Robots*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2012. doi:10.2200/S00440ED1V01Y201208DCT010.

19 Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro, editors. *Distributed Computing by Mobile Entities, Current Research in Moving and Computing*, volume 11340 of *Lecture Notes in Computer Science*. Springer, 2019. doi:10.1007/978-3-030-11072-7.

20 Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Giovanni Viglietta. Distributed computing by mobile robots: uniform circle formation. *Distributed Comput.*, 30(6):413–457, 2017. doi:10.1007/S00446-016-0291-X.

21 László Hegedüs and Benedek Nagy. Representations of circular words. In *14th International Conference on Automata and Formal Languages, AFL 2014*, pages 261–270, 2014. doi:10.4204/EPTCS.151.18.

22 Manash Kumar Kundu, Pritam Goswami, Satakshi Ghosh, and Buddhadeb Sau. Arbitrary pattern formation by opaque fat robots on infinite grid. *Int. J. Parallel Emergent Distributed Syst.*, 37(5):542–570, 2022. doi:10.1080/17445760.2022.2088750.

23 Giuseppe Antonio Di Luna, Paola Flocchini, Sruti Gan Chaudhuri, Nicola Santoro, and Giovanni Viglietta. Robots with lights: Overcoming obstructed visibility without colliding. In *16th International Symposium on Stabilization, Safety, and Security of Distributed Systems, SSS 2014*, pages 150–164. Springer, 2014. doi:10.1007/978-3-319-11764-5_11.

24 R.C. Lyndon and M.P. Schützenberger. The equation $a^M = b^N c^P$ in a free group. *Michigan Math. J.*, 9:289–298, 1962. doi:10.1307/mmj/1028998766.

25 Moumita Mondal and Sruti Gan Chaudhuri. Uniform circle formation by mobile robots. In *19th International Conference on Distributed Computing and Networking, ICDCN 2018*, pages 20:1–20:2. ACM, 2018. doi:10.1145/3170521.3170541.

26 Moumita Mondal and Sruti Gan Chaudhuri. Uniform circle formation by swarm robots under limited visibility. In *16th International Conference Distributed Computing and Internet Technology, ICDCIT 2020*, pages 420–428. Springer, 2020. doi:10.1007/978-3-030-36987-3_28.

27 Gokarna Sharma, Ramachandran Vaidyanathan, and Jerry L. Trahan. Constant-time complete visibility for asynchronous robots with lights. In *19th International Symposium on Stabilization, Safety, and Security of Distributed Systems, SSS 2017*, pages 265–281. Springer, 2017. doi:10.1007/978-3-319-69084-1_18.

28 Gokarna Sharma, Ramachandran Vaidyanathan, Jerry L. Trahan, Costas Busch, and Suresh Rai. Complete visibility for robots with lights in O(1) time. In *18th International Symposium on Stabilization, Safety, and Security of Distributed Systems, SSS 2016*, pages 327–345, 2016. doi:10.1007/978-3-319-49259-9_26.

29 Yossi Shiloach. Fast canonization of circular strings. *J. Algorithms*, 2(2):107–121, 1981. doi:10.1016/0196-6774(81)90013-4.

30 Kazuo Sugihara and Ichiro Suzuki. Distributed algorithms for formation of geometric patterns with many mobile robots. *J. Field Robotics*, 13(3):127–139, 1996. doi:10.1002/(SICI)1097-4563(199603)13:3\%3C127::AID-ROB1\%3E3.0.CO;2-U.

31 Ichiro Suzuki and Masafumi Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM J. Comput.*, 28(4):1347–1363, 1999. doi:10.1137/S009753979628292X.

32 Giovanni Viglietta. Uniform circle formation. In *Distributed Computing by Mobile Entities, Current Research in Moving and Computing*, pages 83–108. Springer, 2019. doi:10.1007/978-3-030-11072-7_5.

33 Masafumi Yamashita and Ichiro Suzuki. Characterizing geometric patterns formable by oblivious anonymous mobile robots. *Theor. Comput. Sci.*, 411(26-28):2433–2453, 2010. doi:10.1016/J.TCS.2010.01.037.

A Algorithm

Algorithm 1 Pseudo-code of the *Task Loop* on the performance area φ_ρ.

Input: A performance arc $\widehat{\rho}$ where robots lay on.

$m \leftarrow$ robots on $\widehat{\rho}$ which are not pivot or angle;
while $m \geq 4$ **do**
  TBLG setting:
    The m robots color themselves as *top*, *bottom*, *loner*, *guard*, and *guard_c* in case;
    Guards arrange themselves as in Figure 4a and Figure 4b;
  Loner pairing:
    The uppermost loners move to pair themselves (see Figure 4b);
  Δ **setting:**
    if guards do not see any *projection* robot in φ_ρ **then**
      Outer guards approach each other to fix the distance Δ;
    end if
  Pair approaching:
    Each top robot approaches its bottom robot to form a distance δ;
  Top robots departure:
    Each top robot travels towards line h, perpendicularly;
  Guard green light on:
    Guards color themselves as *guard_green* or *guard_c_green*;
  Top robots on l**:**
    Top robots on h recompute the SEC and their target position on it;
    Top robots move to the projection of their target position on line l, setting their color as *projection*;
  Gap fix:
    if the number of bottom robots is odd **then**
      A bottom robot colors as *projection* and travels to l on the missing projection;
    end if
  Guard green light off:
    Guards set their color as *guard_end* or *guard_c_end*;
end while
The remaining robots on $\widehat{\rho}$, which are not pivots, angles or guards, reach l;

Output: Performance area φ_ρ where all robots lay on l (*projection* robots) or on $\widehat{\rho}$ (pivots, angles, guards).

B Lemmas and Theorems

In this section, all the lemmas and theorems used in the current paper are stated and proved.

B.1 Collision-free trajectories

▶ **Theorem 2.** *Let us assume a circle configuration where all robots lay on the SEC. Let* $\widehat{a}$ *be an arc of the SEC, and let* $T = \{t_1, \ldots, t_h\}$ *be* h *distinct robot-free (target) points of* $\widehat{a}$*, no one lying on the middle point of* $\widehat{a}$*. Assume at least* h *robots lay on distinct points of* $\widehat{a} \setminus T$*. Then, there exists an unambiguous way to elect* h *robots on* $\widehat{a}$ *and make them reach the* h *targets without colliding, and always guaranteeing complete visibility on the whole configuration.*

Proof. Firstly, we need an unambiguous way to elect h robots which will reach the target points. For this purpose, we select the h robots which are closer to the target points. In case of two robots equidistant from the same target point, we can use the geometry of the configuration (the distance from the closest endpoint of $\widehat{a}$) to elect one of them.
Let $w = w_1 \dots w_{2h}$ be the boolean string that represents the ordered displacement of the elected robots and the target points on $\widehat{a}$ such that: $w_i = 0$ symbolizes a target point, whereas $w_i = 1$ symbolizes a robot (e.g. $w = 11100010$ in Figure 7, following the clockwise direction). Let us factorize w in the shortest factors such that each factor has the same number of 0 and 1. Indeed, this factorization is unique and each factor cannot be a palindrome. Let w' be a factor of w according to the above factorization. We indicate with 1_i (resp. 0_i) the i-th 1 (resp. 0) in such a factor, where $1 \leq i \leq |\frac{w'}{2}|$. So, robots in that factor move following the following sequential schema:

for $i = |\frac{w'}{2}| \dots 1$ **do**
  if no robot in $\widehat{a}$ is moving **then**
    robot symbolized by 1_i heads to the target point symbolized by 0_i
  end if
end for

This schema assures no moving robot creates collinearity with other robots in the swarm, thus guaranteeing complete visibility. This is easily provable by induction on $|\frac{w'}{2}|$. Moreover, if h is constant, this strategy accomplishes the task in $\mathcal{O}(1)$ epochs. ◀

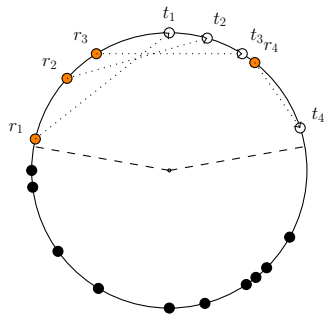


Figure 7 Trajectories of four robots assuring complete visibility to all the swarm. If a robot is moving, no other robot on the same arc moves.

B.2 Circular strings

In this section, we present some results about circular string properties which are used to study and analyze some geometric configurations of the robots on the SEC, especially in the rotational symmetry case. Specifically, given a configuration $\mathcal{C}$, we aim to figure out some properties on Γ_1, Γ_2 (i.e. on the presence of one or two eligible classes) thanks to the string properties of the sectors circular string $\sigma(\beta), \sigma(\beta^R)$ of $\mathcal{C}$.

▶ **Lemma 3** (Lyndon and Schützenberger, [24]). *Let x and y be two non-empty strings. If $xy = yx$, then there exist a string z and two integers n and m, such that $x = z^n$ and $y = z^m$.*

Proof. Let us proceed by induction on $|y| - |x|$, assuming w.l.o.g. that $|x| \leq |y|$.
If $|x| = |y|$ then the fact holds: in fact $x = y$, thus $z = x = y$ and $m = n = 1$. Otherwise, since $yx = xy$, y starts with a prefix that is equal to x. So $y = xu \Rightarrow\ xux = xxu \Rightarrow\ ux = xu$. By induction hypothesis, there exist z, m, n such that $u = z^m$, $x = z^n$, then $y = z^{n+m}$. ◀

▶ **Lemma 4.** *Let x and y be two non-empty strings. If xx^Ryy^R is a palindrome, then there exist a string z and two integers n and m, such that $x = z^n$ and $y = z^m$.*

Proof. Let us proceed by induction on $|y| - |x|$, assuming w.l.o.g. that $|x| \leq |y|$.
If $|x| = |y|$ then the fact holds: in fact $x = y$, thus $z = x = y$ and $m = n = 1$.
Let us assume that $|x| < |y|$. Hence, since xx^Ryy^R is a palindrome, y^R ends with a suffix which is equal to x^R. So

$$y^R = u^Rx^R \Rightarrow xx^Rxuu^Rx^R \text{ is a palindrome}$$

By simplifying, we have that x^Rxuu^R is a palindrome as well. By induction hypothesis, there exist z, m, n such that $u = z^m$, $x = z^n$, then $y = z^{n+m}$. ◀

▶ **Lemma 5.** *Let $\sigma(w)$ be a minimal circular string, where $w = w_0 \dots w_{n-1}$. Then $\sigma_i(w) \neq \sigma_j(w)$ for each $i \neq j$.*

Proof. The proof follows directly from Lemma 3. In fact, let us assume that there exist two different indexes such that $\sigma_i(w) = \sigma_j(w)$. Let us call them $w' = \sigma_i(w)$ and $w'' = \sigma_j(w)$, and w.l.o.g. let us assume $i < j$. Let z be the factor $w_i \dots w_{j-1}$, let x be the factor $w_j \dots w_{n-1}$, and let y be the factor $w_0 \dots w_{i-1}$, so that $w' = zxy$, and $w'' = xyz$. By Lemma 3, we can conclude that $\sigma(w)$ contains a power string, contradicting the hypothesis. ◀

▶ **Lemma 6.** *Given the two sector circular strings $\sigma(\beta), \sigma(\beta^R)$ of a configuration $\mathcal{C}$, there exist at most two classes of robots that read the same string γ. In particular, in case of two classes P_a and P_b, they reads γ in opposite directions (clockwise or counter-clockwise).*

Proof. Let us proceed by contradiction and let us suppose there is a third class P_c which reads the same string γ as P_a in the same direction. Let $a \in P_a$, $b \in P_b$, and $c \in P_c$ three robots belonging to the same sector. Let $w' = xyz$ be the string read by a and let $w'' = yzx$ be the string read by c (x is the string between a and c). Since $w' = w''$, then we can conclude that $xyz = yzx$ (for sake of simplicity, $xs = sx$). By Lemma 3, this means that w' and w'' are the power of the same factor v, which is in contrast with the initial hypothesis (a sector circular string must be minimal). ◀

▶ **Theorem 7.** *Let $\sigma(\beta), \sigma(\beta^R)$ be the two sector circular strings of a configuration $\mathcal{C}$. If $|\beta| = q$ is odd, then Γ_1 contains at least one string. So, there exists always a unique eligible class of symmetry.*

Proof. The proof follows by the fact that *(i)* the same string in $\sigma(\beta) \cup \sigma(\beta^R)$ can be read by at most 2 classes (by Lemma 6) and *(ii)*, in the worst case, $q - 1$ (even) classes read the same string two by two. So, at least one class reads a unique string. ◀

▶ **Lemma 8.** *Let $\sigma(\beta), \sigma(\beta^R)$ be the two sector circular strings of a configuration $\mathcal{C}$, where $|\beta| = q$ is even. If $\sigma(\beta)$ contains a mirrored string, then it contains exactly two different mirrored strings, whose rotation distance is $\frac{q}{2}$.*

Proof. By hypothesis, $\sigma(\beta)$ is minimal (by definition of sector circular string) and contains a string $w' = xx^R$ (and its related x^Rx). Obviously, $x^Rx = \sigma_{\frac{q}{2}}(xx^R)$. Note that $xx^R \neq x^Rx$. In fact, if $xx^R = x^Rx$, for Lemma 3 it follows that xx^R is a power string, contradicting our hypothesis ($\sigma(\beta)$ is minimal). Let us suppose that $\sigma(\beta)$ contains another pair of palindromes, say $w'' = yy^R$ (and y^Ry), at a rotation distance $k < q/2$ from the first pair. Let us assume that $x = zs$ where $|z| = k$. Thus we have that $w' = zss^Rz^R$ and $w'' = ss^Rz^Rz$. By Lemma 4, we can conclude that $w'' = v^n$ (for some not trivial v and n), again contradicting our hypothesis about $\sigma(\beta)$. ◀

▶ **Theorem 9.** *Let $\sigma(\beta), \sigma(\beta^R)$ be the two sector circular strings of a configuration $\mathcal{C}$, where $|\beta| = q$ is even. If $\sigma(\beta)$ contains a mirrored string, then Γ_1 contains at least one string. So, there exists always a unique eligible class of symmetry.*

Proof. According to Lemma 8, there exist just 2 different palindromes in $\sigma(\beta)$, say xx^R and x^Rx, which are read by two different classes of symmetry, say P_i and P_j. In particular, P_i reads xx^R in both directions, and P_j reads x^Rx in both directions. The fact that a third class P_k reads xx^R is excluded by Lemma 4. In fact, if the same palindrome string is read from two different classes, whose distance is less than $\frac{q}{2}$, then xx^R is in the form zz^Ryy^R, contradicting the hypothesis. Since $x \neq x^R$ (otherwise $\sigma(\beta)$ would not be minimal), P_i (resp. P_j) is the unique class reading xx^R (resp. x^Rx). So, Γ_1 contains at least xx^R and x^Rx. ◀

▶ **Theorem 10.** *Let $\sigma(\beta), \sigma(\beta^R)$ be the two sector circular strings of a configuration $\mathcal{C}$, where P_i and P_j are the two classes of placeholders. For each pair of adjacent placeholders $r_i \in P_i$ and $r_j \in P_j$ (no other placeholder lays on the arc $\overset{\frown}{p_ip_j}$), an axis of symmetry passes between them.*

Proof. By hypothesis, r_i and r_j read the same angle string in opposite directions. Let us suppose that the string distance between r_i and r_j is $k < |\beta|$. Thus, r_i reads $w' = x_1 \dots x_k y$ and r_j reads $w'' = x_k \dots x_1 y^R$. Since $w' = w''$, we can conclude that $x_k \dots x_1 = x_1 \dots x_k$ and $y^R = y$ (i.e. $x_1 \dots x_k$ and y are palindrome). Thus, between the arc delimited by r_i and r_j (in both directions), there is an axis of symmetry. ◀

▶ **Theorem 11.** *Let $\sigma(\beta), \sigma(\beta^R)$ be the two sector circular strings of a configuration $\mathcal{C}$, where P_i and P_j are the two eligible classes of placeholders. Then the axis of symmetry passing through two adjacent placeholders r_i, r_j does not pass across any robot on the arc $\overset{\frown}{r_ir_j}$.*

Proof. Since r_i and r_j are placeholders, they read the same angle-string w. Let us assume by contradiction that the axis passes through a robot and let xx^R be the prefix of w which is common in r_i and r_j (which is a mirrored string, since it can be read in opposite directions and its length is even). So, r_i reads the angle-string xx^Ry, whereas r_j reads xx^Ry^R. Since $|\beta|$ and $|xx^R|$ are both even (otherwise we would have just one eligible class, by Theorem 7), then $|y|$ is even too. Since $xx^Ry = xx^Ry^R$, we have that y is a mirrored string too. Let vv^R a factorization for y. Thus, $\sigma(\beta)$ turns out to contain a rotation of w which is a mirrored string (x^Rvv^Rx). By Theorem 9, since $\sigma(\beta)$ contains a mirrored string, then there is a unique eligible class of symmetry. The assumption that the axis passes through a robot must be wrong with these hypotheses. ◀