Forming Sequences of Patterns With Luminous Robots

SHANTANU DAS, PAOLA FLOCCHINI, GIUSEPPE PRENCIPE, AND NICOLA SANTORO

1 LIS, Aix-Marseille University, Parc Scientifique et Technologique de Luminy, 13009 Marseille, France
2 SEECS, University of Ottawa, Ottawa, ON K1N 6N5, Canada
3 Dipartimento di Informatica, Università di Pisa, 56127 Pisa, Italy
4 School of Computer Science, Carleton University, Ottawa, ON K1S 5B6, Canada

Corresponding author: Giuseppe Prencipe (giuseppe.prencipe@unipi.it)

This work was supported in part by the Progetto, Università di Pisa, Pisa, Italy, under Grant PRA_2018_43, and in part by The Natural Sciences and Engineering Research Council of Canada through the Discovery Grants Program.

ABSTRACT The extensive studies on computing by a team of identical mobile robots operating in the plane in Look-Compute-Move cycles have been carried out mainly in the traditional OBLOT model, where the robots are silent (have no communication capabilities) and oblivious (in a cycle, they have no memory of previous cycles). To partially overcome the limits of obliviousness and silence while maintaining some of their advantages, the stronger model of luminous robots, LUMI, has been introduced where the robots, otherwise oblivious and silent, carry a visible light that can take a number of different colors; a color can be seen by observing robots, and persists from a cycle to the next. In the study of the computational impact of lights, an immediate concern has been to understand and determine the additional computational strength of LUMI over OBLOT. Within this line of investigation, we examine the problem of forming a sequence of geometric patterns, PATTERNSEQUENCEFORMATION. A complete characterization of the sequences of patterns formable from a given starting configuration has been determined in the OBLOT model. In this paper, we study the formation of sequences of patterns in the LUMI model and provide a complete characterization. The characterization is constructive: our universal protocol forms all formable sequences, and it does so asynchronously and without rigidity. This characterization explicitly and clearly identifies the computational strength of LUMI over OBLOT with respect to the PATTERNSEQUENCEFORMATION problem.

INDEX TERMS Autonomous mobile robots, distributed computing, oblivious, pattern formation, sequence of patterns, visible lights.

I. INTRODUCTION

The control and coordination of systems of autonomous mobile robots operating in continuous spaces has been the object of intensive investigations in a variety of fields, including robotics, control, AI, and distributed computing.

A. FRAMEWORK

Within distributed computing, the robots are seen as mobile computational entities operating in the Euclidean space in Look-Compute-Move cycles. During a cycle, a robot obtains a snapshot of the positions of the other robots, expressed in its own local coordinate system (Look); using the snapshot as an input, it executes an algorithm, the same for all robots, to determine a destination (Compute); and it moves towards the computed destination (Move); after a cycle, a robot may be inactive for some time. The main focus has been on determining the minimal capabilities the robots need to have to be able to solve a problem, and the extensive literature has almost exclusively focused on simple computational entities operating under strong adversarial conditions. The robots are anonymous (without ids or distinguishable features), autonomous (without central or external control), homogeneous (they execute the same program), silent (without explicit means of communication), and oblivious (no recollection of computations and observations done in previous cycles).

In this standard model of robots, typically called OBLOT, depending on the activation schedule and timing assumptions, three main sub-models have been studied in the literature: the asynchronous model (ASYNC), where no assumptions are made on synchronization among the robots’ cycles.
nor their duration; and the fully synchronous (\(F_{\text{SYNC}}\)) and semi-synchronous (\(S_{\text{SYNC}}\)) models where the robots operate in synchronous atomic rounds, the only difference being whether all robots are activated in every round or, subject to some fairness condition, a possibly different subset is activated in each round. Extensive research has been carried out to determine, in all three sub-models, the computational boundaries as well as the impact that factors such as presence/absence of common coordinate system, movement rigidity (i.e., being able to always reach the destination), etc., have on computability and complexity (e.g., see [3], [6], [14]–[16], [18], [28], [29], [31], and the recent books and chapters therein [12], [13]). Although this research is highly theoretical, its influence is sometimes felt in more practical settings (e.g., see [5], [19], [20], [22], [27] in the recent literature).

Among all the restricting conditions, being oblivious and silent are the most limiting constraints for the robots, forcing them to rely only on movement and observation of the environment to coordinate and decide their next step. On the other hand, both obliviousness and silence are very desirable characteristics, as the former provides a form of self-stabilization, while the latter allows for applications in environments where communication is impossible or dangerous.

Recently, to partially overcome the limits of obliviousness and silence while maintaining some of their advantages, the stronger model of luminous robots, LUMI, has been introduced where robots, otherwise oblivious and silent, carry a visible light that can take (a bounded number of) different colors, can be seen by observing robots, and persists from cycle to cycle [7], [8].

The availability of visible lights clearly changes the computational power of the robots, which have now the means for a bounded amount of persistent memory (by coding information into the colors), and limited communication (by using the change of colors to transmit a bounded amount of information to the other robots). The research is now ongoing to determine the impact that the availability of visible lights has on the computability power of the robots, and how lights can be used to overcome additional limitations (e.g., obstructed visibility). The first crucial result is the existence of an algorithmic transformation that allows any semi-synchronous protocol for luminous robots with \(k\) colors to be executed asynchronously with \(O(k)\) colors [8]. In other words, when solving a problem for asynchronous luminous robots, it suffices to solve it for semi-synchronous ones; the price to pay is a (small) constant multiplicative factor in the number of colors. Furthermore, since robots in OBLOT are luminous robots with \(k = 1\) color, this result implies that asynchronous luminous robots are at least as powerful as semi-synchronous traditional robots. Since their introduction, a large amount of work has been done on luminous robots ([11], [2], [4], [8], [10], [17], [21], [23]–[26], [30]; see [11] for a recent review). In this paper we continue the investigation on the computational impact of lights, and examine the problem of forming a sequence of geometric patterns.

### B. FORMING A SEQUENCE OF PATTERNS

The research on the computational aspects of autonomous mobile robots has in large part focused on the fundamental class of Geometric Pattern Formation problems. A geometric pattern (or simply pattern) \(P\) is a set of points in the plane; the robots form the pattern \(P\) at time \(t\) if the configuration of the robots (i.e., the set of their positions) at time \(t\) is similar to \(P\) (i.e., coincident with \(P\) up to scaling, rotation, translation, and reflection); a pattern \(P\) is formable if there exists an algorithm that allows the robots to form \(P\) within finite time and no longer move.

Determining which patterns are formable under what conditions has been the subject of extensive studies in a variety of settings (e.g., see [16], [18], [28], [29], [31]). A crucial role is played by the notion of symmetricity of a set of points, which informally measures the amount of symmetries of the set. Indeed, for a pattern \(P\) to be formable, its symmetricity must divide the one of the starting configuration (i.e. initial placement of the robots in the plane). The obtained results indicate that, in most settings, the robots can form single arbitrary geometric patterns in spite of their silence and obliviousness. In other words, apart from the inevitable arbitrary limitations due to symmetry, obliviousness and silence are not limiting factors to form a single pattern.

The problem of forming more than one pattern in a specified order is clearly much more difficult due to the obliviousness of the robots. In fact, to be able to correctly transition from a pattern to the next in the prescribed order, the robots must create in the environment the mechanism to “remember” and “communicate”.

Let \(S = (S_0, S_1, \ldots, S_{m-1})\) be an ordered sequence of patterns. Forming sequence \(S\) requires the robots to form the infinite periodic sequence \(S^\infty = (S_0, S_1, \ldots, S_{m-1})^\infty\), called coreography; that is, after forming pattern \(S_i\), the robots must form pattern \(S_{i+[i+1] \mod m}\), where the first formed pattern can be any pattern in the sequence [9]. The problem of determining which coreographies can be formed from a given initial configuration, the \(\text{PATTERNSEQUENCEFORMATION}\) problem, has been studied in the \(\text{OBLOT}\) model in [9], where a complete characterization was given for the set of the sequences formable from the initial configuration \(\Gamma\) by semi-synchronous robots with rigid movements (i.e., when moving, a robot always reaches its destination), assuming chirality (i.e., clockwise orientation). Specifically, the authors proved that a sequence \(S\) of patterns formable from \(\Gamma\) can be formed by rigid robots in \(\text{SYNC}\) only if (R1) the number of points in each pattern is the same and (R2) no pattern appears more than once in the sequence.

Since traditional robots without lights are the particular case of luminous robots with one color, the results proven in [8] imply that all sequences formable by robots in \(\text{SYNC}\) with rigid movements and chirality can be done by asynchronous luminous robots under the same conditions.

This naturally poses several questions on the sequence of patterns formable by asynchronous luminous robots vs those formable by traditional robots. Specifically, are asynchronous...
luminous robots more powerful than traditional robots in $\textit{SYNC}$? If so, what other sequence of patterns can be formed? How can they be formed without rigidity? In this paper, we provide a complete answer to these questions.

C. CONTRIBUTIONS

We provide a complete characterization of the sequences of patterns formable by (non-rigid) luminous robots with chirality, starting from an input configuration $\Gamma$.

The characterization shows that, in contrast with the case without lights, the sequences $\mathcal{F}(\Gamma)$ of patterns that can be formed do not suffer from restrictions (R1) and (R2); in fact, formable sequences can contain repeated patterns (repetitions), and the number of points in the patterns can be different (contractions). The number of colors needed by the robots, and used by the algorithm, depends necessarily on the number of repetitions and contractions in the sequence of patterns; i.e., it is bounded but not by a constant. Furthermore, unlike the case without lights, the set $\mathcal{F}(\Gamma)$ of sequences formable by luminous robots is the same regardless of (a)synchrony and regardless of rigidity.

The characterization is constructive. We first identify the sequences that cannot be formed even if the robots are fully synchronous and rigid; we then provide a universal protocol that, for any given arbitrary initial configuration $\Gamma$ and any given ordered sequence $S = \langle S_0, \ldots, S_{m-1} \rangle$ of patterns formable from $\Gamma$, allows the robots to form $S$ starting from $\Gamma$; furthermore, this is done asynchronously, without requiring rigidity, and always starting from $S_0$.

This characterization explicitly and clearly identifies the computational strength of $\textit{LUMI}$ over $\textit{OBLOT}$ with respect to the $\textit{PATTERNSEQUENCEFORMATION}$ problem.

The paper is organized as follows: in Section II, we introduce some notation and definitions; in Section III, we establish some basic limitations of the robots as well as a lower bound on the number of colors to be employed; in Section IV, we describe an Algorithm to solve the problem and analyze its correctness; in Section V, we establish bounds on the number of colors necessary to solve the problem; finally, in Section VI we draw some conclusions and discuss open research directions.

II. MODEL AND DEFINITIONS

A. THE MODEL

The system is composed of a team $\mathcal{R}$ of mobile entities, called robots, each modelled as a computational unit provided with its own local memory and capable of performing local computations.

The robots are placed in a spatial universe, here assumed to be $\mathbb{R}^2$, and they are viewed as points in $\mathbb{R}^2$. We assume that the robots always start from distinct points in the plane, but during the course of the algorithm multiple robots may occupy the same point in $\mathbb{R}^2$. Each robot has its own local coordinate system that might not be consistent with that of the other robots. A robot is endowed with sensorial capabilities and it observes the world by activating its sensors, which return a snapshot of the positions of the other robots in its local coordinate system.

The robots are identical and anonymous: they are indistinguishable by their appearance, execute the same protocol, and do not have distinct identities. The robots are autonomous, without a central control. The robots are silent, meaning that a robot cannot directly communicate with another robot (e.g., using wireless communication). The robots can however communicate information indirectly, e.g. using specific movements or using lights, as explained later.

Each robot is endowed with motorial capabilities, and can freely move in the plane.

At any point in time, a robot is either active or inactive. When active, a robot executes a Look-Compute-Move (LCM) cycle. In Look, a robot obtains an instantaneous snapshot of the positions of all robots (i.e., the set of their coordinates) expressed in its own coordinate system, which might be different in different cycles. In Compute, the robot executes its algorithm, using the snapshot as input; the result of the computation is a destination point. In Move, the robots move toward the computed destination; if the destination is the current location, the robot stays still (performs a null movement). When inactive, a robot is idle. All robots are initially inactive. The amount of time to complete a cycle is assumed to be finite.

The algorithm to be executed, with all the functions it uses, as well as the target to be achieved (in our case, the sequence of patterns) are the only information persistently available to a robot. Otherwise, the robots are oblivious: they start each cycle without any information of past observations and computations. The local coordinate systems of the robots may not be consistent with each other, and might differ in different cycles; however, a robot always sees itself at the origin of the system and, as in [9], [18], it shares the same chirality (e.g., a clockwise orientation of the plane) with all the other robots.

In a move, a robot might always reach its destination, in which case the move is said to be rigid; alternatively, the move might stop before reaching its destination, e.g., because of limits to its motion energy. In the non-rigid case, the distance traveled in a move is neither infinite nor infinitesimally small. More precisely, there exists an (arbitrarily small) constant $\delta > 0$ such that, if the destination point is closer than $\delta$, the robot will reach it; otherwise, it will move towards it a distance of at least $\delta$; the quantity $\delta$ might not be known to the robots. In this paper, the proposed algorithms are able to operate in this more difficult general setting of non-rigid movements.

With respect to the activation schedule of the robots and their Look-Compute-Move cycles, the most common models are the fully-synchronous, the semi-synchronous, and the asynchronous. In the fully-synchronous ($\textit{FSYNC}$) model, the activations of all robots can be logically divided into global rounds; in each round, the robots are all activated, obtain the same snapshot, compute and perform their move. Note that this is computationally equivalent to a fully synchronized
system in which all robots are activated simultaneously and all operations are instantaneous. The semi-synchronous (SYNC) model is like the fully-synchronous model where however not all robots are necessarily activated in each round. Based on the fairness of the activation scheduler, sub-models can be obviously defined; it is assumed that every robot is activated infinitely often. On the opposite side of the spectrum, in the asynchronous (ASYNC) model, the robots do not have a common notion of time; they are activated independently from each other, and the duration of each \( \text{Compute, Move} \) and inactivity is finite but unpredictable.

As a consequence, robots perceived in a snapshot might be performing any operation and, in particular, they might be moving; moreover, computations might be made based on obsolete observations. In this paper, the proposed algorithm is able to operate in this more difficult but more general model.

A luminous robot \( r \) is a robot that, in addition to its capabilities, is endowed with a persistent and externally visible state variable \( \text{Light}[r] \), called light, whose values are from a finite set \( C \) of states called colors (including the color that represents the state when the light is off). Note that if \( |C| = 1 \), then the light is always off; thus, this case corresponds to the traditional setting of robots without lights. In this paper, we assume \( |C| > 1 \) for (truly) luminous robots. The value of \( \text{Light}[r] \) (i.e., its color) can be changed in each cycle by \( r \) at the end of its \( \text{Compute} \) operation. A light is externally visible in the sense that its color at time \( t \) is visible to all robots that perform a \( \text{Look} \) operation at that time; we assume weak chromatic multiplicity detection: the robots can distinguish the colors of robots co-located at a point, but not their number.\(^1\) A light is persistent from one computational cycle to the next; while robot \( r \) might be oblivious, forgetting all other information from previous cycles, the color is not automatically reset at the end of a cycle.

### B. CONFIGURATIONS, SYMMETRIES, AND PATTERNS

Let \( V \) be a set of points in the Euclidean plane, \( \text{size}(V) \) be the cardinality of \( V \), and \( \text{SEC}(V) \) be the smallest circle enclosing all the points (refer to Figure 2 for a summary of the used notation). The set \( V \) can be decomposed into a set of concentric circles centred in the centre of \( \text{SEC}(V) \), each containing at least a point of \( V \); let \( \sigma(V) \) be their number, and let us denote them as \( \text{Cir}_1, \ldots, \text{Cir}_{\sigma(V)} \), with \( \text{Cir}_1 \) being the smallest one; in the following, when clear from the context, the superscript will be omitted. As well known, there exists a \( q \geq 1 \) divisor of \( \text{size}(V) \), such that every \( \text{Cir}_i \) can be decomposed into a set of \( q \)-gons; if there is a point at the centre of \( \text{SEC}(V) \), this point is, by definition, a (degenerated) 1-gon. The largest \( q \) for which this holds is called \( \text{symmetry class} \) of \( V \) and denoted by \( q(V) \) (refer also to examples in Figure 1). The set of points in each \( q(V) \)-gon is called a \( \text{symmetry class} \), or simply class; we denote by \( \sigma(V) \) the number of classes in \( V \). We extend in a natural way the definition of symmetry also to multisets; in a multiset \( V \), a point of multiplicity \( k \) at the centre of \( \text{SEC}(V) \) is, by definition, a (degenerated) \( k \)-gon.

For two \( k \)-gons \( A \) and \( B \) placed on two concentric circles \( C_A \) and \( C_B \), we denote by \( \theta(C_A, C_B) \) the clockwise rotational angle between \( A \) and \( B \).

Given a set of distinct colors \( C = \{c_1, \ldots, c_k\} \) and a set of points \( V \), we define a coloring of \( V \) with \( C \) any function \( \lambda : V \rightarrow C \). We say that \( \lambda \) is proper when \( \lambda(x) = \lambda(y) \) iff \( x \) and \( y \) belong to the same class of symmetry in \( V \). We denote by \( q_c(V, \lambda) \) the \( \text{chromatic symmetry} \) of \( V \) when colored by \( \lambda \); in the following, when \( \lambda \) is understood, we use just \( q_c(V) \). We extend in a natural way the definition of chromatic symmetry also to multi-sets.

In the following, we will describe the global positions of the robots using a fixed coordinate system (not available to the robots); let \( r_i(t) \) denote the position of \( r_i \) at time \( t \) in this system.

The configuration of the robots on the plane at time \( t \) is the multi-set \( \Gamma(t) = \{(r_i(t), \lambda(r_i(t))): r \in R\} \) where \( \lambda(r_i(t)) \) is the color of robot \( r \) at time \( t \); the set of distinct points occupied by the robots in \( \Gamma(t) \) is denoted by \( L(\Gamma(t)) \). A configuration \( \Gamma \) is said to be central if there is a point at the center of \( \text{SEC}(\Gamma) \), not-central otherwise; we denote by \( \Gamma^\circ \) the configuration where the robots at the center of \( \text{SEC}(\Gamma) \), if any, are removed. In the initial configuration \( \Gamma \), we assume that all robots occupy distinct positions and that all robots have the same color, called \( \text{Off} \).

---

\(^1\) Note that this requirement corresponds to absence of multiplicity detection if the robots are not luminous.
A pattern $P$ is a set of distinct points. A pattern $P$ is said to be central if there is a point at the center of $SEC(P)$, non-central otherwise. A pattern $P_i$ is said to be isomorphic to a pattern $P_j$, denoted $P_i \equiv P_j$, if $P_j$ can be obtained by a combination of translation, rotation and scaling of pattern $P_i$. Two patterns that are not isomorphic to each other are said to be distinct. We will denote the size of a pattern $P$ by $\text{size}(P)$, and denote by $P^-$ the pattern where the point at the center of $SEC(P)$, if any, is removed.

We say that the robots have formed the pattern $P$ at time $t$ if $L(\Gamma(t)) \equiv P$.

Let $S = \langle S_0, \ldots, S_{m-1} \rangle$ be an ordered sequence of patterns with $S_i \neq S_{i+1}$. The maximum number of points in the patterns is called size of the choreography and it is denoted by $\text{size}(S) = \max_i(\text{size}(S_i))$. The maximum number of symmetry classes in $S$ is defined as $\alpha(S) = \max_i(\alpha(S_i))$. Without any loss of generality, let $\alpha(S_0) = \alpha(S)$. Given a pattern $P \in S$, we denote by $\mu(P)$ the number of occurrences of $P$ in $S$, i.e. $\forall P \in S, \mu(P) = \{|S_i \in S : S_i \equiv P\}|$. Let $\mu(S) = \max_i(\mu(S_i))$. We say that the sequence $S$ has repetitions, if $\mu(S) > 1$, and contractions, if there is $P$ in $S$ such that $\mu(P) < \alpha(S)$ (refer to Figure 3). Also, if $\alpha(S_i) < \alpha(S_{i+1})$, we will say that $S_i$ contracts.

A set of luminous robots executing an algorithm $A$ starting from a configuration $\Gamma(0)$ is said to form $S$ if in every possible execution of $A$ from configuration $\Gamma(0)$, there exist increasing time instances $t'_0, \ldots, t'_{m-1}$, with $t'_0 \leq t'_i$ such that $L(\Gamma(t'_i)) \equiv S_i$.

Given the sequence $S = \langle S_0, \ldots, S_{m-1} \rangle$, we call choreography the infinite periodic sequence $S^\infty = \langle S_0, S_2, \ldots, S_{m-1} \rangle^{\infty}$. We say that a set of robots executing an algorithm $A$, starting from configuration $\Gamma(0)$ performs the choreography $S^\infty = \langle S_0, S_2, \ldots, S_{m-1} \rangle^{\infty}$ if they repeatedly form $S$.

In the following, for simplicity, with an abuse of notation, we will refer to a choreography $S^\infty$ simply as $S$.

### III. CHARACTERIZATION AND LIMITATIONS

#### A. BASIC LIMITATIONS

We first establish an obvious basic limitation of pattern-formation by luminous robots.

**Lemma 1**: Consider a set of $n$ luminous robots in configuration $\Gamma$ with chromatic symmetry $q_c(\Gamma)$. The robots cannot form a configuration $\Gamma'$ where $q_c(\Gamma')/q_c(\Gamma)$ is not an integer, even in $F_{SYNC}$, with rigid movements, and regardless of the number of available colors.

**Proof**: By contradiction, let $A$ be an algorithm that, starting from configuration $\Gamma$, correctly allows the robots to reach configuration $\Gamma'$, where $q_c(\Gamma')/q_c(\Gamma)$ is not an integer, and it does so in all possible executions, regardless of the local coordinate systems of the robots. For the $q_c(\Gamma)$ robots of a class not at the center of $SEC(L(\Gamma'))$ chirality induces a natural cyclical order; for the $q_c(\Gamma)$ robots of a class at the center of $SEC(L(\Gamma'))$, consider an arbitrary cyclical order. Consider now a fully synchronous execution of $A$ where the coordinate system of each robot is a rotation by $\frac{2\pi}{q_c(\Gamma)}$ of that of the robot before it in the cyclic order of its class. In every step of such an execution, all the robots in the same chromatic class of $\Gamma$ will obtain the same chromatic snapshot, perform the same computation, choose the same new color, and perform a move which will not change their symmetry; hence, they will remain in the same chromatic class. It is possible that two (or more) distinct classes choose the same color and their movements position them into a symmetric configuration; however, this will cause the size of the chromatic classes to either stay unchanged or to increase by an integer factor. This means that, in each step $i$ of the execution, we have $q_c(\Gamma(t - 1)) \geq q_c(\Gamma(t))$ and $q_c(\Gamma(t)) = q_c(\Gamma(\Gamma(0)))$ is an integer. This will hold also when the robots finally reach $\Gamma'$ at step $t'$. That is, $\Gamma(t') = \Gamma'$ but $q_c(\Gamma(t'))$ is an integer; a contradiction.

The basic limit to performing a choreography immediately follows.

**Theorem 2**: A set of $n$ luminous robots starting from initial configuration $\Gamma$ where all robots are in distinct positions and have the same color, cannot perform a choreography $S$, if any of the following holds:

1. $\text{size}(S_i) > n$, for some $S_i \in S$.
2. $q(S_i^-)$ is not a multiple of $q(\Gamma)$, for some $S_i \in S$.
3. $S$ contains both central and non-central patterns, $q(\Gamma) > 1$, and $\alpha(\Gamma) = \alpha(S)$.

This holds even in $F_{SYNC}$, with rigid movements, and regardless of the number of available colors.

**Proof**: We consider each condition separately.

(1) This condition trivially holds since $\text{size}(\Gamma) = n$ robots cannot occupy more than $n$ distinct positions at any one time.

(2) This condition follows from Lemma 1 observing that, since in the initial configuration $\Gamma$ all robots have the same color, $q_c(\Gamma) = q(\Gamma)$.

(3) Let $S$ contains both central and non-central patterns, $q(\Gamma) > 1$, and $\alpha(\Gamma) = \alpha(S)$. By contradiction, let $S$ be formable; that is, there is an algorithm $A$ that starting from $\Gamma$ always correctly performs choreography $S$ in all possible executions.

By condition (2) of this theorem, for all $S_i \in S$, $q(S_i^-)$ must be a multiple of $q(\Gamma)$; thus $q(\Gamma) = \max_i(q(S_i^-)) \geq q(\Gamma)$. Since $\alpha(\Gamma) = \alpha(S)$ by assumption, it follows that $\text{size}(S) = \alpha(S)q(\Gamma) \geq \alpha(\Gamma)q(\Gamma) = \text{size}(\Gamma)$. Since, by condition (1) of this theorem, to be formable $S$ must have $\text{size}(\Gamma) \geq \text{size}(S)$,
it follows that \( \text{size}(\Gamma) = \text{size}(\mathcal{S}) \); that is, there exists a \( S_{\text{max}} \in \mathcal{S} \) such that \( \text{size}(S_{\text{max}}) = \text{size}(\Gamma) \).

Consider now a fully synchronous execution of \( \mathcal{A} \) where initially the coordinate system of each robot is a rotation by \( 2\pi/q_c(\Gamma) \) of that of the robot before it in the cyclic order of its class. By assumption, \( \mathcal{S} \) contains central and non-central patterns. Let \( S_k \) be the first central pattern formed in this execution (i.e., all \( S_i \) with \( 0 \leq i < k \) are non-central), and let \( \Gamma_0 \) be the configuration when the robots form \( S_k \) for the first time. Since in \( \overline{\Gamma} \) the robots are in distinct positions and \( q(\overline{\Gamma}) > 1 \), to form \( \Gamma_0 \) at least one robot must move to the center. In this execution, when a robot moves to the center, all the robots in the same class will necessarily do the same ("class collapse"). It follows that \( \text{size}(S_k) < \text{size}(\overline{\Gamma}) \).

Continue now the synchronous execution of \( \mathcal{A} \) with all robots belonging to the same (chromatic) class located in the center of \( \Gamma_0 \) having the same coordinate system; these robots will, from this moment on, move as one robot, always taking the same decision and moving to the same point. In other words, the number of distinct points occupied by robots in any subsequent configuration is less than \( \text{size}(\overline{\Gamma}) \). Recall that there exists a pattern \( S_{\text{max}} \in \mathcal{S} \) with \( \text{size}(S_{\text{max}}) = \text{size}(\overline{\Gamma}) \) distinct points; this means that the pattern \( S_{\text{max}} \) cannot ever be formed again, contradicting the correctness of \( \mathcal{A} \).

A choreography is feasible from initial configuration \( \overline{\Gamma} \) if none of the three forbidden conditions stated in the previous theorem holds. Let \( \mathcal{F}(\overline{\Gamma}) \) denotes the class of all feasible choreographies starting from \( \overline{\Gamma} \).

Notice that, as a consequence of Theorem 2, unless the initial configuration is asymmetric or the number of robots is greater than the number of points in the largest pattern, feasible choreographies can only be composed of exclusively non-central patterns, or of exclusively central patterns: in the former case, we will say that \( \mathcal{S} \) is non-central, and in the latter that \( \mathcal{S} \) is central.

For simplicity, in the following, we will use \( \text{size}(\Gamma(t)), q(\Gamma(t)), \) and \( \alpha(\Gamma(t)) \) to denote \( \text{size}(L(\Gamma(t))), q(L(\Gamma(t))), \) and \( \alpha(L(\Gamma(t))) \), respectively, and when no ambiguity arises we will omit the variable \( t \).

### IV. THE ALGORITHM

In this section we present an algorithm that, starting from an arbitrary initial configuration \( \overline{\Gamma} \), allows the robots to perform any choreography in \( \mathcal{F}(\overline{\Gamma}) \) even if the execution is asynchronous and the movements are not rigid. In the following, for simplicity, we assume \( \alpha(\overline{\Gamma}) = \alpha(\mathcal{S}) \); the case \( \alpha(\overline{\Gamma}) > \alpha(\mathcal{S}) \) is easily handled, as discussed in Section VI.

We will first consider the case when the initial configuration is symmetric (i.e., \( q(\overline{\Gamma}) > 1 \)); for ease of presentation, we will introduce in Sections from IV-A to IV-H a solution that assumes \( \mathcal{S} \) non-central; in Section IV-I we will describe how to extend such a solution to handle also the case when \( \mathcal{S} \) is central.

The case of asymmetric initial configuration (i.e., \( q(\overline{\Gamma}) = 1 \)), can be solved with a simpler protocol, and will be described in Section IV-J.

### A. SYMMETRIC INITIAL CONFIGURATION

Let the initial configuration \( \overline{\Gamma} \) be symmetric and \( \mathcal{S} \) be non-central; that is, \( q(\overline{\Gamma}) > 1 \), and all patterns \( S_i \in \mathcal{S} \) are non-central.

We start with an informal overview of the algorithm for this case (Algorithm 1).

#### Algorithm 1 PATTERNSEQUENCE()

1. **SETUP();** %Algorithm 2%
2. **PATTERNIDENTIFICATION();** %Algorithm 3%

   **If** \( \Gamma \) is ready-to-separate or in-separation **Then**
   1. **SEPARATION();** %Algorithm 5%
   **If** \( \Gamma \) is separated or in-rotation **Then**
   1. **ROTATION();** %Algorithm 7%
   **If** \( \Gamma \) is rotated or in-formation **Then**
   1. **FORMATION();** %Algorithm 8%

Whenever the robots have to form the next (initially, the first) pattern in \( \mathcal{S} \), from a global point of view, the algorithm makes the robots perform two main phases: a pre-processing phase and an actual formation phase. The pre-processing phase is articulated into a 2-step process:

- **Setup.** A class is elected as leader.
- **Pattern Identification.** The robots identify the pattern \( S_i \) of the sequence that has to be formed.

Initially all the robots are colored OFF. From this initial configuration, as a preliminary step, a class of robots is elected as leader in SETUP(): robots in this class assume a special color that will uniquely identify them from this moment on. At this point the robots start the actual formation process.

The first action undertaken by the robots is to ensure that the next pattern to be formed, say \( S_i \), is recognizable during the entire formation process. This is done in PATTERNIDENTIFICATION() by positioning the leader robots on a special circle, centered in the SEC of the current configuration, and having a special radius that uniquely identifies \( S_i \); hence, once all the leader robots reach that circle, their position enables the other robots to identify \( S_i \) as the next pattern to be formed. The algorithm ensures that, while the leaders move towards that circle, they can understand from the position and colors of the other robots, the radius they have to reach.

Next, the algorithm starts the actual formation phase of \( S_i \), that is articulated into a 3-step process:

- **Separation.** Each class of robots places itself on a distinct concentric circle centered in the center of SEC.
- **Rotation.** Each class rotates so that the relative angles of the pattern to be formed are respected.
- **Formation.** The robots move to their final destination, thus forming \( S_i \).

In particular, the algorithm now determines which of these three steps the current configuration of the robots \( \Gamma \) is in. This check is carried out by testing conditions...
ready-to-separate, in-separation, separated, in-rotation, rotated, and in-formation that will be detailed in Sections IV-E, IV-G, and IV-H: these tests are carried out by considering the positions and the colors of the robots. We will prove that, at each time, at most one of these conditions can be verified.

To start the actual formation of \( S_i \), each class of robots is first placed on a distinct concentric circle, thus separating the different classes, and then it is mapped onto a “target” class of \( S_i \) (\text{SEPARATION()}). The robots of every class then start rotating (one class at the time and never leaving their own circle) in order to be at the correct relative rotational angle with respect to their own target class (\text{ROTATION()}).

Once all classes, including the one containing the leader robots, are placed at the correct rotational angle, the robots start moving, one class at the time, towards their target positions, forming \( S_i \) (\text{FORMATION()}). As soon as the robots of a class reach their final destination, they assume a proper color, assigned by a coloring function defined in Section IV-C; the last class to be moved in this process is the leader class. Once the process is completed, it starts again to form the next pattern \( S_{((i+1)\mod m)} \).

The difficulties obviously derive from the fact that, in this endless process, each robot performs all the required operations in complete asynchrony, without communication or memory (other than the colors), and its movements can be interrupted (they are not rigid).

We now proceed with the detailed description and discussion of the \text{PATTERNSEQUENCE} algorithm and of its components. The algorithm is expressed from a robot’s point of view, and it is executed, from the start, by a robot every time it performs the \text{Compute} operation.

B. SET UP AND LEADERS IDENTIFICATION

Since the robots agree on chirality, it follows that, in the initial configuration \( \Gamma \), the robots can compute and agree on a total ordering of the symmetry classes of \( \Gamma \). For instance, one such ordering can be obtained as follows: classes are ordered from the ones lying on the largest concentric circle (i.e., the \text{SEC}) to the ones on the smallest concentric circle of \( \Gamma \); if on a concentric circle there is more than one class, these are ordered according to the angles between them in the clockwise direction.

As a consequence, one class can always be elected as leader class. Let \text{IDENTIFYLEADERS}(\Gamma) denote the procedure that determines a leader class among those on the \text{SEC}. The set of robots in this class is denoted by \( R_l \), and they will assume the special color \text{GOLD} (see Algorithm 2). Note that, due to asynchrony, some robots in \( R_l \) might become \text{GOLD} while others are not active yet; still, the identification of \( R_l \) is not affected as long as the robots do not move. Therefore:

Lemma 3: Within finite time from the start of the algorithm, the robots agree on the leaders, and the leaders are colored \text{GOLD}.

### Algorithm 2 SETUP()

\textbf{Requires:} \( \Gamma \) is the current configuration.

\begin{algorithmic}
\State \textbf{If} All robots’ colors are \text{GOLD} and/or \text{OFF} \textbf{Then}
\Statex \( R_l := \text{IDENTIFYLEADERS}(\Gamma) \);
\Statex \( q = |R_l| \);
\Statex \textbf{If} All robots in \( R_l \) are \text{GOLD} \textbf{Then}
\Statex \textbf{Return}
\Statex \textbf{Else}
\Statex \textbf{If} I am in \( R_l \) and I am not \text{GOLD} \textbf{Then}
\Statex \textbf{Become} \text{GOLD}.
\Statex \textbf{Else}
\Statex \( q = \text{Number of robots having color in} \ \{\text{GOLD, GOLD-IN, GOLD-OUT}\}; \)
\Statex \textbf{Return}
\end{algorithmic}

Throughout the lifetime of the system, the leaders can transition only through color \text{GOLD-OUT} or \text{GOLD-IN} before becoming \text{GOLD} again. Thus, once the initial set-up has been completed, the set \( R_l \) of leader is always uniquely identifiable by those colors.

C. COLORS

Before describing and analyzing the core steps of the algorithm, we describe the mechanism to decide the colors used by the robots. As already mentioned, the \text{GOLD} color is used exclusively by the leader class.

Our algorithm, as will be evident later, always preserves during its execution the symmetry classes \( Q \) of the initial configuration \( \Gamma \). For each pattern \( S_i \), it maps the symmetry classes \( Q \) to those \( Q_i \) of \( S_i \) (specifying in this way which class of \( S_i \) should be formed by which classes of robots). Let \( \tau_i : Q \rightarrow Q_i \) denote such a mapping; notice that this mapping is not necessarily injective, as \( \alpha(S_i) \) is possibly smaller than \( \alpha(\Gamma) \). To take into account repetitions and contractions in the sequence, as well as non-rigid movements and asynchrony of the robots, colors are assigned to classes carefully during the whole process.

The algorithm uses a set \( \Lambda(Q, S) \) of coloring functions \( \lambda_i : Q \rightarrow C, 0 \leq i < \alpha(\Gamma) \), where \( \lambda_i \) specifies which color the robots of a class should assume when forming \( S_i \). Thus, since \( \alpha(S_i) \) is possibly smaller than \( \alpha(\Gamma) \), \( \lambda_i \) indirectly (i.e., in conjunction with \( \tau_i \)) specifies which set of colors will be associated to each symmetry class of \( S_i \) once the robots form \( S_i \). Let \( S_i \) denote the pattern \( S_i \) colored according to \( \lambda_i \), and let \( q_i(S_i) \) denote its chromatic symmetry.

The set \( \Lambda(Q, S) \), called overall coloring, is chosen so to have the following properties:

**Coloring Properties:**

1. If \( r' \) and \( r'' \) belong to the same symmetry class in \( \Gamma \), then \( \forall i, \lambda_i(r') = \lambda_i(r'') \). That is, robots that initially belong to the same symmetry class, will never be assigned different colors.
2. The number of chromatic symmetry classes of each colored pattern is the same and equal to the number
of classes of the initial configuration; i.e., \( \forall i, q_i(S_i) = \alpha(T) \).
3) For \( i \neq j \), \( \hat{S}_i \neq \hat{S}_j \).
4) \( \forall i, j, \lambda_i^{-1}(\text{GOLD}) = \lambda_j^{-1}(\text{GOLD}) \). That is, a \text{GOLD} robot is always assigned that color in every pattern of the sequence.
5) \( \forall i \), the class colored \text{GOLD} according to \( \lambda_i \) is on the \text{SEC} of \( S_i \).
6) For \( i \neq j \), \( \hat{S}_i \neq \text{GOLD} \neq \hat{S}_j \neq \text{GOLD} \), where \( \hat{S}_i \neq \text{GOLD} \) denotes the colored pattern \( \hat{S}_i \) from which the color \text{GOLD} has been removed; if the class colored \text{GOLD} had no other colors in \( \hat{S}_i \), then \( \hat{S}_i \neq \text{GOLD} \) denotes the colored pattern \( \hat{S}_i \) from which that class has been removed.

Any overall coloring with such properties, if deterministically identifiable from \( Q \) and \( S \), can be employed by our algorithm. Note that there are many such colorings. Consider, for example, the overall coloring \( A^*(Q, S) \) defined by the following process: (i) Assign a distinct color to each class of \( Q \), with \text{GOLD} assigned to the leader class, and \text{MUTANT}_0 \) to an arbitrarily chosen class, called \text{mutant} class; (ii) every class maintains its originally assigned color in all patterns, except for the mutant class which assumes a new color \text{MUTANT}_1 \) at each repetition of the same pattern (see the example depicted in Figure 4). This overall coloring clearly satisfies the Coloring Properties, and is deterministically identifiable from \( Q \) and \( S \).

In the following, we denote by \( A = A(Q, S) \) the global coloring employed by the algorithm. As already mentioned, the leader class determined in the \text{SetUp} step is assigned color \text{GOLD} by \( A \). We will denote by \text{GetColors}(S_i) \) the routine that returns the coloring function \( \lambda_i \) for \( S_i \) according to \( A \); finally, we will say that a robot is \( \lambda_i \)-colored, or simply colored, when colored according to \( \lambda_i \).

In addition to the colors employed by \( A \), the algorithm uses some special service colors: \text{WHITE-S}, \text{WHITE-R}, \text{WHITE-F}, \text{BRONZE}, \text{SILVER}, \text{PLATINUM}, \text{GOLD-IN}, \text{GOLD-OUT}, \text{BLACK}, \) as well as \text{OFF} which is the initial color of all robots.

D. PATTERN IDENTIFICATION

This step of the algorithm is used to identify the pattern of the choreography that the robots should form next or are currently forming (refer to Algorithm 3). To aid in this process, two tools are used by the algorithm.

Algorithm 3 PatternIdentification()

Requires: The current configuration.

\( R_i := \) set of robots colored \text{GOLD}, \text{GOLD-OUT}, and \text{GOLD-IN};

If Color of all robots but those in \( R_i \) is \text{OFF} Then

\( P := S_0 ; \)

\( \text{SIGNAL}(P) ; \)

Else

If \( \exists \lambda_i \in A[R \setminus R_i] \) are colored according to \( \lambda_i \) \{ \text{GOLD, GOLD-OUT, GOLD-IN} \} Then

\( P := S_{i+1 \ mod \ m} ; \)

\( \text{SIGNAL}(P) ; \)

Else

\( \# \) A pattern is already under formation */

\( \text{rad} := \) Radius of \text{SEC}(R) ;

\( c := \) Center of \text{SEC}(R) ;

\( \text{rad'} := \) Radius of smallest non-degenerated populated circle centered in \( c ; \)

\( P := S_i, 0 \leq i \leq m - 1, \) such that \( f(S_i) \cdot \text{rad'} = \text{rad} ; \)

Return

The first tool is an injective function \( f : S \mapsto \mathbb{R} \) that maps each element \( S_i \in S \) into a real number \( f(S_i) \). The only requirement on \( f(S_i) \) is the following: let \( C \) be the smallest populated circle in \( \Gamma \), let \( S(C) \) be the pattern \( S \) scaled so that its smallest populated circle has the same radius as \( C \), let \( \text{SEC}(C) \) be the \text{SEC} of \( S(C) \), and let \( \text{SEC}(C) \) be the largest of all \( \text{SEC}(C) \); then, each \( f(S_i) \) must be greater than the radius of \( \text{SEC}(C) \).

This mapping is used to inform the non-leader robots of the pattern of the sequence, say \( S_i \), they are currently forming or should form. This is done by having the leader robots positioning themselves so that the smallest non-degenerated

![Figure 4](image_url)

**Figure 4.** An example of coloring function (Section IV-C) with one \text{mutant} class (the squared robots).
formed, at the non-leader robots, the leaders know the pattern to be $\text{GOLD}$ formation; in both cases, it identifies the pattern, say $P$ (described in Algorithm 3) determines whether (1) a new $\text{ET}$ that, at the very beginning, after the $S$ go so to communicate the identity of the next pattern. Note that, during these movements, the leaders temporarily change color to $\text{GOLD-OFF}$, so that this transition phase can be clearly identified by all robots. Within finite time, a configuration in $\text{RATIO}(f(P))$ is formed, signalling to the other robots that pattern $P \in S$ has to be formed, and the leaders change color back to $\text{GOLD}$. Summarizing:

\begin{algorithm}
\caption{$\text{Signal}(P)$}
\begin{algorithmic}
\STATE $x := \text{Compute} \ f(P)$;
\STATE $c := \text{Center of} \ \text{SEC}({R} \ \setminus \ R_l)$;
\STATE $\text{rad} := \text{Radius of smallest non-degenerated populated circle centered in} \ c$;
\STATE $\overline{w} := \text{half-line from} \ c \ \text{through my current position}$;
\STATE $p := \text{Point on} \ \overline{w}$, at distance $\text{rad} \cdot x$ from $c$;
\IF {I am $\text{GOLD}$ and I am not at $p$}
\STATE Become $\text{GOLD-OFF}$.
\STATE Move towards $p$.
\ELSE
\IF {I am $\text{GOLD-OFF}$}
\STATE If I am not at $p$ Then
\STATE Move towards $p$.
\ELSE
\STATE Become $\text{GOLD}$.
\ENDIF
\ENDIF
\IF {All $R_l$ are $\text{GOLD}$ and at distance $\text{rad} \cdot x$ from $c$}
\STATE \textbf{Return}
\ENDIF
\end{algorithmic}
\end{algorithm}

Lemma 4: Starting either from the initial configuration or from any $S_i$ colored according to $\lambda_i$, in finite time the leaders reach, avoiding any collision, a configuration where they are at distance $f(S_{i+1} \mod m)$.

\textbf{Proof:} First of all, note that once a robot turns its color from $\text{OFF}$ to some other color, it will never become $\text{OFF}$ again. Hence, by Lemma 3, within finite time the leader class $R_l$ is identified, and those robots are colored $\text{GOLD}$.

Let us assume now that, at time $t$, the robots are either in the initial configuration (i.e., all robots, but the leaders, are colored $\text{OFF}$); or they form pattern $S_i \in S$, with all robots colored according to $\lambda_i$. As soon as a robot becomes active after time $t$, we distinguish the two possible cases:

1) At time $t$ all robots are $\text{OFF}$ (i.e., we are in the initial configuration, and the robots are about to form $S_0$); hence, $\text{Signal}(P)$ is executed, which places the robots in a configuration in $\text{RATIO}(f(S_0))$. In particular, the only robots that move are the leaders: they first compute $f(S_0)$; then, they change their color to $\text{GOLD-OFF}$. Once all leaders are colored $\text{GOLD-OFF}$, they all move radially at distance $\text{rad} \cdot f(S_0)$ from $c$, where $c$ is the center of $\text{SEC}({R} \ \setminus \ R_l)$ and $\text{rad}$ is the radius of the smallest non-degenerated populated circle centered in $c$. Once they reach such a point, by construction, $\text{RATIO}(f(S_0))$ is achieved and the lemma follows.
2) At time $t$ the robots form pattern $S_t \in \mathcal{S}$, with all robots colored according to $\lambda_t$; that is, the robots just completed the formation of $S_t$, and the next pattern to be formed is $S_{(i+1)\mod m}$. As soon as a robot becomes active after time $t$, it executes $\text{Signal}(P)$.

Following the argument of the previous case, the leaders compute $f(S_{(i+1)\mod m})$, change color to $\text{Gold-Out}$, and they all move radially at distance $\text{rad} \cdot f(S_t)$ from $c$. During these moves, no other robot is allowed to move; hence, within finite time all leaders will reach $p$. At this time, by construction, $\text{Ratio}(f(S_{(i+1)\mod m}))$ is achieved and the lemma follows.

Finally, we note that both Algorithm 2 and 3 are collision free: in fact, the only robots that are allowed to move are the leaders, and they move outwards radially starting from the $\text{SEC}$ of the current configuration.

After the leaders reach distance $f(\cdot)$, the non-leader robots will start forming the next pattern in the sequence within the circle $C$, by performing the $\text{Separate}$, $\text{Rotate}$, and $\text{Form}$ steps of the algorithm. Finally, if $\text{PatternIdentification}()$ instead determines that pattern $P$ is already under formation, the robots will continue with their execution of the algorithm.

## E. Separation

After $\text{PatternIdentification}()$, the next step of the algorithm is $\text{Separation}()$ (Algorithm 5). The purpose of this step is to create a configuration where each of the $\alpha(\Gamma)$ classes of the initial configuration is on a distinct concentric circle. This step uses the control color $\text{Silver}$ and the service color $\text{White-S}$. Before describing it and discussing its behaviour and properties, some useful definitions of properties of a configuration during the execution of this step are in order.

**Definition 5 (Ready-to-Separate):** We say that a configuration $\Gamma$ is ready-to-separate under coloring $\lambda_i \in \Lambda$ if the following two conditions hold:

1) the robots on $\text{SEC}$ are $\text{Gold}$ and $\Gamma \in \text{Ratio}(f(S_t))$;

2) the robots not on $\text{SEC}$ are either all $\text{Off}$ or they form $S_{i-1}\text{-Gold}$.

Note that the condition that all robots not on $\text{SEC}$ are $\text{Off}$ is satisfied only after the first execution of $\text{PatternIdentification}()$ (Algorithm 3).

**Definition 6 (In-Separation):** We say that a configuration $\Gamma$ is in-separation under coloring $\lambda_i \in \Lambda$ if the following conditions hold:

1) the robots on $\text{SEC}$ are $\text{Gold}$ and $\Gamma \in \text{Ratio}(f(S_t))$;

2) among the robots not on $\text{SEC}$:
   a) there is at least a $\text{Silver}$ robot; or
   b) there is at least a $\text{White-S}$ robot and $\sigma(\Gamma) \neq \alpha(\Gamma)$; or
   c) not all robots are $\text{White-S}$ and $\sigma(\Gamma) = \alpha(\Gamma)$.

**Definition 7 (Separated):** We say that a configuration $\Gamma$ is separated under coloring $\lambda_i \in \Lambda$ if the following two conditions hold:

1) the robots on $\text{SEC}$ are $\text{Gold}$ and $\Gamma \in \text{Ratio}(f(S_t))$;

2) the robots not on $\text{SEC}$ are all $\text{White-S}$, and $\sigma(\Gamma) = \alpha(\Gamma)$.

The $\text{Separation}(\Gamma)$ step transforms a ready-to-separate configuration $\Gamma$ into a separated configuration $\Gamma'$ where each of the $\alpha(\Gamma)$ classes of the initial configuration is on a distinct concentric circle. Let $\text{Cir}_1, \ldots, \text{Cir}_{\alpha(\Gamma)}$ denote the populated concentric circles of the ready-to-separate $\Gamma$, with $\text{Cir}_1$ being the smallest one. Informally, starting from $i = 1$, if $\text{Cir}_i$ contains multiple (chromatic) classes, we move one class at a time, in an ordered fashion, to a slightly smaller concentric circle, until all classes originally on $\text{Cir}_i$ have been separated on different circles. When a class is alone on a circle, it is then colored $\text{White-S}$; the process is then applied to $\text{Cir}_{i+1}$.

In more detail, the separation is performed by iterating the strategy described below.

Since at the beginning of the separation process the configuration is not separated yet, there exists at least one circle that contains multiple classes: let $C_B$ be the largest one. Since the robots agree on a common chirality, the classes on $C_B$ can be totally ordered: let $K_1, \ldots, K_b$ be the distinct classes on $C_B$. Now, all the robots in $K_1$ become $\text{Silver}$, and no robot can move as long as there is a non $\text{Silver}$ robot in $K_1$. In particular, this process is carried out as follows: let $q$ be the number of $\text{Gold}$ robots; if a robot $r \in K_1$ observes at least two classes on $C_B$ and it sees that the number of $\text{Silver}$ robots on $C_B$ is not $q$, then $r$ colors itself $\text{Silver}$. After a finite amount of time, all robots in $K_1$ become $\text{Silver}$. Once this occurs, $C_B$ contains an integral number $k$ of classes, with $k \geq 2$, and exactly $q$ robots on $C_B$ are $\text{Silver}$ (those in $K_1$).

Let $C_A$ be the largest populated circle, smaller than $C_B$, that does not contain any $\text{Silver}$ robot, and let $\text{dis}$ be the distance between $C_A$ and $C_B$. Now the algorithm forces all robots in $K_1$ to move inwards on the circle at distance $\text{dis}/2$ from $C_B$, call it $C_T$. Note that, during their movements, due to asynchrony and non-rigidity, these $\text{Silver}$ robots will create circles between $C_B$ and $C_A$ that contain only $\text{Silver}$ robots. Routine $\text{MoveToCir()}$ (Algorithm 6) makes sure that all $\text{Silver}$ robots reach $C_T$. Once all $\text{Silver}$ robots reach $C_T$, $C_T$ is the only circle that is populated with just $\text{Silver}$ robots: these robots can now uncolor themselves.

Now $C_B$ contains one less class. These operations are repeated until $C_B$ contains only one class; when this happens this class uncolors itself. The entire process is applied to the new largest circle that contains multiple classes, until the configuration is fully separated.

**Lemma 8:** Starting from a ready-to-separate configuration $\Gamma$ under coloring $\lambda$, within finite time the robots reach a separated configuration $\Gamma'$ under coloring $\lambda$, such that $\alpha(\Gamma') = \sigma(\Gamma) = \alpha(\Gamma)$.

**Proof:** Since $\Gamma$ is ready-to-separate, then, by definition, no robot is $\text{Silver}$ or $\text{White-S}$. Any robot observing $\Gamma$ in its $\text{Look}$ operation, will correctly identify the largest circle $C$ other than $\text{SEC}$ (which contains only the $\text{Gold}$ robots);
hence, within finite time, one or more of the robots on $C$ will independently and asynchronously become \texttt{WHITE-S} or \texttt{SILVER} (depending on whether on $C$ there is one or more than one class, respectively). As soon as the first of these robots changes color, the configuration becomes in-separation and appears as such to any robot performing its
Algorithm 5 Separation(Γ)
Requires: The current configuration Γ.

If Γ is ready-to-separate Then
    C = largest populated circle other than the SEC;
    If C contains only one class and I am in C Then
        Become WHITE-S.
    Else
        K₁, ..., Kₖ = The classes on C;
        If I am in Kᵢ Then
            Become SILVER.
    If Γ is in-separation Then
    If I am SILVER Then
        C = The largest circle that contains SILVER robots;
        If C contains colored robots Then
            C₄ = C;
        Else
            C₄ = Smallest populated circle, larger than C, where
                there are colored robots;
            C₃ = The largest populated circle, smaller than C₄, that
                does not contain SILVER robots; if no such a circle
                exists then C₃ = center of SEC;
            If C contains q robots and C is halfway between C₄ and
                C₃ Then
                Become WHITE-S.
    Else
        If there are q SILVER robots Then
            %I should move to destination if not there already%
            MoveToCir(C₄, C₅).
    Else
        C = largest populated circle where there are colored robots;
        If I am in C and I am colored Then
            If C contains only one class Then
                Become WHITE-S.
            Else
                If there are no SILVER or all SILVER robots are
                    on C Then
                    K₁, ..., Kₖ = The distinct classes on C;
                If there are < q SILVER robots and I am in
                    Kᵢ Then
                    Become SILVER.

Look operation. Notice that no robot can move during this time.

From this moment on, the algorithm proceeds as follows. Let C₄ be the largest circle with no GOLD robots on it, con-
taining colored robots; let K₁, ..., Kₖ be the colored classes
of robots on C₄, and let t₀ be the first time a robot executes the Separation(Γ) process.

1) If k = 1, according to the Algorithm, within finite
time these robots will first become SILVER, and then
WHITE-S, and no robot can move during this time.

Algorithm 6 MoveToCir(C₄, C₅)
dis := Distance between C₄ and C₅;
C₇ = Circle of radius dis/2
If I am not on C₉ Then
    Move inwards radially to C₉.

2) If k > 1, let C₄ be the largest circle smaller than C₅
with colored robots on it, and C₅ be the circle halfway
between C₆ and C₇.
    a) Within finite time, say at time t₁ > t₀, all robots in
       K₁ become SILVER, and no robot moves between
       t₀ and t₁.
    b) Within finite time, each SILVER robot will leave
       C₆, moving radially toward C₇ (MoveToCir(C₄, C₅));
       during this time no other robot is allowed to
       move, hence C₆, C₇, and C₈ will stay invariant. Hence,
       within finite time, say at time t₂ > t₁, all SILVER
       robots will be on C₇. Note that at this time the
       number of chromatic classes is still α(Γ).
    c) Once all SILVER robots are on C₇ (hence, there
       are exactly q robots on it), each SILVER robot
       becomes WHITE-S and, within finite time, say at
       time t₃ > t₂, all robots on C₇ become WHITE-S.

Thus, at time t₃, the number of classes on C₅ decreased
by one. By iterating the above process, within finite
time, say at t₄ > t₃, between C₅ and C₆ there will be
k − 1 circles each containing exactly one class, each
colored WHITE-S (note that at time t₄ also C₅ contains
one class colored WHITE-S).

Finally, by iterating the above argument, the claim follows. ■

F. TARGETS

Before continuing the description and analysis of the algo-

rithm, some discussion is now required on the mechanism
used to decide where each robot should go to form the next
pattern Sᵢ.

As shown in the previous section, procedure Separation()
disassembles the previous configuration (or initially Γ) so
that each symmetry class Q ∈ Q of Γ is in a distinct
concentric circle. Once this is done, the algorithm will start
the assembly of the next pattern.

In this process the algorithm uses the mapping ℰᵢ : Q →
Qᵢ, where Qᵢ denotes the set of symmetry classes of Sᵢ.
It assigns to each class Q ∈ Q a unique symmetry class
ℰᵢ(Q) ∈ Qᵢ of Sᵢ, called target class, following the global
ordering of the symmetry classes of Sᵢ (this ordering is
uniquely identified because of the common chirality). In other
words, the mapping ℰᵢ, called targeting function, specifies
that the goal of the robots of Q should be to move to form
ℰᵢ(Q). Notice that this mapping, while surjective, is not nec-
necessarily injective since α(Sᵢ) is possibly smaller than α(Γ).

Among the many possible targeting functions usable by the
algorithm, we are going to employ the one described in the
following. Let Q = Q₁, ..., Qᵣ be the a = α(Γ) classes,
we will use the concepts of targets and targeting function each on a distinct concentric circle, after S different next pattern
that of $S_k$; where $\phi_1 = 4/4 = 1$; $k = \lceil 4/1 \rceil$, for $1 \leq j \leq 4 + 1 = 4$. Hence, $\tau_i(Q_1) = 1$, $\tau_i(Q_2) = 2$, $\tau_i(Q_3) = 3$, $\tau_i(Q_4) = 4$. Here, the only extra class is $Q_0$, whose target is $Q_{\star}$. (c) A different next pattern $S_j$ to be formed, where: $a = 2$; $\phi_j = 8/4 = 2$; $k = \lceil 7/2 \rceil$, for $1 \leq j \leq 2 * 2 = 4$. Hence, $\tau_j(Q_1) = 1$, $\tau_j(Q_2) = 1$, $\tau_j(Q_3) = 2$, $\tau_j(Q_4) = 2$. Also, here the only extra class is $Q_3$, whose target is $Q_{\star}$. each on a distinct concentric circle, after SEPARATION(), where $Q_j$ is in a smaller circle than $Q_{j+1}$ ($1 \leq j < a$). Let $Q' = Q'_1, \ldots, Q'_{a'}$ be the (totally ordered) $a' = a(S_i) \leq a$ classes of $S_i$, where $Q'_k$ is in the same circle as or in a smaller circle that $Q_{k+1}'$ ($1 \leq k \leq a'$). Let $\phi_i = \frac{a(S_i)}{a(G')}$. Then $\tau_i(Q_k) = Q'_1 \tau_i(Q_1) = \min\{a', \lceil \frac{a}{a'} \rceil \}$.

In other words, $\phi_i$ classes in $\Gamma$ are needed to form a single class in $S_i$. Hence, there are $e = a - a' \cdot \phi_i$ classes which are not necessary to form $S_i$; these extra classes will be all positions on $Q'_{a'}$ (refer to Figure 7).

G. ROTATION

After SEPARATION(), the next step of the algorithm is ROTATION(), reported in Algorithm 7: this step begins when the current configuration $\Gamma$ is separated under some coloring $\lambda_i \in \Lambda$. By construction, after SEPARATION() the robots on each of the concentric circles $Cir_i$ on $\Gamma$, with $1 \leq i \leq \sigma(\Gamma) = \alpha(\Gamma)$, belong to the same symmetricity class, and on each $Cir_i$ there is exactly one symmetricity class. In the following, we will use the concepts of targets and targeting function introduced and defined in the previous section (Section IV-F).

The aim of this phase is to rotate each symmetricity class in $\Gamma$ so that, if each $Q_j$, $1 \leq j \leq a$, is projected onto $r(Q_j)$, then $S_j$ would be formed. More specifically, let us first introduce the following definition (refer to Figure 8):

Definition 9 (Properly Rotated): A properly rotated robot is defined as follows:

1) All robots on $Q_1$ are properly rotated.
2) Robots on $Q_j$, with $j \leq a' \cdot \phi_i$ and $(j \mod \phi_i) \neq 1$, are properly rotated if they form an angle of $\theta' = 360\theta=q(S_i)$ degrees with respect to $Q_{j-1}$.

3) Given $Q_j$, with $j \leq a' \cdot \phi_i$ and $(j \mod \phi_i) = 1$, let $Q'_j = \tau(Q_j)$, and $\theta''$ be the clockwise angle between $Q'_j$ and $Q_j$. Robots in $Q_j$ are properly rotated are all robots in $Q'_r$, $r < j$, are properly rotated, and $Q_j$ forms a clockwise angle of $\theta''$ with respect to $Q_1$.

4) Given $Q_j$, with $j > a' \cdot \phi_i$, robots in $Q_j$ are properly rotated if all robots in $Q'_r$, $r < j$, are properly rotated, and $Q_j$ form an angle of zero degrees with respect to $Q_{a'}$. We will refer to the angle $\theta''$ used in 2) and 3) as the proper rotation angle. We will say that $Q_j$ is properly rotated if all robots on $Q_j$ are properly rotated.

This phase uses service colors BRONZE and WHITE-R to rotate all robots, one class at the time according to their total order so that, within finite time, all of them are properly rotated. In particular, by definition, all robots on $Q_1$ are properly rotated, so they just color themselves WHITE-R. Then, robots on $Q_2$ color themselves BRONZE, compute their proper rotational angle, and rotate so to become properly rotated; once their rotation process is over, they all color themselves WHITE-R. And so on, one concentric circle at a time, until also the robots on SEC (the GOLD robots) rotate, completing this phase.

Note that the existence of a BRONZE robot in some $Cir_i$ implies that $Cir_i$ is under the process of rotation. When the rotation is completed all robots will become either WHITE-R (the non leaders) or GOLD (the leaders). This process is denoted by ROTATION($\Gamma$), and reported in Algorithm 7, where routine ProperlyRotated() and ComputeProperRotationAngle() are defined according to Definition 9.

In the following, we will also make use of the following definitions:

Definition 10 (In-Rotation): We say that a configuration $\Gamma$ is in-rotation under coloring function $\lambda_i \in \Lambda$ if the following conditions hold:

1) the robots on SEC are GOLD, and $\Gamma \in RATIO(f(S))$;
2) the robots are on $\sigma(\Gamma) = \alpha(\Gamma)$ concentric circles each containing $q$ robots, the robots in $Q_1$ are all WHITE-R, and there are robots not WHITE-R (beside the leaders).

Definition 11 (Rotated): We say that a configuration $\Gamma$ with coloring function $\lambda$ is rotated if the following conditions hold:

1) the robots on SEC are GOLD, and $\Gamma \in RATIO(f(S))$, for some $S_i \in S$;
2) the robots on SEC are all WHITE-R, and $q_i(\Gamma, \lambda) = \sigma(\Gamma)$;
3) all robots are properly rotated.

Lemma 12: Starting from a separated configuration $\Gamma$, with coloring function $\lambda$, within finite time the robots reach a rotated configuration $\Gamma'$ with coloring function $\lambda'$ such that $q_i(\Gamma, \lambda) = q_i(\Gamma', \lambda')$, $\sigma(\Gamma) = \sigma(\Gamma')$.

Proof: By hypothesis, in $\Gamma$ there are either WHITE-S or GOLD robots. By construction, all robots on $Q_1$ are properly rotated (these robots are on $Cir^1$); hence, by Algorithm 7,
within finite time, say at time $t$, they all become $\text{WHITE-R}$, and no other robot is allowed to move until this time.

Now, let $C_i, 1 < i \leq \sigma(\Gamma)$, be the smallest concentric circle that it is not properly rotated. Also, let $t_0 \geq t$ be the first time any robot executes the $\text{ROTATION}(\Gamma)$ process.

1) Within finite time, say at time $t_1 > t_0$, all robots on $C_i$ will become $\text{BRONZE}$, and no robot moves between time $t_0$ and $t_1$. Thus, at $t_1$, on $C_i$ there are $q$ $\text{BRONZE}$ robots.

2) Now, the robots on $C_i$ compute the proper rotation angle $\theta$, according to Definition 9, and they rotate on $C_i$ of an angle $\theta$. Within finite time, say at $t_2 > t_1$, all robots on $C_i$ complete their rotation.

3) Finally, within finite time, say at $t_3 > t_2$, all robots on $C_i$ change their color: they become $\text{WHITE-R}$.

Finally, by Definition 9 and by induction on the number of concentric circles in $\Gamma$, the lemma follows.

**H. FORMATION**

In this last phase, the robots start from a separated and rotated configuration $\Gamma$; its outcome is to bring the robots in their correct final positions, so that the next pattern of the sequence, say $S_l$, is formed.

Let us first compute the positions of the target circles where the robots have to move to form $S_l$, call them $T_1, \ldots, T_{\sigma(S_l)}$; these are obtained by mapping the smallest concentric circle of $S_l$ (i.e., the circle where the first class $Q'_1 = \tau_i(Q_1)$ resides), onto the smallest concentric circle of $\Gamma$ (i.e., the one containing $Q_1$), and by scaling consequently all other concentric circles of $S_l$. Also, let $k_l$ be the number of classes that have to end up on $T_1$ in order to correctly form $S_l$.

**Definition 13 (In-Formation):** We say that a configuration $\Gamma$ with coloring function $\lambda$ is in-formation if the following conditions hold:

1) the robots on $\text{SEC}$ are $\text{GOLD}$, and $\Gamma \in \text{RATIO}(f(S_l))$, for some $S_l \in S$;

2) the robots not on $\text{SEC}$ are either $\text{PLATINUM}$ or $\text{WHITE-R}$ or $\lambda$-colored; if a robot is $\lambda$-colored, it is on one of the target circles.

**Definition 14 (Formed):** We say that a configuration $\Gamma$ with coloring function $\lambda$ is formed if $\Gamma = S_l$, for some $S_l \in S$, and $\lambda = \lambda_l$.

We say that a target $T_l$ is complete with respect to $S_l$ at time $t$ if the positions of the robots on $T_l$ form $\text{Cir}_{T_l}^S$, and they are correctly colored according to $\lambda_l$. By convention, we will say that $T_0$ is complete.

By construction, the robots on the smallest concentric circle in $\Gamma$, $\text{Cir}_{T_1}$, are already on $T_1$, i.e. they are on the correct positions to form $S_l$. The formation process iterates on all concentric circles in $\Gamma$, one at the time, from the smallest to the largest one.

In particular, let us assume that the first $j-1$ concentric circles in $\Gamma$ are complete and that there are no robots below $T_{j-1}$ that are not on their final targets; hence, $T_j$ is the first target...
that is not complete. The algorithm, reported in Algorithm 8, distinguishes the only three possible cases:

1) Between $T_j$ and $T_{j-1}$ there are robots whose targets are not $T_j$: these robots must move above $T_j$. In this case, the class of robots whose target is not $T_j$ and that is closest to $T_j$ is first colored PLATINUM, and then moved to a new circle larger than $T_j$. When this occurs, the number of classes whose target is not $T_j$, between $T_j$ and above $T_{j-1}$, decreases; this process is repeated until this case no longer applies.

2) Between $T_j$ and $T_{j-1}$ there are only classes whose target is $T_j$: in this case, these robots are moved, one class at the time, to $T_j$, using the special PLATINUM color. After finite time, this class reaches $T_j$, and the number of classes on $T_j$ increases.

3) Between $T_j$ and $T_{j-1}$ there are no robots: the class of robots that is on the smallest circle closest to $T_j$ whose target is $T_j$ is moved onto $T_j$, using the special PLATINUM color. After finite time, this class reaches $T_j$, and the number of classes on $T_j$ increases.

Thus, after finite time, all circles in $\Gamma$ are correctly processed, and $S_i$ correctly formed.

**Lemma 15:** Starting from a rotated configuration $\Gamma$ with coloring function $\lambda$, within finite time the robots reach a formed configuration $\Gamma'$ with coloring function $\lambda'$ with respect to $S$, where $S$ is the next pattern of the sequence to be formed, such that $q_c(\Gamma, \lambda) = q_c(\Gamma', \lambda')$.

**Proof:** By definition, right after the rotation process in $\Gamma$ there are no PLATINUM robots; in particular, all robots not on SEC are WHITE-R, and those on SEC are GOLD. By construction, the robots on the smallest concentric circle of $\Gamma$ are already on their target, $T_1$: within finite time, say at time $t_0$, they have computed and assumed the color assigned by routine GETCOLORS($S$). Starting from this moment,
FIGURE 10. An example for the formation step of FORMATION(\textsuperscript{1}) in Algorithm 8. The \textsuperscript{GOLD} robots are not included in the figure.

(a) The next pattern to be formed, \(S_i\).
(b) Starting from the rotated configuration of Figure 9(a), the robots on the smallest concentric circle are already on their target circle \(T_1\), hence they become colored. Dotted circles represent target circles.
(c) The robots on the second smallest circle have to reach the second target circle \(T_2\) (\(u_{p2} = 1\)); before moving, they all become \textsuperscript{PLATINUM}. Let us assume that 2 classes of robots have as target \(T_2\) (i.e., \(k_2 = 2\)).
(d) Here, \(0 < u_{p1} = 1 \leq k_2 = 2\); hence, all \textsuperscript{PLATINUM} robots move towards \(T_2\).

(e) Once all \textsuperscript{PLATINUM} robots reach \(T_2\), they can become colored.
(f) Here, \(T_2\) is not completed yet (2 classes of robots have as target \(T_2\)); also, \(u_{p2} = 0\). Hence, the robots on the third smallest circle move towards \(T_2\). Before moving, they all become \textsuperscript{PLATINUM}.
(g) All \textsuperscript{PLATINUM} robots reached \(T_2\), that is now completed.
(h) The \textsuperscript{PLATINUM} robots on \(T_2\) can now become colored.

(i) Finally, \(T_3\) is addressed: here, \(u_{p3} = 0\), and \(k_3 = 1\). Thus, the robots on the third smallest circle move towards \(T_3\). Again, before moving, they all become \textsuperscript{PLATINUM}.
(j) All \textsuperscript{PLATINUM} robots reached their final destinations on \(T_3\), that is now completed.
(k) The formation step has been completed for all but \textsuperscript{GOLD} robots: they will be the last ones to reach the last target circle.

let \(T_j, j \geq 1\), be the smallest target circle which is not complete; i.e., some of the classes that have to be on \(T_j\) are still missing. Also, let us denote by \(u_{p_j}\) the number of classes between \(T_{j-1}\) and \(T_j\), including \(T_j\), whose target is not \(T_j\) (by convention, \(T_0\) is complete). The algorithm now distinguishes three cases:

1) \(u_{p_j} > 0\). In this case, we have that between \(T_{j-1}\) and \(T_j\) there are robots whose target is not \(T_j\).
   a) By construction of routine \textsc{MoveFromToDes()} (Algorithm 9), within finite time the class of robots (whose target is not \(T_j\)) on the largest concentric circle between \(T_{j-1}\) and \(T_j\), including \(T_j\), say \(C_{from}\), starts becoming \textsuperscript{PLATINUM}. Hence, there is a time \(t_1 > t_0\) where there are exactly \(q\) \textsuperscript{PLATINUM} robots on \(C_{from}\).

b) At this stage, the goal of the algorithm is to move above \(T_j\) all robots on \(C_{from}\). In particular, all these robots will be moved to an empty circle \(C_{des}\) computed as follows.
Let \(C^*\) be the smallest circle populated by non \textsuperscript{PLATINUM} robots between \(T_j\) and \(T_{j+1}\), including \(T_{j+1}\) (note that, at time \(t_1\), there are no \textsuperscript{PLATINUM}
If \( S. \) Das

If \( \Gamma \) is in-rotation Then
  If \( \Gamma \) contains BRONZE robots Then
    \( C := \) the circle that contains BRONZE robots;
  If I am in \( C \) Then
    If I am WHITE-S Then
      Become BRONZE.
    If all robots in \( C \) are BRONZE Then
      \( \theta := \) ComputeProperRotationAngle();
      If Rotation of BRONZE robots has not been completed Then
        If I am \( \neg \) ProperlyRotated() Then
          Rotate(\( \theta \)).
        Else
          Become WHITE-R.
  Else
    If \( \Gamma \) contains WHITE-S robots Then
      \( C := \) the smallest circle that contains WHITE-S robots;
    If I am in \( C \) and I am WHITE-S Then
      Become BRONZE.
  If Rotation of robots on SEC has not been completed Then
    If All robots not on SEC are WHITE-R Then
      If There are GOLD robots on SEC Then
        If I am on SEC and I am GOLD Then
          Become WHITE-R.
      Else
        If I am on SEC Then
          \( \theta := \) ComputeProperRotationAngle();
          Rotate(\( \theta \)).
      Else
        If There are robots on SEC that are not GOLD Then
          If I am on SEC and I am not GOLD Then
            Become GOLD.

robots; hence \( C^* \) is the smallest populated circle between \( T_j \) and \( T_{j+1} \). Then, \( C_{\text{des}} \) is the circle halfway between \( T_j \) and \( C^* \). Starting at \( t_1 \), robots on \( C_{\text{from}} \) move radially to \( C_{\text{des}} \), again by calling routine \textsc{MoveFromToDes}(). Within finite time, say at \( t_2 > t_1 \), all \textsc{Platinum} robots will be on \( C_{\text{des}} \).

c) Once all \( q \) \textsc{Platinum} robots are on \( C_{\text{des}} \), they all become \textsc{White-F} at some time \( t_3 > t_2 \).

Note that, between time \( t_1 \) and \( t_3 \) no other robot is allowed neither to move, hence \( C_{\text{des}} \) stays invariant, nor to change color. Thus, at time \( t_3 \), the number of classes between \( T_{j-1} \) and \( T_j \) whose target is not on \( T_j \) (i.e., the value of \( u_p_j \)), decreases by one.

2) \( u_p_j = 0 \) and between \( T_j \) and \( T_{j-1} \) there are only classes whose target is \( T_j \). In this case, all these robots, one class at the time, move to \( T_j \). At this point, the proof is analogous to the previous case, where \( C_{\text{des}} \) is \( T_j \), and the color assigned at the end of the movement is the one assigned by \textsc{GetColors}(S); this is because the robots have reached their final destination to form \( S \). Hence, within finite time, one more class with target \( T_j \) reaches \( T_j \).
Algorithm 9 MOVEFROMToDes(From, Des, MoveColor, FinalColor)

If I am MoveColor Then
If There are q MoveColor robots and I am not on Des Then
Move radially to Des.
If All MoveColor robots are on Des Then
Become FinalColor.
Else
If There are less than q MoveColor robots, and all of them are on From Then
If I am on From Then Become MoveColor.

3) $u_p$ = 0 and all robots whose target is $T_j$ are on circles larger than $T_j$. Also in this case, all these robots, one class at the time, move onto $T_j$ by calling routine MOVEFROMToDES(); the only difference with the previous case is that the robots are moving from a larger circle, $T_{from}$, to a smaller circle, $T_j$. Hence, within finite time, one more class with target $T_j$ reaches $T_j$. Hence, by induction, within finite time, the last class to move is that of the GOLD robots. Once again, this case is handled by routine MOVEFROMToDES(), where the robots use the color GOLD-IN instead of PLATINUM, and GOLD instead of the the color assigned by GETCOLORS(S). Therefore, within finite time all targets become complete. Since, by definition, a complete target $T_j$ coincides with $Cir_i^S$, the theorem follows.

I. $\bar{T}$ SYMMETRIC AND S CENTRAL

Algorithm 1 assumes both $q(\bar{T}) > 1$ and $S$ non-central. In the following, we will briefly outline how to extend it to cover also the case when $q(\bar{T}) > 1$ and $S$ is central.

In this case, by Theorem 2, $S$ is feasible if $q(S^-)$ is a multiple of $q(\bar{T})$, for all $i$. Thus, at the beginning, the first action to do is to move one class to the center $c$ of SEC (note that, being $\bar{T}$ symmetric, there cannot be any robot at $c$ at the beginning); this class will assume the special color BLACK.

In more detail, the SetUp() algorithm (Algorithm 2) is changed as follows: (I) the guard of the first If statement checks also for the presence of BLACK robots (All robots’ colors are GOLD and/or OFF and/or BLACK); (II) instead of the first Return statement, routine FixCentral() (Algorithm 10) is called. This routine ensures that all robots in the class of $\Gamma$ closest to $c$ first change their color to BLACK, and then move to $c$.

At this point the execution runs following Algorithm 1, where BLACK robots never move; also, they are never considered by all the other robots (i.e., it is like they do not exist). In other words, $\Gamma^r$ it is considered instead of $\Gamma$, and $S_i^{-}$ instead of $S_i$, for all $S_i \in S$.

By following the same techniques used in the previous algorithms, it is easy to see that within finite time Algorithm 10 moves one class to $c$: the first class on the smallest populated circle according to the total ordering of the classes. At this point, the BLACK robots will never move again, and the correctness of the overall solution (that covers both central and non-central sequences) follows from previous Lemmas 3–16.

J. ASYMMETRIC INITIAL CONFIGURATION

In case the initial configuration is asymmetric, that is $q(\bar{T}) = 1$, the entire protocol is much simpler. Note that, since $q(\bar{T}) = 1$, each class contains only one robot, and there is a total order of the classes; hence, the robots can agree on a unique leader and on a common orientation of SEC, i.e., there is a common chirality. In the algorithm, the main difference is that we keep invariant the initial SEC, instead of the smallest concentric populated circle of the starting configuration. Another difference is that the unique leader uses color RAINBOW instead of GOLD (and thus RAINBOW-OUT and RAINBOW-IN instead of GOLD-OUT, GOLD-IN) throughout the algorithm.

More precisely, there are only three phases: SetUp, PatternIdentification and Formation (Algorithm 11), described in the following.

Algorithm 10 FixCentral()

Requires: $\Gamma$ is the current configuration.

If There are robots at $c$ and all BLACK are at $c$ Then Return
If No robot is at $c$ and less than $q$ robots are BLACK Then
$K := $ First class of robots closest to the center $c$ of SEC;
If I am in $K$ and I am not BLACK Then
Become BLACK.
Else
If I am BLACK and I am not at $c$ Then
Move radially towards $c$.

1) SETUP PHASE

In this phase, if no robot is colored RAINBOW, the first one in the total order whose removal does not destroy the SEC becomes RAINBOW.

2) PATTERNIDENTIFICATION PHASE

This phase, described in Algorithms 12 and 13, follows an idea similar to that described in Section IV-D. The main differences with Algorithms 3 and 4 is that rad+1 in Algorithm 12 is computed with respect to the SEC \ $\{r_K\}$.

3) FORMATION PHASE

In this phase, we have the following three steps:
Algorithm 12 PatternIdentification()
Requires: The current configuration.

\[ r_g := \text{robot colored } \{\text{RAINBOW}, \text{RAINBOW-OUT}, \text{RAINBOW-IN} \} \];

If Color of robots in \( \mathcal{R} \setminus \{r_g\} \) is OFF Then
\[ P := S_0; \]
\[ \text{SIGNAL}(P); \]
Else
If \( \exists \lambda_i \in \Lambda / \mathcal{R} \setminus \{r_g\} \) are colored according to \( \lambda_i \{\text{RAINBOW, RAINBOW-OUT, RAINBOW-IN} \} \) Then
\[ P := S_{i+1} \mod m; \]
\[ \text{SIGNAL}(P); \]
Else
/* A new pattern is already under construction */
\[ c := \text{Center of } \text{SEC}(\mathcal{R}); \]
\[ \text{rad} := \text{Segment between } c \text{ and the robot RAINBOW}; \]
\[ \text{rad}' := \text{Radius of } \text{SEC}(\mathcal{R} \setminus \{r_g\}); \]
\[ P := S_i, 0 \leq i \leq m - 1, \text{such that } f(S_i) \cdot \text{rad}' = \text{rad} \]
Return

Algorithm 13 Signal(P)
\[ x := \text{Compute } f(P); \]
\[ c := \text{Center of } \text{SEC}(\mathcal{R} \setminus \{r_g\}); \]
\[ \text{rad} := \text{Radius of } \text{SEC}(\mathcal{R} \setminus \{r_g\}); \]
\[ w := \text{half-line from } c \text{ through my current position}; \]
\[ p := \text{Point on } w, \text{ at distance } \text{rad} \cdot x \text{ from } c; \]
If I am RAINBOW and I am not at \( p \) Then
\[ \text{Become RAINBOW-OUT}; \]
Move towards \( p \).
Else
If I am RAINBOW-OUT Then
\[ \text{If I am not at } p \text{ Then} \]
Move towards \( p \).
Else
\[ \text{Become RAINBOW.} \]
If \( R_p \) is at distance \( r \cdot x \) from \( c \) Then Return

1) \( \text{SEC}(\mathcal{R} \setminus \{r_g\}) \) is mapped to \( \text{SEC}(S) \); the first robot on \( \text{SEC}(\mathcal{R} \setminus \{r_g\}) \) is considered already in its final position, and thus changes color according to \( \text{GetColors}(S) \). At this point, the pattern to be formed is scaled and rotated accordingly, and the final destination the robots have to reach in order to correctly form \( S \) can be computed. Now, a one-to-one mapping is established between the other robots in \( \Gamma \) and the other destinations of the pattern to be formed for which there exists a sequential scheduling of the movements of the robots onto the assigned destination, such that the \( \text{SEC}(\mathcal{R} \setminus \{r_g\}) \) does not change during each movement. Note that such a scheduler trivially exists.

2) Each robot, excluding the one colored RAINBOW, one at the time (according to scheduling associated to the one-to-one mapping):
   a) Becomes PLATINUM;
   b) Moves to its destination;
   c) Gets the color assigned by \( \text{GetColors}(S) \).
3) Finally, the only robot that still needs to be placed is the RAINBOW: it moves to the last available target, by using RAINBOW-IN color during its movement. Once it reaches its destination, it becomes RAINBOW.

The correctness of the case with \( q = 1 \) follows the same lines of previous Lemmas 3–15, thus we can conclude that:

**Lemma 16:** Starting from any asymmetric initial configuration \( \Gamma \) (i.e., \( q(\Gamma) = 1 \)), \text{ASYMMETRICPatternSequence} allows the robots to perform any feasible choreography in \( 
\mathcal{F}(\overline{\Gamma}) \).

**K. Final Algorithm**

The final algorithm can be easily obtained by merging the two cases for \( q(\overline{\Gamma}) = 1 \) and \( q(\overline{\Gamma}) > 1 \). The distinction between these two cases is trivially done at the starting of the execution. In any subsequent configuration, it can be done depending on whether there are GOLD or RAINBOW robots.

By previous Lemmas 3–16, we can conclude that:

**Theorem 17:** Starting from any initial configuration \( \Gamma \), the robots can perform any feasible choreography in \( 
\mathcal{F}(\overline{\Gamma}) \).

**V. Limitations to the Number of Colors**

Since the focus of this paper is on the characterization of the set \( \mathcal{F}(\Gamma) \) of sequences of patterns that can be formed by luminous robots starting from an arbitrary initial configuration \( \Gamma \), and the feasibility of constructing them, the number of colors used in the construction has not been a concern. It is however an important element worth discussing.

Our protocol employs a coloring function \( \Lambda \), which assigns colors to the classes of \( \Gamma \) so that, once those classes are mapped by our algorithm to the classes of the pattern to be formed, the Coloring Properties (see Section IV-C) are guaranteed to hold.

From the point of view of characterization and feasibility, the choice of \( \Lambda \) is irrelevant. However, it has an impact on the number of colors used by the robots. For example, the function \( \Lambda^* \) described in Section IV-C, provides the following upperbound:

\[
c(\overline{\Gamma}) \leq \alpha(S) + \mu(S) + O(1)
\]

(1)

The immediate question that arises is how many colors are truly necessary to form any sequence \( S \in \mathcal{F}(\overline{\Gamma}) \), starting from a configuration \( \Gamma \). Note that in the \text{OBLLOT} model, where the robots have a single color, it is possible only to form sequences without repetitions and contractions [9]. Intuitively, this is because, if two robots move to occupy the same point performing a contraction, they will not be able to separate anymore and will thus not be able to form alternatively patterns with different number of vertices.
Moreover, the presence of repetitions would not allow the robots to “understand” which repetition of the same pattern they are currently forming without an external indication. Multiple colors are needed to deal with contractions, repetitions, as well as with asynchrony. We now establish bounds on how many colors are actually needed.

**Theorem 18:** A set of luminous robots starting from initial configuration $\Gamma$, cannot perform a choreography $S^\infty = (S_0, \ldots, S_{m-1})^\infty$ with less than $\max(\alpha(S)/\min_i(\alpha(S_i)), \sqrt[|S|]{\mu(S)})$ colors. The lower bound holds even if the robots are fully-synchronous and rigid.

**Proof:** First of all, the robots need to distinguish between at least $a = \alpha(S)$ distinct classes, during the execution of any algorithm. Even if the initial configuration $\Gamma$ is such that $
abla(\Gamma) > \alpha(S)$, $\Gamma$ can be contracted to a configuration containing $a = \alpha(S)$ distinct classes, without any loss. However, for forming any pattern $S_i \in S$ with $\alpha(S_i) < \alpha(S)$, after the contraction, the robots from distinct classes must be distinctly colored before they are contracted to a single class. Thus at least $\alpha(S)/\alpha(S_i)$ colors are needed to perform this contraction and thus the total number of colors $c(\Gamma)$ must satisfy

$$c(\Gamma) \geq \alpha(S)/\min_i(\alpha(S_i)) \quad \text{(2)}$$

For two patterns $S_i$ and $S_j$, such that $i \neq j$ and $S_i \equiv S_j$, the coloring $\lambda_i$ and $\lambda_j$ must be visibly distinguishable by the robots. With $a$ distinct (ordered) classes and $c$ colors, it is possible to have $c^a$ visibly distinct colorings for patterns having $a$ classes. This gives the following bound on the total number of colors

$$c(\Gamma)^\alpha(S) \geq \mu(S) = \max_i(\mu(S_i)) \quad \text{(3)}$$

Combining the above two inequalities we obtain the following lower bound on the number of colors for forming sequence $S \in F(\Gamma)$

$$c(\Gamma) \geq \max \left\{ \frac{\alpha(S)}{\min_i(\alpha(S_i))}, \sqrt{|S|}/\mu(S) \right\} \quad \text{(4)}$$

There is a large gap between the lower bound of $(4)$ and the upper bound of $(1)$. To narrow this gap, e.g. by identifying a more efficient coloring or by designing a different pattern formation protocol, is a very interesting and important open question.

**VI. CONCLUDING REMARKS**

In this paper we have characterized the set $F(\Gamma)$ of sequences of patterns that can be formed by luminous robots starting from an initial configuration $\Gamma$, showing that $F(\Gamma)$ is much larger than the set of sequences of patterns formable without lights characterized in [9].

Our characterization is complete and constructive: We have provided a universal protocol that, for any initial configuration $\Gamma$ and for any pattern sequence $S \in F(\Gamma)$, allows the robots to form $S$ even if asynchronous and with non-rigid movements.

Note that we have assumed that $\alpha(\Gamma) = \alpha(S)$. It is actually possible that $\alpha(\Gamma) > \alpha(S)$; should this be the case, it suffices to apply a simple pre-processing phase that will reduce the number of classes to $\alpha(S)$. In more detail, all extra classes of $\Gamma$ are moved to the center $c$ of SEC and colored BLACK, following an approach similar to the one adopted in Section IV-I to handle central sequences. From now on: if the next pattern to be formed is central, the BLACK will occupy $c$; otherwise, after all other classes have occupied their final destinations (hence the pattern is formed up to the robots at $c$) each of them will choose as destination one of the points of any other class (e.g., the closest class to $c$), and move there. Once the pattern has been formed they will move back to the center to start the formation of the next pattern.

There are several interesting open questions. The results for formation of sequences of patterns without lights established in [9], and those with luminous robots established in this paper have been derived under the assumption that there is chirality; what happens when there is no chirality? What happens when robots may fail? What kind of failures can be tolerated in forming patterns? Is it possible to have self-stabilizing algorithms when the robots may start from any arbitrary color?

As mentioned in the previous section, the determination of a better upper-bound on the number of colors, through a more efficient coloring or different pattern formation protocol, is an immediate and natural open problem.

**ACKNOWLEDGMENT**

Some of these results were presented at the 7th International Conference on Fun with Algorithms.

**REFERENCES**

[1] A. Heriban, X. Défago, and S. Tixeuil, “Optimally gathering two robots,” in Proc. 19th Int. Conf. Distrib. Comput. Netw., Jan. 2018, pp. 3:1–3:10.

[2] M. Alcántara, A. Castañeda, D. Flores-Peñaloza, and S. Rajsbaum, “The topology of look-compute-move robot wait-free algorithms with hard termination,” Distrib. Comput., vol. 32, no. 3, pp. 235–255, Jun. 2019.

[3] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita, “Distributed memoryless point convergence algorithm for mobile robots with limited visibility,” IEEE Trans. Robot. Autom., vol. 15, no. 5, pp. 818–828, Oct. 1999.

[4] S. Bhagat and K. Mukhopadhyaya, “Optimum algorithm for mutual visibility among asynchronous robots with lights,” in Proc. 19th Int. Symp. Stabilization, Saf., Secur. Distrib. Syst. (SSS), 2017, pp. 341–355.

[5] S. Choi and J. Kim, “Three dimensional formation control to pursue an underwater evader utilizing underwater robots measuring the sound generated from the evader,” IEEE Access, vol. 7, pp. 150720–150728, 2019.

[6] M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro, “Distributed computing by mobile robots: Gathering,” SIAM J. Comput., vol. 41, no. 4, pp. 829–879, Jan. 2012.

[7] S. Das, P. Flocchini, G. Prencipe, N. Santoro, and M. Yamashita, “The power of lights: Synchronizing asynchronous robots using visible bits,” in Proc. IEEE 32nd Int. Conf. Distrib. Comput. Syst., Jun. 2012, pp. 506–515.

[8] S. Das, P. Flocchini, G. Prencipe, N. Santoro, and M. Yamashita, “Autonomous mobile robots with lights,” Theor. Comput. Sci., vol. 609, pp. 171–184, Jan. 2016.

[9] S. Das, P. Flocchini, N. Santoro, and M. Yamashita, “Forming sequences of geometric patterns with oblivious mobile robots,” Distrib. Comput., vol. 28, no. 2, pp. 131–145, Apr. 2015.
