Vol.2015-AL-152 No.10
2015/3/3

Randomized Pattern Formation Algorithm for Mobile Robots

Yukiko Yamauchi[1,a)] Masafumi Yamashita[1,b)]

Abstract: We consider the pattern formation problem by autonomous mobile robots, which is one of the most important problems for distributed control of swarm of mobile robots. We present a randomized pattern formation algorithm for asynchronous oblivious (i.e., memory-less) mobile robots that enables formation of any target pattern. As for deterministic pattern formation algorithms, the class of patterns formable from an initial configuration I is characterized by the symmetricity (i.e., the order of rotational symmetry) of I, and in particular, every pattern is formable from I if its symmetricity is 1. The randomized pattern formation algorithm ψ_{PF} we present in this paper consists of two phases: The first phase transforms a given initial configuration I into a configuration I' such that its symmetricity is 1, and the second phase invokes a deterministic pattern formation algorithm ψ_{CWM} by Fujinaga et al. (DISC 2012) for asynchronous oblivious mobile robots to finally form the target pattern.
There are two hurdles to overcome to realize ψ_{PF}. First, all robots must simultaneously stop and agree on the end of the first phase, to safely start the second phase, since the correctness of ψ_{CWM} is guaranteed only for an initial configuration in which all robots are stationary. Second, the sets of configurations in the two phases must be disjoint, so that even oblivious robots can recognize which phase they are working on. We provide a set of tricks to overcome these hurdles.

1. Introduction

Consider a distributed system consisting of anonymous, asynchronous, oblivious (i.e., memory-less) mobile robots that do not have access to a global coordinate system and are not equipped with communication devices. We investigate the problem of forming a given pattern F from *any* initial configuration I, whose goal is to design a distributed algorithm that works on each robot to navigate it so that the robots as a whole eventually form F from any I. Besides the theoretical interest how the robots with extremely weak capability can collaborate, the fact that self-organization is a key property desired for autonomous distributed systems motivates our work. However, existing papers [2], [3], [4], [5], [6], [7] have showed that the problem is not solvable by a deterministic algorithm, intuitively because the symmetry among robots cannot be broken by a deterministic algorithm. Specifically, let $\rho(P)$ be the (geometric) symmetricity of a set P of points, where $\rho(P)$ is defined as the number of angles θ (in $[0, 2\pi)$) such that rotating P by θ around the center of the smallest enclosing circle of P produces P itself.[*1] Then F is formable from I by a deterministic algorithm, if and only if $\rho(I)$ divides $\rho(F)$, which suggests us to explore a *randomized solution*.

This paper presents a randomized pattern formation algorithm ψ_{PF}. Algorithm ψ_{PF} is *universal* in the sense that for any given target pattern F, it forms F from *any* initial configuration I (not only from I such that $\rho(I)$ divides $\rho(F)$). We however need the following assumptions; the number of robots $n \geq 5$, and both I and F do not contain multiplicities. The idea behind ψ_{PF} is simple and natural; first the symmetry breaking phase realized by randomized algorithm ψ_{SB} translates I into another configuration I' such that $\rho(I') = 1$ with probability 1 if $\rho(I) > 1$, and then the second phase invokes the (deterministic) pattern formation algorithm ψ_{CWM} in [5], which forms F from any initial configuration I' such that $\rho(I') = 1$.[*2] Since randomization is a traditional tool to break symmetry, one might claim that ψ_{PF} is trivial. It is not the case at all, mainly because our robots are asynchronous. We return to this issue later in this section, after a brief introduction of our robot model.

In the literature [2], [3], [4], [5], [6], [7], the robots are modeled by points on a two dimensional Euclidean plane. Each robot repeats a Look-Compute-Move cycle, where it obtains the positions of other robots (in Look phase), computes the curve to a next position with a pattern formation algorithm (in Compute phase), and moves along the curve (in Move phase). We assume that the execution of each cycle ends in finite time. Each robot has no access to the global x-y coordinate system; it has its own x-y local coordinate system, and the robots' positions in Look phase and the curve to its next position in Compute and Move phases are given in its x-y local coordinate system. The x-y local coordinate systems are all right-handed. The robots are oblivious in the sense that the algorithm is a function of the robots' positions (in its x-y local coordinate system) observed in the preceding Look phase. We assume discrete time $0, 1, \ldots$, and introduce three types of

1 Kyushu University, 744 Motooka, Nishi-ku, Fukuoka 819-0395, Japan
a) yamauchi@inf.kyushu-u.ac.jp
b) mak@inf.kyushu-u.ac.jp

*1 That is, P is rotational symmetry of order $\rho(P)$.

*2 Of course we can also use the pattern formation algorithm in [2] since it keeps the terminal agreement of ψ_{SB} (i.e., the leader), during the formation.

© 2015 Information Processing Society of Japan

asynchrony. In the *fully-synchronous* (FSYNC) model, robots execute Look-Compute-Move cycles synchronously at each time instance. In the *semi-synchronous* (SSYNC) model, once activated, robots execute Look-Compute-Move cycles synchronously. We do not make any assumption on synchrony for the *asynchronous* (ASYNC) model.

A crucial assumption here is that a robot can sense the position of another robot, but cannot sense its velocity. In the SSYNC (and hence FSYNC) model, a robot never observe moving robots by definition, while in the ASYNC model, a robot does but cannot tell which of them are moving. This is an essential difficulty in designing a randomized algorithm for the ASYNC model. In this paper, we devise a trick to overcome this problem. Specifically, in order for ψ_{CWM} to start working in safe, in the terminal configuration of ψ_{SB} all robots must simultaneously stop and agree on the end of the symmetry breaking phase. We solve the symmetricity breaking problem in two phases: The *randomized leader election* phase and the *termination agreement* phase. In the randomized leader election phase, robots randomly select the leader on the largest empty circle, which is the largest circle centered at the center of the smallest enclosing circle of robots and contains no robot in its interior. The robots on the largest empty circle move by randomly selected small distance along the circumference of the largest empty circle, and when they break the symmetry, some of the robots enter the interior of the largest empty circle to form a new largest empty circle. They repeat this random selection phase until the system reaches a configuration where exactly one robot is on the current largest empty circle. We call this robot the leader. At this point, some robots may be still circulating on the previous largest empty circles. Now, the problem is to check the termination of these random movements when we have the leader. The leader defines a static destination point for each of these robots, such that they cannot reach by their small random movement. The randomly moving robots should start deterministic new movement. Eventually, all these robots stop and the leader moves closer to the the center of the smallest enclosing circle so that the robots agree the termination. Finally, robots start a pattern formation phase.

Related works. The pattern formation problem in FSYNC model and SSYNC model was first investigated by Suzuki and Yamashita [6], [7]. First, they showed that any target pattern formable by non-oblivious robots in the FSYNC model is formable by oblivious robots in the SSYNC model, except point formation of two robots. They also showed that point formation of two robots is unsolvable in the SSYNC model, while there is a trivial solution in the FSYNC model. Second, they characterized the formable patterns by non-oblivious robots in the FSYNC model. A necessary and sufficient condition to from a target pattern F from a given initial configuration I is $\rho(I)|\rho(F)$. Later, ASYNC model was introduced by Flocchini et al. [3]. Since we cannot apply pattern formation algorithms for the FSYNC or SSYNC model to the ASYNC model, the pattern formation problem in the ASYNC model has been an open problem. Dieudonné et al. proposed a universal pattern formation algorithm with a unique leader for more than three oblivious robots in the ASYNC model [2]. Fujinaga et al. presented an embedded pattern formation algorithm for oblivious robots in the ASYNC model, where each robot obtains an embedded target pattern in its local coordinate system [4]. Their algorithm is based on a minimum weight perfect matching between the target points and the positions of robots, which is called *clockwise matching*. Finally, Fujinaga et al. presented a pattern formation algorithm for oblivious robots in the ASYNC model that uses the embedded pattern formation algorithm [5]. Cieliebak et al. presented a gathering algorithm for more than two oblivious robots in the ASYNC model [1].

All these papers investigate robots with deterministic algorithms. To the best of our knowledge, randomized symmetricity breaking is a new notion which works as a fundamental preprocessing for many tasks of robots.

2. System model

Let $R = \{r_1, r_2, \dots, r_n\}$ be a set of anonymous robots in a two-dimensional Euclidean plane. Each robot r_i is a point and does not have any identifier, but we use r_i just for description.

A *configuration* is a set of positions of all robots at a given time. In the ASYNC model, when no robot observes a configuration, the configuration does not affect the behavior of any robots. Hence, we consider the sequence of configurations, in each of which at least one robot executes a Look phase. In other words, without loss of generality, we consider discrete time $1, 2, \dots$. A robot starting a Look-Compute-Move cycle at time t obtains the positions of other robots at time $t' \geq t$ (Look phase), computes a curve to the next location (Compute phase), and starts moving along the curve at time $t'' \geq t'$ (Move phase). The Move phase finishes at some time $t''' \geq t''$. Let $p_i(t)$ (in the global coordinate system Z_0) be the position of r_i ($r_i \in R$) at time t ($t \geq 0$). $P(t) = \{p_1(t), p_2(t), \dots, p_n(t)\}$ is a configuration of robots at time t. The robots initially occupy distinct locations, i.e., $|P(0)| = n$.

The robots do not agree on the coordinate system, and each robot r_i has its own *x-y local coordinate system* denoted by $Z_i(t)$ such that the origin of $Z_i(t)$ is its current position.[*3] We assume each local coordinate system is right-handed, and it has an arbitrary unit distance. For a set of points P (in Z_0), we denote by $Z_i(t)[P]$ the positions of $p \in P$ observed in $Z_i(t)$.

An algorithm is a function, say ψ, that returns a curve to the next location in the two-dimensional Euclidean plane when given a set of positions. Each robot has an independent private source of randomness and an algorithm can use it to generate a random rational number. A robot is *oblivious* in the sense that it does not remember past cycles. Hence, ψ uses only the observation in the Look phase of the current cycle.

In each Move phase, each robot moves at least $\delta > 0$ (in the global coordinate system) along the computed curve, or if the length of the curve is smaller than δ, the robot stops at the destination. However, after δ, a robot stops at an arbitrary point of the curve. All robots do not know this minimum moving distance δ. During movement, a robot always proceeds along the computed curve without stopping temporarily. We call this assumption *strict progress property*.

[*3] During a Move phase, we assume that the origin of the local coordinate system of robot r_i is fixed to the position where the movement starts, and when the Move phase finishes, the origin is the current position of r_i.

An execution is a sequence of configurations, $P(0), P(1), P(2), \ldots$. The execution is not uniquely determined even when it starts from a fixed initial configuration. Rather, there are many possible executions depending on the activation schedule of robots, execution of phases, and movement of robots. The *adversary* can choose the activation schedule, execution of phases, and how the robots move and stop on the curve. We assume that the adversary knows the algorithm, but does not know any random number generated at each robot before it is generated. Once a robot generates a random number, the adversary can use it to control all robots.

Pattern Formation. A target pattern F is given to every robot r_i as a set of points $Z_0[F] = \{Z_0[p] | p \in F\}$. We assume that $|Z_0[F]| = n$. In the following, as long as it is clear from the context, we identify $p \in F$ with $Z_0[p]$ and write, for example, "F is given to r_i" instead of "$Z_0[F]$ is given to r_i." It is enough emphasizing that F is not given to a robot in terms of its local coordinate system.

Let $\mathbb{T}$ be a set of all coordinate systems, which can be identified with the set of all transformations, rotations, uniform scalings, and their combinations. Let $\mathcal{P}_n$ be the set of all patterns of n points. For any $P, P' \in \mathcal{P}_n$, P is *similar* to P', if there exists $Z \in \mathbb{T}$ such that $Z[P] = P'$, denoted by $P \simeq P'$.

We say that algorithm ψ forms pattern $F \in \mathcal{P}_n$ from an initial configuration I, if for any execution $P(0)(= I), P(1), P(2), \ldots$, there exists a time instance t such that $P(t') \simeq F$ for all $t' \geq t$.

For any $P \in \mathcal{P}_n$, let $C(P)$ be the smallest enclosing circle of P, and $c(P)$ be the center of $C(P)$. Formally, the *symmetricity* $\rho(P)$ of P is defined by

$$\rho(P) = \begin{cases} 1 & \text{if } c(P) \in P, \\ |\{Z \in \mathbb{T} : P = Z[P]\}| & \text{otherwise.} \end{cases}$$

We can also define $\rho(P)$ in the following way [6]: P can be divided into regular k-gons centered at $c(P)$, and $\rho(P)$ is the maximum of such k. Here, any point is a regular 1-gon with an arbitrary center, and any pair of points $\{p, q\}$ is a regular 2-gon with its center $(p + q)/2$.

For any configuration P ($c(P) \notin P$), let $P_1, P_2, \ldots, P_{n/\rho(P)}$ be a decomposition of P into the above mentioned regular $\rho(P)$-gons centered at $c(P)$. Yamashita and Suzuki [7] showed that even when each robot observes P in its local coordinate system, all robots can agree on the order of P_i's such that the distance of the points in P_i from $c(P)$ is no greater than the distance of the points in P_{i+1} from $c(P)$, and each robot is conscious of the group P_i it belongs to. We call the decomposition $P_1, P_2, \ldots, P_{n/\rho(P)}$ ordered by this condition the *regular* $\rho(P)$*-decomposition* of P.

A point on the circumference of $C(P)$ is said to be "on circle $C(P)$" and "the interior of $C(P)$" ("the exterior", respectively) does not include the circumference. We denote the interior (exterior, respectively) of $C(P)$ by $Int(C(P))$ ($Ext(C(P))$). We denote the radius of $C(P)$ by $r(P)$. Given two points p and p' on $C(P)$, we denote the arc from p to p' in the clockwise direction by $arc(p, p')$. When it is clear from the context, we also denote the length of $arc(p, p')$ by $arc(p, p')$. The largest empty circle $L(P)$ of P is the largest circle centered at $c(P)$ such that there is no robot in its interior, hence there is at least one robot on its circumference.

Algorithm with termination agreement. A robot is *static* when it is not in a Move phase, i.e., in a Look phase or a Compute phase, or not executing a cycle. A configuration is *static* if all robots are static. Because robots in the ASYNC model cannot recognize static configurations, we further define stationary configurations. A configuration P is *stationary* for an algorithm ψ, if in any execution starting from P, configuration does not change.

We say algorithm ψ guarantees termination agreement if in any execution $P(0), P(1), \ldots$ of ψ, there exists a time instance t such that $P(t)$ is a stationary configuration, in $P(t')$ ($t' \geq t$), ψ outputs $\emptyset$ at any robot, and all robots know the fact. Specifically, $\psi(Z'[P(t')]) = \emptyset$ in any local coordinate system Z'. This property is useful when we compose multiple algorithms to complete a task.

3. Randomized pattern formation algorithm

The idea of the proposed universal pattern formation algorithm is to translate a given initial configuration I with $\rho(I) > 1$ into a configuration I' with $\rho(I') = 1$ with probability 1, and after that the robots start the execution of a pattern formation algorithm. We formally define the problem.

Definition 1 *The symmetricity breaking problem* is to change a given initial configuration I into a stationary configuration I' with $\rho(I') = 1$.

In Section 3.1, we present a randomized symmetricity breaking algorithm ψ_{SB} with termination agreement. In the following, we assume $n \geq 5$ and I and F do not contain any multiplicities. Additionally, we assume that for a given initial configuration I, no robot occupies $c(I)$, i.e., $c(I) \cap I = \emptyset$.[*4] Due to the page limitation, we omit the pseudo code of ψ_{SB}.

In Section 3.2, we present a randomized universal pattern formation algorithm ψ_{PF}, that uses ψ_{SB} and a pattern formation algorithm ψ_{CWM} [5] with slight modification.

3.1 Randomized symmetricity breaking algorithm ψ_{SB}

In the proposed algorithm ψ_{SB}, robots elect a single leader that occupies a point nearest to the center of the smallest enclosing circle. Clearly, the symmetricity of such configuration is one.

We use a sequence of circles to show the progress of ψ_{SB}. In configuration P, let $C_i(P)$ be the circle centered at $c(P)$ with radius $r(P)/2^i$. Hence, $C_0(P) = C(P)$. We denote the radius of $C_i(P)$ by γ_i. We call the infinite set of circles $C_0(P), C_1(P), \ldots$ the set of *binary circles*. Because ψ_{SB} keeps the smallest enclosing circle of robots unchanged during any execution, we use C_i instead of $C_i(P)$. We call C_i the *front circle* if C_i is the largest binary circle in $L(P)$ including the circumference of $L(P)$, and we call C_{i-1} the *backward circle* (Fig. 1). We denote the number of robots in C_i and on C_i by n_i. Hence, if the current front circle C_i is the largest empty circle, n_i is the number of robots on C_i, otherwise it is smaller than the number of robots on C_i.

Recall that all local coordinate systems are right handed. Hence, all robots agree on the clockwise direction on each binary circle. For C_i ($i \geq 0$) and a robot r on C_i, we call the next

[*4] If there is a robot on $c(I)$, we move the robot by some small distance from $c(I)$ to satisfy the conditions of the terminal configuration of ψ_{SB}.

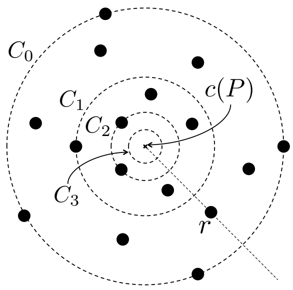
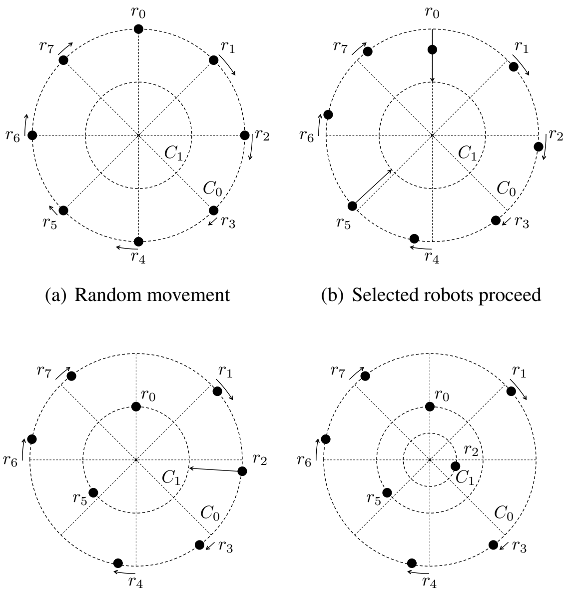


Fig. 1 The set of binary circles and radial track of r, where C_0 is the smallest enclosing circle, C_1 is the backward circle, and C_2 is the front circle.



(a) Random movement (b) Selected robots proceed

(c) Following robot proceeds (d) Single leader is selected

Fig. 2 Random selection

robot on C_i in its clockwise direction *predecessor*, denoted by $pre(r)$, and the one in the counter-clockwise direction *successor*, denoted by $suc(r)$. When there are only two robots r and r' on C_i, $pre(r) = suc(r) = r'$. We say r is neighboring to r' if $r' = pre(r)$ or $r' = suc(r)$. For example, in Fig. 2(a), $pre(r_0)$ is r_1, $suc(r_0)$ is r_7, and r_1 and r_7 are neighbors of r_0.

During an execution of the proposed algorithm, robot r moves to an inner binary circle along a half-line starting from the center of the smallest enclosing circle and passing r's current position. We call this half-line the *radial track* of r (Fig. 1). When r moves from a point on C_i to C_{i+1} along its radial track, we say *r proceeds to C_{i+1}*.

Algorithm ψ_{SB} first sends each robot to its inner nearest binary circle along its radial track if the robot is not on any binary circle. Hence, the current front circle is also the largest empty circle.

Then, ψ_{SB} probabilistically selects at least one robot on the current front circle C_i, and make them proceed to C_{i+1}. These selected robots repeat the selection on C_{i+1}. By repeating this, the number of robots on a current front circle reaches 1 with probability 1. The single robot on the front circle is called the *leader*.

We will show the detailed selection procedure on each front circle. We have two cases depending on the positions of robots when the selection of a front circle C_i starts. One is the *regular polygon case* where robots on C_i form a regular n_i-gon, and the other is the *non-regular polygon case* where n_i robots on C_i form a non-regular polygon.

Selection in the regular polygon case. When robots on the current front circle C_i form a regular n_i-gon (i.e., for all robot r on C_i, $arc(suc(r), r) = 2\pi\gamma_i/n_i$), it is difficult to select some of the robots. Especially, when the symmetricity of the current configuration is n_i, it is impossible to deterministically select some of the robots. In a regular n_i-gon case, ψ_{SB} makes these robots randomly circulate on C_i. Then, a robot that do not catch up with its predecessor and caught by its successor is selected and proceeds to C_{i+1}.

First, if robot r on C_i finds that the robots on C_i form a regular n_i-gon, r randomly selects "stop" or "move." If it selects "move," it generates a random number v in (0..1], and moves $v(1/4)(2\pi\gamma_i/n_i)$ along C_i in the clockwise direction (Fig. 2(a)). This procedure ensures that the regular n_i-gon is broken with probability 1. When r finds that the regular n_i-gon is broken, r stops.

Uniform moving direction ensures the following invariants:

(1) Once r finds that it is caught by $suc(r)$, i.e., the following inequality holds, r never leave from $suc(r)$.

$$Caught(r) = arc(suc(r), r) \leq 2\pi\gamma_i/n_i$$

(2) Once r finds that it missed $pre(r)$, i.e., the following inequality holds, r never catch up with $pre(r)$.

$$Missing(r) = 2\pi\gamma_i/n_i < arc(r, pre(r)) \leq (5/4)(2\pi\gamma_i/n_i)$$

We say robot r is *selected* if it finds that the following predicate holds.

$$Selected(r) = Caught(r) \wedge Missing(r)$$

Then, a selected robot proceeds to C_{i+1} (Fig. 2(b)). Since no two neighboring robots satisfy $Selected$ at a same time, while $Selected(r)$ holds at r, $suc(r)$ and $pre(r)$ wait for r to proceed to C_1. Even when $n_i = 2$, when they are not in the symmetric position, just one of the two robots becomes selected. Note that other robots cannot check whether r is selected or not in the ASYNC model because they do not know whether r has observed the configuration and found that $Selected(r)$ holds.

Observation 2 During the above random movement on the current front circle C_i, $(3/4)(2\pi\gamma_i/n_i) \leq arc(r, pre(r)) \leq (5/4)(2\pi\gamma_i/n_i)$ holds at each robot r on C_i. Let $r' = pre(r)$ and $r'' = suc(r)$ for r on C_i. If r becomes selected and proceeds to C_{i+1}, then $arc(suc(r'), r') > (5/4)(2\pi\gamma_i/n_i)$ and $arc(r'', pre(r'')) > (5/4)(2\pi\gamma_i/n_i)$ hold thereafter even when robots move.

After some selected robots proceed to C_{i+1}, other robots might be still moving on C_i and may become selected later. However, in the ASYNC model, no robot can determine which robot is moving on C_i. For the robots on C_{i+1} to ensure that no more robot will join C_{i+1}, ψ_{SB} makes some of the non-selected robots on C_i proceed to C_{i+1}. The robots on C_i are classified into three types, rejected, following, and undefined.

The predecessor and the successor of a selected robot are classified into *rejected*, and each rejected robot stays on C_i. All robots can check whether robot r is rejected or not with the following condition:

$$\begin{aligned} Rejected(r) = \quad & (arc(r, pre(r)) > (5/4)(2\pi\gamma_i/n_i)) \vee \\ & (arc(suc(r), r) > (5/4)(2\pi\gamma_i/n_i)). \end{aligned}$$

Non-rejected robot r becomes *following* if r finds that at least one of the following three conditions hold:

$$\begin{aligned} FollowPre(r) &= \neg Rejected(r) \wedge Rejected(pre(r)) \\ &\quad \wedge Caught(r) \\ FollowSuc(r) &= \neg Rejected(r) \wedge Rejected(suc(r)) \\ &\quad \wedge Missing(r) \\ FollowBoth(r) &= \neg Rejected(r) \wedge Rejected(pre(r)) \\ &\quad \wedge Rejected(suc(r)). \end{aligned}$$

Hence, we have

$$Following(r) = FollowPre(r) \vee FollowSuc(r) \vee FollowBoth(r).$$

Intuitively, the predecessor and the successor of a following robot never become selected nor following. Algorithm ψ_{SB} makes each following robot proceed to C_{i+1} (Fig. 2(c)).

Finally, robots on C_i that are neither selected, rejected nor following are classified into *undefined*.

Note that $Rejected(r)$ implies $\neg Selected(r)$ and $\neg Following(r)$. Additionally, $Selected(r)$ and $Following(r)$ may hold at a same time.

Eventually, all robots on C_i recognize their classification from selected, following, and rejected. We can show that once a robot finds its classification, it never changes. Then, selected robots and following robots leave C_i and only rejected robots remain on C_0. During the random selection phase, n_i does not change since robots moves in $Int(C_i) \cup C_i$. Hence, all robots can check whether a robot r on C_i is rejected or not with $Rejected(r)$, and the robots on C_{i+1} agree that no more robot proceeds to C_{i+1}. These robots start a new (random) selection on C_{i+1}.

Consider the case where $i = 0$. When $n = 5$, the length of the random movement is largest, and each robot circulates at most $\pi/10$. Hence, no two robots form a diameter. Additionally, ψ_{SB} guarantees that no two neighboring robots leave C_0. Hence, ψ_{SB} keeps C_0 during the random selection. In the same way, when $n \geq 5$, the random selection does not change C_0.

Selection for non-regular polygon case. When robots on the current front circle C_i does not form a regular n_i-gon, ψ_{SB} basically follows the random selection. Thus, robots do not circulate on C_i randomly, but check their classification with the three conditions.

Because robots do not form a regular n_i-gon on C_i, there exists a robot r on C_i that satisfies $arc(suc(r), r) < 2\pi\gamma_i/n_i$. However, there exists many positions of n_i robots on C_i where all such robot r are also rejected, i.e., $arc(r, pre(r)) > (5/4)(2\pi\gamma_i/n_i)$, from which no robot becomes selected nor following (Fig. 3).

In this case, we add one more condition $NRSelected(r)$. We say r satisfies $NRSelected(r)$ when r is on the front circle C_i, all robots on C_i do not satisfy $Selected$ nor $Following$, and $arc(r, pre(r)) > (5/4)(2\pi\gamma_i/n_i)$ and $arc(suc(r), r) \leq 2\pi\gamma_i/n_i$ hold. We note that no two neighboring robots satisfies $NRSelected$. Robot r proceeds half way to C_{i+1}, and waits for all robots satisfying $NRSelected$ to proceed.[*5] Robots in between C_i and C_{i+1} can reconstruct the non-regular polygon on C_i with their radial tracks and after all robots satisfied $NRSelected$ leaves C_i, the robots in $Ext(C_{i+1}) \cap Int(C_i)$ proceeds to C_{i+1}. Note that during a random selection, no robot on C_i satisfies $NRSelected$.

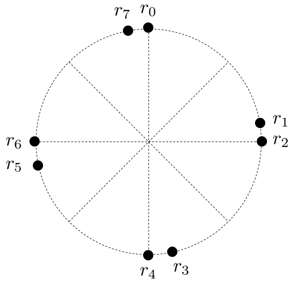
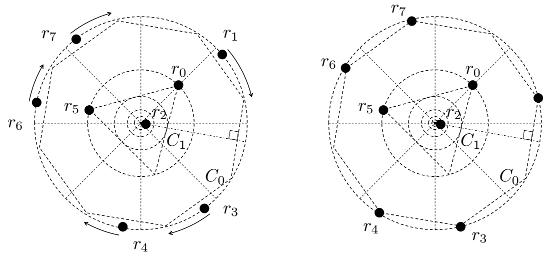


Fig. 3 Non-regular case. All robots are rejected, and no robot proceeds to C_1 with the two conditions $Rejected$ and $Following$.



(a) Embedding on C_0 (b) Terminal configuration

Fig. 4 Stopping rejected robots when the leader is first generated on C_2. (a) The leader embeds a regular octagon on C_0 by its position on C_4. (b) After all robots C_0 have reached the corners of embedded polygons, r_L proceeds to C_5.

We consider one more exception case for initial configurations where robots form a non-regular polygon on C_0. In this case, each robot r first examines $NRSelected(r)$. If proceeding all robots satisfying $NRSelected$ changes C_0, the successor of such robot proceeds to C_1 instead of them. Assume that r is one of such robots satisfying $NRSelected(r)$. Because C_0 is broken after all robots satisfying $NRSelected$ proceeds, in the initial configuration $arc(r, pre(r)) = \pi\gamma_0$. Otherwise, there exists a rejected robot that does not satisfy $NRSelected$ in the initial configuration. Hence, proceeding $suc(r)$ does not change C_0.

After that, robots on C_i determine their classification by using $Rejected$, $Following$, and following robots proceed to C_{i+1}. Eventually all following robots leave C_i, and only rejected robots remain on C_i.

Termination agreement. By repeating the above procedure on each binary circle, with probability 1, only one robot reaches the inner most binary circle, with all other robots rejected (Fig. 2(d)). We say this robot is selected as a single leader. However, rejected robots may be still moving on the binary circles. Thus, the leader robot starts a new phase to stop all rejected robots, so that the

[*5] Otherwise, r cannot distinguish how many robots satisfied $NRSelected$.

terminal configuration is stationary.

Let r_L be the single leader and C_i be the front circle for $R \setminus \{r_L\}$ (this implies the leader is selected during the random selection on C_i). Intuitively, r_L checks the termination of C_{i-j} ($i - j \geq 0$) when r_L is on C_{i+j+2}. Given a current observation, all robots on C_{i-j} are expected to move at most $(1/4)(2\pi\gamma_{i-j}/n_{i-j})$ from corners of some regular n_{i-j}-gon. Hence, there exists an embedding of regular n_{i-j}-gon onto C_{i-j} so that its corners does not overlap these expected tracks. If there is no such embedding, then randomized selection has not been executed on C_{i-j}, and r_L embeds an arbitrary regular n_{i-j}-gon on C_{i-j}. Robot r_L shows the embedding by its position on C_{i+j+2}, i.e., r_L's radial track is the perpendicular bisector of an edge of the regular n_{i-j}-gon (Fig. 4(a)).

Then, ψ_{SB} makes robots on C_{i-j} occupy distinct corners of the regular n_{i-j}-gon. The target points of these robots are determined by the clockwise matching algorithm [4]. We restrict the matching edges before we compute the clockwise matching. Specifically, we use arcs on C_{i-j} instead of direct edges, and direction of each matching edge (from a robot to its destination position) is always in the clockwise direction. Note that under this restriction, the clockwise matching algorithm works correctly on C_{i-j}.[*6] The robots on C_{i-j} has to start a new movement with fixed target positions. Because robots can agree the clockwise matching irrespective of their local coordinate systems, r_L can check whether robots on C_{i-j} finish the random movement.

Then, r_L calculates its next position on C_{i+j+3} in the same way for robots on C_{i-j-1}, and moves to that point.

The leader finishes checking all binary circles on C_{2i+2}, then it proceeds to C_{2i+3} to show the termination of ψ_{SB} (See Fig. 4(b)). However, ψ_{SB} carefully moves robots on C_0 to keep the smallest enclosing circle. When there are just two robots on C_0, then the random selection has not been executed on C_0, and r_L does not check the embedding. When there are more than three robots, there is at least one robot that can move toward its destination with keeping the smallest enclosing circle, and ψ_{SB} first moves such a robot.

For any configuration P satisfying the following two conditions, ψ_{SB} outputs $\emptyset$ at any robot irrespective of its local coordinate system. Hence, such configuration P is a stationary configuration of ψ_{SB}.

(1) P contains a single leader on the front circle, denoted by C_b.
(2) All other robots are in $Ext(C_k) \cup C_k$, satisfying $b \geq 2k + 3$.

Clearly, ψ_{SB} guarantees terminal agreement among all robots.

Algorithm ψ_{SB} guarantees the reachability to a terminal configuration with probability 1, and the terminal configuration is deterministically checkable by any robots in its local coordinate system.

*6 Algorithm ψ_{CWM} [4] reconstructs a clockwise matching from all minimum weight perfect matchings between robots and target points, i.e., for a set of overlapping edges, ψ_{CWM} selects some of them in a "clockwise" manner. The critical assumption is that the number of robots is equal to the number of target points. When ψ_{SB} uses ψ_{CWM}, it restricts the direction of edges when considering the set of all minimum weight matchings. Because the number of target points is larger than the number of robots, without this restriction, a robot in the middle point of two target points may increase the number of target points.

3.2 Randomized pattern formation algorithm ψ_{PF}

We present a randomized pattern formation algorithm ψ_{PF}. Algorithm ψ_{PF} executes ψ_{SB} when the configuration does not satisfy the two conditions of the terminal configuration of ψ_{SB}. When the current configuration satisfies the two terminal conditions of ψ_{SB}, ψ_{PF} starts a pattern formation phase.

Fujinaga et al. proposed a pattern formation algorithm ψ_{CWM} in the ASYNC model, which uses the clockwise minimum weight perfect matching between the robots and an embedded target pattern [5]. The embedding of the target pattern is determined by the robots on the largest empty circle. Additionally, when there is a single robot on the largest empty circle, ψ_{CWM} keeps this robot the nearest robot to the center of the smallest enclosing circle during any execution. We use this property to separate the configurations that appears executions of ψ_{SB} and those of ψ_{CWM}.

Algorithm ψ_{PF} uses ψ_{CWM} after ψ_{SB} terminates, however, to compose ψ_{SB} and ψ_{CWM}, we modify the terminal configuration of ψ_{SB} to keep the leader showing the termination of ψ_{SB}. Let P be a given terminal configuration of ψ_{SB}, and the single leader be r_L on the front circle C_L. Given a target pattern F, let $F_1, F_2, \ldots, F_{n/\rho(F)}$ be the regular $\rho(F)$-decomposition of F. Then, ψ_{CWM} embeds F so that $f \in F_1$ lies on the radial track of r_L, and $r(F) = r(P)$. When $c(F) \in F$, ψ_{CWM} also perturbs this target point. Let F' be this embedding.

Then, ψ_{PF} first moves r_L as follows: Let $L(F')$ be the largest empty circle of F' and $\ell(F')$ be its radius. Let k ($k > 0$) be an integer such that C_k be the largest binary circle in $L(F')$. If C_{2k+3} is in C_L, r_L proceeds to C_{2k+3}. When C_{2k+3} is in $Ext(C_L)$, r_L does not move. Then, ψ_{PF} starts the execution of ψ_{CWM}. After $R \setminus \{r_L\}$ reach their destination positions, r_L moves to its target point along its radial track.

4. Correctness

We will show a sketch of the proof of ψ_{PF}. Let I be an initial configuration where robots form a regular n-gon. We first show that ψ_{SB} randomly selects at least one and at most $n/2$ robots and C_0 does not change by robots' random movement on C_0. Intuitively, the adversary has no choice to let the robots move with keeping the regular n-gon. However, the proposed algorithm outputs a moving distance smaller than the minimum moving distance δ, and the adversary cannot stop other robots. Consequently, because of the strict progress property, the robots then observe a non-regular configuration.

Lemma 3 Starting from an initial configuration I where the robots form a regular n-gon, with probability 1, any execution of ψ_{SB} in the ASYNC model reaches a configuration where at least one robot is selected, and until then ψ_{SB} does not change the smallest enclosing circle of robots.

A selected robot r proceeds to C_1 and while $Selected(r)$ holds, $Selected$ and $Following$ do not hold at its neighbors, and the neighbors become rejected after r proceeds. We have the same property for any following robot. Eventually, all robots recognize their classification and selected and following robots reach C_1. No two neighboring robots in $P(0)$ enters the interior of C_0 in the randomized selection on C_0. Hence, the smallest enclosing circle does not change during any execution of ψ_{SB} when $n \geq 5$. Consequently, with probability 1, the system reaches a configuration where C_0 contains only rejected robots.

The rejected robots on C_0 do not become selected nor following even when robots on C_1 moves, because n_0 does not change and all robots can check their states with the predicate $Rejected$. Hence, robots on C_1 start a new random selection phase. We obtained the base case. Clearly, we can apply above results to robots on any front circle. The system reaches a configuration where only one robot is on the front circle, and all robots in the backward circle are rejected.

Then, ψ_{SB} makes the leader check whether the robots on each binary circle C_i have stopped by embedding a regular n_i-gon so that robots on C_i starts a new deterministic movement to reach the corners of the regular n_i-gon. The system eventually reaches a terminal configuration of ψ_{SB} with probability 1.

From a static terminal configuration of ψ_{SB}, robots execute ψ_{CWM}, and we have the following theorem.

Theorem 4 Algorithm ψ_{PF} forms any target pattern from any initial configuration with probability 1.

5. Conclusion

We present a randomized pattern formation algorithm for oblivious robots in the ASYNC model. The proposed algorithm consists of a randomized symmetricity breaking algorithm and a pattern formation algorithm proposed by Fujinaga et al. [5]. One of our future directions is to extend our results to the robots with limited visibility, where oblivious robots easily increase the symmetricity [8].

References

[1] M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro, Distributed computing by mobile robots: gathering, *SIAM J. of Comput.,* **41**, 4, pp.829–879, 2012.
[2] Y. Dieudonné, F. Petit, and V. Villain, Leader election problem versus pattern formation problem. *Proc. of DISC 2010*, pp.267–281 (2010).
[3] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer, Arbitrary pattern formation by asynchronous, anonymous, oblivious robots, *Theor. Comput. Sci.,* **407**, pp.412–447 (2008).
[4] N. Fujinaga, H. Ono, S. Kijima, and M. Yamashita, Pattern formation through optimum matching by oblivious CORDA robots, *Proc. of OPODIS 2010*, pp.1–15 (2010).
[5] N. Fujinaga, Y. Yamauchi, S. Kijima, and M. Yamashita, Asynchronous pattern formation by anonymous oblivious mobile robots, *Proc. of DISC 2012*, pp.312–325 (2012).
[6] I. Suzuki, and M. Yamashita, Distributed anonymous mobile robots: Formation of geometric patterns, *SIAM J. on Comput.*, **28**(4), pp.1347–1363 (1999).
[7] M. Yamashita, and I. Suzuki, Characterizing geometric patterns formable by oblivious anonymous mobile robots, *Theor. Comput. Sci*, **411**, pp.2433–2453 (2010).
[8] Y. Yamauchi, and M. Yamashita, Pattern formation by mobile robots with limited visibility, *Proc. of SIROCCO 2013*, pp.201-212, (2013).

© 2015 Information Processing Society of Japan