Theoretical Computer Science 568 (2015) 84–96

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

On the self-stabilization of mobile oblivious robots in uniform rings [☆]

Fukuhito Ooshita [a,*], Sébastien Tixeuil [b,c,d]

[a] Graduate School of Information Science and Technology, Osaka University, 1-5 Yamadaoka, Suita, Osaka 565-0871, Japan
[b] Sorbonne Universités, UPMC Univ. Paris 06, UMR 7606, LIP6, F-75005, Paris, France
[c] CNRS, UMR 7606, LIP6, F-75005, Paris, France
[d] Institut Universitaire de France, France

ARTICLE INFO

Article history:
Received 9 December 2013
Received in revised form 6 December 2014
Accepted 15 December 2014
Available online 23 December 2014
Communicated by D. Peleg

Keywords:
Mobile robots
Rings
Gathering
Orientation
Set formation

ABSTRACT

We investigate self-stabilizing algorithms for anonymous and oblivious robots in uniform ring networks, that is, we focus on algorithms that can start from any initial configuration (including those with multiplicity points). First, we show that no probabilistic self-stabilizing gathering algorithm exists in the asynchronous (ASYNC) model or if only global-weak and local-strong multiplicity detection is available. This impossibility result implies that a common assumption about initial configurations (no two robots share a node initially) is a very strong one.

On the positive side, we give a probabilistic self-stabilizing algorithm for the gathering and orientation problems in the semi-synchronous (SSYNC) model with global-strong multiplicity detection. With respect to impossibility results, those are the weakest system hypotheses. In addition, as an application of the previous algorithm, we provide a self-stabilizing algorithm for the set formation problem. Our results imply that any static set formation can be realized in a self-stabilizing manner in this model.

1. Introduction

Background and motivation Studies for mobile robot networks have emerged recently in the field of distributed computing. Their goal is to achieve some tasks by a team of mobile robots with weak capabilities. Most studies assume that robots are identical (robots execute the same algorithm and cannot be distinguished by their appearance) and oblivious (robots have no memory and cannot remember the history of their execution). In addition, it is assumed that robots cannot communicate with other robots directly. The communication among robots is done in an implicit way having each robot observe the positions of others.

Since Suzuki and Yamashita presented a pioneering work [1], many results about such robots have been published. In this paper, we focus on unoriented anonymous ring networks since algorithms for ring networks give solutions for the essential difficulties that arise in robot networks such as symmetry breaking. The main interest of previous works on ring networks is to characterize the minimum assumptions that allow deterministic algorithms. Many algorithms for fundamental problems

☆ A preliminary extended abstract of this paper appears in the proceedings of the 14th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS2012).
* Corresponding author. Tel.: +81 6 68794117, fax: +81 6 68794119.
E-mail addresses: f-oosita@ist.osaka-u.ac.jp (F. Ooshita), Sebastien.Tixeuil@lip6.fr (S. Tixeuil).

http://dx.doi.org/10.1016/j.tcs.2014.12.008
0304-3975/© 2014 Elsevier B.V. All rights reserved.

such as the gathering problem, which requires all robots to gather at a non-predetermined single node, have been proposed on various weak assumptions.

Most previous works in the discrete model (*i.e.*, the graph model) for robots make the assumption that initial robot positions are *unique*, that is, in the initial configuration, no two robots share the same node. Still, it is generally accepted that robot algorithms, due to the obliviousness of the robots, are "almost" self-stabilizing, that is, they can recover from an arbitrary initial global state. Characterizing what "almost" means in this context is the topic of this paper, and our goal is to clarify the set of problems that can be solved in a self-stabilizing setting, considering classical hypotheses in robot networks (deterministic *vs.* probabilistic, asynchronous *vs.* semi-synchronous model, global *vs.* local multiplicity detection, strong *vs.* weak multiplicity detection, etc.). Obviously, in an arbitrary initial configuration where occupied nodes host the same number of robots and are symmetric, no deterministic protocol can break the symmetry, which is needed for solving *e.g.* gathering or orienting a ring. Hence, only probabilistic algorithms may be self-stabilizing.

Our contribution In the first part of this paper, we investigate the difficulty of probabilistic self-stabilizing algorithms using weak assumptions. In more details, we show that no probabilistic self-stabilizing algorithm exists that achieves gathering in the asynchronous (ASYNC) model or if only local-strong and global-weak multiplicity detection is available. This impossibility means that the assumption about initial configurations made in previous works is very strong: removing it requires many additional assumptions. Simply put, previous robot algorithms on graphs are *not* "almost" self-stabilizing in the sense that initial configurations are essential to enable the existence of such algorithms under weak assumptions. Interestingly, the impossibility results hold even if we assume rings are oriented (or all robots have the same chirality). That is, the assumption of orientation is not strong enough to allow self-stabilization though the assumption is helpful for many problems.

In the second part, we investigate which problems can be solved in a self-stabilizing manner, and we focus on the gathering and orientation problems. Those two problems are essential to solve most tasks in mobile robot networks. In fact, the difficulty of most problems comes from the lack of agreement on robots' views, *i.e.*, no origin and no orientation in the network. For example, consider the problem that deploys a minimal independent set (MIS) of robots in the network. It is easy for robots to recognize the same form of a MIS from the number of nodes in the ring network. If the network has some origin and orientation, robots can easily recognize the corresponding nodes that construct a MIS. However, if the network has no origin or no orientation, robots cannot uniquely recognize the same nodes due to symmetry of the network configuration even though they know the exact shape of a MIS for a particular number of robots and ring size. Construction of a global origin and orientation is realized by the gathering and orientation problem.

We give probabilistic self-stabilizing algorithms for the gathering and orientation problems on unoriented anonymous rings. We assume the weakest possible model with respect to impossibility results: we assume the semi-synchronous (SSYNC) model and global-strong multiplicity detection. First, we give a self-stabilizing gathering algorithm that achieves gathering in $O(n \log k)$ expected asynchronous rounds and $O(kn)$ expected moves, where k is the number of robots and n is the number of nodes. Since the gathering requires $\Omega(kn)$ moves from initial configurations where robots are evenly scattered, the proposed algorithm is asymptotically optimal for the number of moves. Second, we give a self-stabilizing orientation algorithm using the gathering algorithm. This algorithm not only provides an orientation of the ring but also extracts ℓ robots from the robot pool created by the gathering algorithm, where ℓ is the number of robots required to solve an application problem. This algorithm works if $k \geq \ell + 2$, and requires $O((\log k + \ell)n)$ expected asynchronous rounds and $O(kn)$ expected moves. Finally, as an application of the proposed algorithms, we provide a self-stabilizing algorithm for the set formation problem. The set formation problem can form any static set such as a uniform distribution and a MIS. For the set of size s such that $s \leq k - 1$, this algorithm requires $O((\log k + s)n)$ expected asynchronous rounds and $O(kn)$ expected moves.

Related works Many results about a network of mobile robots have been published since Suzuki and Yamashita presented the pioneering work [1]. They formalize a network of mobile robots in two-dimensional Euclidean space, which is known as the continuous model. They give possibility and impossibility results of the gathering and convergence problem, and characterize the class of geometric patterns that robots can form. Note that they prove that two robots cannot achieve gathering deterministically in the model. Consequently any deterministic gathering algorithm assumes some conditions on the number of robots or the initial positions. Dieudonné and Petit [2] show that, with global-strong multiplicity detection, a deterministic self-stabilizing gathering algorithm exists in the continuous model if and only if the number of robots is odd. Probabilistic self-stabilizing gathering algorithms in the continuous model are proposed in [3,4]. Recently, algorithms for robots with other assumptions such as limited visibility [5], fat robots [6], fat robots with slim cameras [7], and visible lights [8] have been considered in the continuous model.

On the other hand, there are many algorithms for mobile robots in the discrete model (*i.e.*, the graph model). For grid networks, D'Angelo et al. [9] propose deterministic gathering algorithms in the asynchronous model without multiplicity detection. For ring networks, Klasing et al. [10,11] propose deterministic gathering algorithms in the asynchronous model with global-weak multiplicity detection. They also show that there exist some initial configurations where any deterministic algorithm cannot achieve gathering. D'Angelo et al. propose a single algorithm that achieves gathering for any solvable initial configuration in the model [12]. Izumi et al. [13] provide a deterministic gathering algorithm with local-weak multiplicity detection. The algorithm assumes that initial configurations are non-symmetric and non-periodic, and the number of robots is less than half number of nodes. For odd number of robots or odd number of nodes in the same model, Kamei et al. [14,15]

propose the gathering algorithm that also works in symmetric configurations. Note that all of the above works assume some initial configurations and thus they are not self-stabilizing.

The exploration algorithm for rings is considered in [16–18]. Traditionally the exploration algorithm requires termination after the exploration. This implies that robots have to distinguish the terminal configurations from other configurations. Therefore it is impossible to construct a self-stabilizing algorithm for the exploration with termination because the initial configuration can be a terminal configuration.

Many algorithms for pattern formation problems in the continuous model are proposed, however algorithms for set formation problems in rings are scarcely proposed. To the best of our knowledge, the work in [19] is the only one that considers the uniform distribution on a ring. The work proposes deterministic algorithms in a weaker model but they also assume that each node is occupied by at most one robot in the initial configuration.

2. Model

System models The system consists of n nodes and k mobile robots. The nodes $v_0, v_1, \dots, v_{n-1}$ construct an undirected ring in this order. We consider two types of rings: oriented and unoriented rings. For simplicity we consider mathematical operations to indices of nodes as operations modulo n. Neither nodes nor links have any identifiers and labels, and consequently robots cannot distinguish nodes and links. Robots occupy some nodes of the ring.

Robots considered here have the following characteristics and capabilities. Robots are *identical*, that is, robots execute the same algorithm and cannot be distinguished by their appearance. Robots are *oblivious*, that is, robots have no memory and cannot remember the history of its execution. Robots cannot communicate with other robots directly, however they can observe the positions of other robots. This means that robots can communicate implicitly by their positions. We assume each robot has some multiplicity detection. We consider two types of multiplicity detection: global-strong multiplicity detection, and local-strong and global-weak multiplicity detection. When each robot has *global-strong multiplicity detection*, each robot can detect the number of robots on each node. When each robot has *local-strong and global-weak multiplicity detection*, each robot can detect the number of robots only on its current node and detect whether the number of robots is one or more than one for every node.

Each robot executes the algorithm by repeating cycles. At the beginning of each cycle, the robot observes the environment and the positions of other robots (look phase). According to the observation, the robot computes whether it moves to its adjacent node or stays idle (compute phase). If the robot decides to move, it moves to the node by the end of the cycle (move phase). To analyze the asynchronous behavior of robots, we introduce the notion of *scheduler* that decides when each robot executes phases. When the scheduler makes robot r execute some phase, we say the scheduler activates the phase of r or simply activates r. We consider two types of synchronicity: the SSYNC (semi-synchronous) model and the ASYNC (asynchronous) model. In the *SSYNC model* (sometimes called SYm or ATOM), a set of robots is selected by the scheduler, and cycles of the selected robots are executed synchronously. In the *ASYNC model* (sometimes called CORDA), cycles of robots are executed asynchronously. Note that in the ASYNC model each robot can move based on the outdated view that the robot observed before. On the other hand, robots can move based on the latest view in the SSYNC model. For both models, the scheduler is *fair*, which guarantees that each robot is activated infinitely often. When we analyze the worst-case performance of algorithms, we consider the scheduler as an adversary. That is, we assume that the scheduler knows all information, such as positions and decisions of all robots, and activates cycles to degrade the performance of algorithms as much as possible.

A *configuration* of the system is defined as the number of robots on each node. If a node is occupied by some robots, the node is called a *robot node*. If a node is occupied by exactly m robots, the node is called a *m-robot node*. When $m \geq 2$ holds, an m-robot node is also called a *tower node*. If a node is occupied by no robots, the node is called a *free node*. In addition, we define a *1-robot block* as a maximal set of consecutive 1-robot nodes. We define the size of a 1-robot block as the number of nodes in the 1-robot block. A node forming a 1-robot block of size 1 is also called an *isolated* 1-robot node. Assuming that node v_i $(0 \leq i \leq n-1)$ is occupied by m_i robots at configuration C, if there exists an index x such that $m_{x+i} = m_{x-i}$ holds for any i or $m_{x+i} = m_{x-(i+1)}$ holds for any i (*i.e.*, there exists an axis of symmetry of the ring), configuration C is called *symmetric*.

When a robot observes the environment, it gets a *view* of the system. Consider a configuration such that nodes $v_{i_0}, v_{i_1}, \dots, v_{i_{w-1}}$ $(i_0 < i_1 < \cdots < i_{w-1})$ are robot nodes and each robot node v_{i_x} is occupied by m_{i_x} robots. A robot obtains two views: a forward view and a backward view. When we assume global-strong multiplicity detection, the forward view and the backward view of a robot on node v_{i_x} are defined as $V_f = (M_0, D_0, M_1, D_1, \dots, M_{w-1}, D_{w-1})$ and $V_b = (M_0, D_{w-1}, M_{w-1}, \dots, M_1, D_0)$ respectively, where M_y is the number of robots on its y-th robot node in the forward direction (*i.e.*, $M_y = m_{i_{(x+y) \bmod w}}$) and D_y is the distance from its y-th node to $(y+1)$-th robot node (*i.e.*, $D_y = (i_{(x+y+1) \bmod w} - i_{(x+y) \bmod w}) \bmod n$). When we assume local-strong and global-weak multiplicity detection, the views are defined similarly except that M_y $(0 < y < w)$ is one or two: $M_y = 1$ implies $m_{i_{(x+y) \bmod w}} = 1$ and $M_y = 2$ implies $m_{i_{(x+y) \bmod w}} > 1$ (note that $M_0 = m_{i_x}$). When we assume unoriented rings (or robots have no chirality), each robot cannot distinguish the forward view and the backward view, and thus it uses the lexicographically bigger one as its view. Fig. 1 shows an example of configurations. If we assume global-strong multiplicity detection, robot r on v_0 obtains two views $(4, 2, 1, 1, 3, 2, 4, 3)$ and $(4, 3, 4, 2, 3, 1, 1, 2)$ and uses $(4, 3, 4, 2, 3, 1, 1, 2)$ as its view. If we assume local-strong and global-weak multiplicity detection, r uses $(4, 3, 2, 2, 2, 1, 1, 2)$ as its view. For node v_{i_x} on unoriented rings, if the forward view

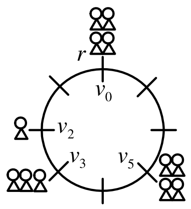

Fig. 1. An example of configurations.

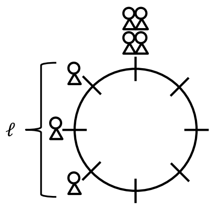

Fig. 2. The goal of the orientation problem.

and the backward view are identical, we say that the view on v_{i_x} is symmetric. In this case robots on v_{i_x} cannot distinguish between the two directions when they move. In such cases, we assume the scheduler decides which direction each robot moves to.

We evaluate the algorithm by the number of expected asynchronous rounds and expected total number of moves. A *round* is defined as the shortest fragment of an execution where each robot executes at least one complete cycle. The total number of moves is the sum of moves each agent makes.

The problems to be solved We give the definition of the problems to be solved in this paper. First, we define the gathering problem, whose goal is to collect all robots at a single node.

Definition 1. An algorithm A solves the gathering problem if and only if the system reaches the configuration where all robots stay at a single node and do not move.

Next, we define the orientation problem, which aims to establish an orientation in an unoriented ring. In this paper, we construct a configuration such that applications can easily use the orientation. To be concrete, we construct a configuration such that there exist exactly one tower node and exactly one 1-robot block of size ℓ neighboring the tower node, where ℓ is the given input for the application (see Fig. 2). As we show later, the configuration is useful because each robot in the 1-robot block can easily recognize its role from the position.

Definition 2. An algorithm A solves the orientation problem (and 1-robot block creation) for given input ℓ if and only if the system reaches the configuration satisfying the following conditions: 1) No robot moves, 2) there exists exactly one tower node, and 3) there exists exactly one 1-robot block of size ℓ that is neighboring to the tower node.

Next, we give the definition of the set formation problem. The aim of the set formation problem is to construct the target set by robot nodes in an unoriented ring. However, since the ring is unoriented and nodes have no identifiers, it is impossible to exactly indicate nodes by the target set. Instead we indicate the target set by the relative set $SET \subset \{0, 1, \ldots, n-1\}$. From the set SET, the family of terminal sets $\mathcal{F}(SET)$ is defined as follows:

$$\mathcal{F}(SET) = \bigcup_{0 \le i \le n-1} \left\{ \bigcup_{j \in SET} \{v_{(i+j) \bmod n}\}, \bigcup_{j \in SET} \{v_{(i-j) \bmod n}\} \right\}.$$

For example, to realize a uniform distribution for 12-node rings, we can define $SET = \{0, 3, 6, 9\}$. Then, node sets such as $\{v_0, v_3, v_6, v_9\}$ and $\{v_1, v_4, v_7, v_{10}\}$ are included in $\mathcal{F}(SET)$. We define the set formation problem as follows.

Definition 3. An algorithm A solves the set formation problem for a given set $SET \subset \{0, 1, \ldots, n-1\}$ if and only if the system reaches the configuration satisfying the following conditions: 1) No robot moves, and 2) letting S_t be the set of robot nodes, $S_t \in \mathcal{F}(SET)$ holds.

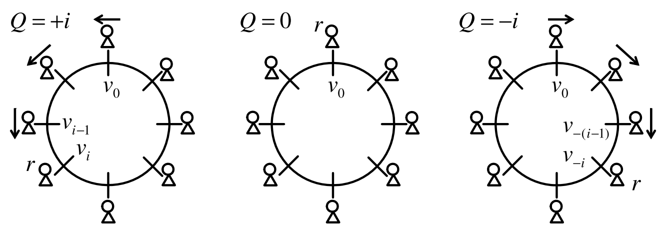

Fig. 3. The configuration $Q = +i$, $Q = 0$, and $Q = -i$. An agent with an arrow decides to move in the direction, but the move phase is not activated.

Finally, we give the definition of probabilistic self-stabilizing algorithms.

Definition 4. An algorithm A is a probabilistic self-stabilizing algorithm for problem $\mathcal{P}$ if and only if, for some function $p : \mathbb{N} \times \mathbb{N} \to (0, 1]$, algorithm A solves $\mathcal{P}$ with probability at least $p(k, n)$ from any initial configuration for n-node and k-robot rings.

3. Impossibility of probabilistic self-stabilizing gathering algorithms on weak models

In this section, we show the impossibility of probabilistic self-stabilizing gathering algorithms on weak models. As we described in Section 1, we assume the SSYNC model and global-strong multiplicity detection to propose algorithms. In this section we show that such strong assumptions are necessary to realize a self-stabilizing gathering algorithm.

First we show the impossibility on a weaker synchronous model, *i.e.*, the ASYNC model. Recall that cycles of robots are executed asynchronously in the ASYNC model. Hence, it is possible that, after robot r observes the configuration and decides to move, some robots repeatedly observe and move before r actually moves. We show the impossibility on the ASYNC model in oriented rings. Since the assumption of oriented rings is stronger than that of unoriented rings, it is clear that the impossibility holds in unoriented rings.

Theorem 1. *No probabilistic self-stabilizing gathering algorithm with global-strong multiplicity detection exists in the ASYNC model even if rings are oriented.*

Proof. Assume that a probabilistic self-stabilizing gathering algorithm A exists that achieves gathering with probability at least $p(k, n)$ for some function p. Consider an n-node oriented ring and $k = n$ robots $r_0, r_1, \dots, r_{k-1}$ such that each node is occupied by one robot in the initial configuration. Note that each robot has the same view in the initial configuration. We consider the adversarial behavior of the scheduler from this initial configuration. First, we assume the procedure of the scheduler *Proc*(X), where X is a positive integer parameter, that applies to this initial configuration. This procedure satisfies the following: 1) *Proc*(X) activates each robot at least once, 2) *Proc*(X) is completed in finite time, and 3) letting $P(X)$ be the probability that the configuration does not change due to the execution of *Proc*(X), we have $\lim_{X\to\infty} P(X) = 1$. If the configuration is not changed, the scheduler can repeatedly execute *Proc*(X). We show the existence of this procedure later.

Let p be a small constant satisfying $p < p(k, n)$. Let X_j $(j = 1, 2, \dots)$ be a positive integer satisfying $P(X_j) > 1 - p/2^j$. We consider the scheduler such that it executes procedures *Proc*(X_1), *Proc*(X_2), ... in this order. Note that this scheduler is fair because each robot is activated infinitely often. To achieve gathering, the configuration should be changed after *Proc*(X_j) for some j. Thus, letting P^* be the probability that A achieves gathering under this scheduler, we have $P^* < (1 - P(X_1)) + (1 - P(X_2)) + (1 - P(X_3)) + \cdots < p$. This is a contradiction.

In the rest of the proof, we construct the above procedure *Proc*(X). When each node is occupied by one robot, every robot decides on the movement with the same probability. We assume a robot on each node v_i moves forward (*i.e.*, to v_{i+1}) with probability p_1, moves backward (*i.e.*, to v_{i-1}) with probability p_2, and stays with probability $1 - p_1 - p_2$. Note that, since rings are oriented, each robot recognizes the common direction. Clearly $p_1 + p_2 > 0$ holds, otherwise robots cannot achieve gathering because they do not move from the initial configuration.

Intuitively *Proc*(X) activates robots one by one and makes them decide to move in the same direction eventually. For each activation, the scheduler cannot decide on the direction of a robot because algorithm A is nondeterministic. However, we can show that the scheduler makes robots decide to move in the same direction with high probability by repeating activations. After all the robots decide to move in the same direction, the scheduler activates the move phases of all robots at the same time. Then, since all robots move in the same direction at the same time, the configuration is not changed.

To describe the behavior, we define Q as the variable that indicates a configuration (see Fig. 3). The value $Q = 0$ indicates the configuration such that no robot decides to move. In the initial configuration, $Q = 0$ holds. The value $Q = +i$ $(0 < i \le n)$ indicates the configuration such that robots on $v_0, v_1, \dots, v_{i-1}$ decide to move forward. The value $Q = -i$ $(0 < i \le n)$ indicates the configuration such that robots on $v_0, v_{-1}, \dots, v_{-(i-1)}$ decide to move backward. Note that $Q = +n$ or $Q = -n$ implies that all robots decide to move in the same direction.

In procedure *Proc*(X), the scheduler first repeats the following steps:

- If $Q = 0$, the scheduler activates look and compute phases of robot r on v_0.
  - If r decides to stay, the configuration is still $Q = 0$.
  - If r decides to move forward, the configuration becomes $Q = +1$.
  - If r decides to move backward, the configuration becomes $Q = -1$.
- If $Q = +i$ $(i > 0)$, the scheduler activates look and compute phases of robot r on v_{+i}.
  - If r decides to stay, the configuration is still $Q = +i$.
  - If r decides to move forward, the configuration becomes $Q = +(i+1)$.
  - If r decides to move backward, the scheduler activates move phases of r and a robot on $v_{+(i-1)}$. Then, the configuration becomes $Q = +(i-1)$.
- If $Q = -i$ $(i > 0)$, the scheduler activates look and compute phases of robot r on v_{-i}.
  - If r decides to stay, the configuration is still $Q = -i$.
  - If r decides to move forward, the scheduler activates move phases of r and a robot on $v_{-(i-1)}$. Then, the configuration becomes $Q = -(i-1)$.
  - If r decides to move backward, the configuration becomes $Q = -(i+1)$.

The scheduler stops executing the steps above when the obtained configuration is $Q = +n$ or $Q = -n$. In this case, the scheduler activates move phases of all robots at the end of *Proc*(X). Then, all robots move in the same direction at the same time, and consequently the configuration is not changed. In addition, the steps above are completed if the scheduler takes X steps. In this case, the scheduler arbitrarily activates the remaining phases of all robots to guarantee fairness.

Clearly, *Proc*(X) satisfies the above conditions 1 and 2. From now, we show that *Proc*(X) satisfies the condition 3. Let Q_h $(h > 0)$ be the value of Q after the h-th step and let $Q_0 = 0$. When $-n < Q_h < n$ holds for h $(h \geq 0)$, $Q_{h+1} = Q_h + 1$ holds with probability p_1, $Q_{h+1} = Q_h - 1$ holds with probability p_2, and $Q_{h+1} = Q_h$ holds with probability $1 - p_1 - p_2$. This implies, from any configuration $Q = i$ $(-n < i < n)$, the configuration reaches $Q = +n$ or $Q = -n$ within $2n$ steps with probability at least $p_1^{2n} + p_2^{2n}$. Hence, the probability that the configuration neither reaches $Q = +n$ nor $Q = -n$ within X steps is at most $(1 - p_1^{2n} - p_2^{2n})^{\lfloor X/(2n) \rfloor}$. Consequently, we have $P(X) \geq 1 - (1 - p_1^{2n} - p_2^{2n})^{\lfloor X/(2n) \rfloor}$. This implies $\lim_{X\to\infty} P(X) = 1$. □

Next we consider the impossibility on a model with weaker multiplicity detection, *i.e.*, local-strong and global-weak multiplicity detection. For this case, we have a similar result.

Theorem 2. *No probabilistic self-stabilizing gathering algorithm with local-strong and global-weak multiplicity detection exists in the SSYNC model even if rings are oriented.*

Proof. Assume that a probabilistic self-stabilizing gathering algorithm A exists that achieves gathering with probability $p(k, n)$ for some function p. Consider an n-node oriented ring and $k = 3n$ robots $r_0, r_1, \ldots, r_{k-1}$ such that each node is occupied by three robots in the initial configuration. Similarly to the proof of Theorem 1, we construct the procedure of the scheduler *Proc*(X) that satisfies the three conditions. Then, we can show a contradiction similarly.

In the rest of the proof, we construct the procedure *Proc*(X). In procedure *Proc*(X), the scheduler activates r_0 repeatedly until r_0 moves back to its initial position. Similarly, the scheduler activates $r_1, r_2, \ldots$ until each robot moves back to its initial position. In more details, the scheduler repeats the following steps:

- For each r_i $(i = 0, 1, \ldots, k-1)$, the scheduler activates r_i as follows:
  1. Let v_x be the node that r_i initially stays at. The scheduler activates r_i once.
  2. If r_i does not move, the scheduler finishes the activation of r_i and transits to the activation of the next robot.
  3. If r_i moves to v_{x-1} or v_{x+1}, the scheduler repeatedly activates r_i until r_i moves back to v_x or it repeats the activation X times.

Clearly, *Proc*(X) satisfies the conditions 1 and 2 in the proof of Theorem 1. From now, we show that *Proc*(X) satisfies the condition 3. We analyze the probability $P(X)$ that all robots move back to their initial nodes after *Proc*(X). Consider the activation of r_i $(0 \leq i \leq k-1)$ in the case that $r_0, r_1, \ldots, r_{i-1}$ have moved back to their initial nodes. Let v_x be the initial position of r_i. If r_i does not move in its first activation, r_i still stays at its initial node. In the following, we consider the case that r_i moves in its first activation. Note that, until r_i moves back to v_x, it gets the view where its current node has four robots and other nodes have more than one robot. From the definition of local-strong and global-weak multiplicity detection, r_i decides on the movement with the same probability until it moves back to v_x. We assume r_i moves forward with probability p_1, moves backward with probability p_2, and stays with probability $1 - p_1 - p_2$. Clearly $p_1 + p_2 > 0$ holds, otherwise robots cannot achieve gathering from the configuration where every node is occupied by four robots. This is because, in this configuration, every robot gets the same view as r_i and does not move.

If r_i moves forward (resp., backward) in its initial activation, it stays at v_{x+1} (resp., v_{x-1}). Then, until r_i moves back to $v_x = v_{x+n}$ (resp., $v_x = v_{x-n}$), r_i continues to move forward with probability p_1 and move backward with probability p_2. This implies, from any node, r_i moves back to v_x within n activations with probability at least $p_1^n + p_2^n$. Hence, the probability that r_i does not move back to v_x within X activations is at most $(1 - p_1^n - p_2^n)^{\lfloor X/n \rfloor}$. Consequently, r_i moves back to v_x with probability at least $1 - (1 - p_1^n - p_2^n)^{\lfloor X/n \rfloor}$. If all robots move back to their initial nodes, the configuration is not changed after *Proc*(X). This implies $P(X) \geq (1 - (1 - p_1^n - p_2^n)^{\lfloor X/n \rfloor})^k$, and consequently we have $\lim_{X\to\infty} P(X) = 1$. □

4. An algorithm for the gathering problem

In the rest of the paper, we propose probabilistic self-stabilizing algorithms for the gathering problem, the orientation problem, and the set formation problem. The rings are unoriented, that is, robots have no chirality. All the algorithms assume the SSYNC model and global-strong multiplicity detection. Recall that we consider n-node and k-robot unoriented rings in the following.

In this section, we consider the gathering problem. The goal of the gathering problem is to form the configuration such that there exists exactly one tower node and all robots are on this tower. We denote a set of such configurations by $\mathcal{C}_g$. For any configuration C, we define $M(C)$ as the maximum number of robots that occupy one node.

The behavior of each robot at configuration C ($C \notin \mathcal{C}_g$) is given as follows.

- **Case 1:** There exists exactly one $M(C)$-robot node. Let v_t be the $M(C)$-robot node. Let $N_C(v_i)$ be the number of robots on node v_i at configuration C.
  - For each x ($0 < x \leq \lceil n/2 \rceil$), every robot on node v_{t+x} such that $\sum_{i=1}^{x} N_C(v_{t+i}) < M(C)$ moves to v_{t+x-1}.
  - For each x ($0 < x < n - \lceil n/2 \rceil$), every robot on node v_{t-x} such that $\sum_{i=1}^{x} N_C(v_{t-i}) < M(C)$ moves to v_{t-x+1}.
- **Case 2:** There exists more than one $M(C)$-robot node. For each $M(C)$-robot node v, we define h_v be the distance from v to its closest neighboring robot node. Let $h_{min} = \min\{h_v \mid v \text{ is a } M(C)\text{-robot node}\}$. Let V_1 be the set of robot nodes such that the distance to an $M(C)$-robot node is h_{min}. Let R_1 be the set of robots that occupy nodes in V_1. Note that R_1 is a set of robots that are closest to some other $M(C)$-robot node.
  - (Case 2-1) If $|R_1| = 1$, the robot in R_1 moves toward its closest $M(C)$-robot node.
  - (Case 2-2) If $|R_1| > 1$, each robot in R_1 moves toward its closest $M(C)$-robot node with probability $1/(2|R_1|)$.

Note that, in some configurations, a robot can move to both directions. In this case, the robot decides on the direction deterministically.

First, we briefly explain the behavior to reach a configuration in $\mathcal{C}_g$. In Case 1, there exists exactly one $M(C)$-robot node and other robots move to the $M(C)$-robot node. Consequently, the system reaches a configuration in $\mathcal{C}_g$. In Case 2, there exists more than one $M(C)$-robot node. In Case 2-1, R_1 contains one robot and so only one (or this) robot moves toward its closest $M(C)$-robot node. Since the robot gets closer to the $M(C)$-robot node, the robot remains closest to an $M(C)$-robot node unless it joins the $M(C)$-robot node. This means that the robot continues to move toward the $M(C)$-robot node until it joins, and the configuration becomes one in Case 1 after it joins. In Case 2-2, exactly one robot moves toward an $M(C)$-robot node with at least constant probability. Then, the configuration becomes one in Case 1 (when the robot moves to an $M(C)$-robot node) or Case 2-1 (when the robot moves to a free node). Therefore, the system reaches a configuration in $\mathcal{C}_g$ eventually.

In the following, we prove the complexity of the algorithm. The important behavior of the algorithm is that, when d robots can make probabilistic moves, each robot decides to move with probability $1/(2d)$. The aim of this procedure is to make exactly one robot move at the same time. To prevent such a behavior, the adversarial scheduler may activate these d robots simultaneously. However, the following lemma says exactly one robot moves with probability at least 1/4 even in this case. Note that, if the scheduler activates them separately, the probability that exactly one robot moves becomes higher.

Lemma 1. *Consider the configuration where d robots move with probability* $1/(2d)$ *and other robots cannot move. Then, the probability that the configuration changes in one round and exactly one robot moves in the configuration change is at least* $1/4$.

Proof. The probability is at least the probability that exactly one robot moves when all the d robots are activated at the same time. This probability is

$$\binom{d}{1} \cdot \frac{1}{2d} \cdot \left(1 - \frac{1}{2d}\right)^{d-1} \geq d \cdot \frac{1}{2d} \cdot \left(1 - \frac{d-1}{2d}\right) > \frac{1}{4}$$

since $(1 - (1/x))^y \geq 1 - (y/x)$ holds for any positive integer x and non-negative integer y (Bernoulli's inequality). □

In the following, we show that the system achieves gathering, that is, the system reaches a configuration in $\mathcal{C}_g$ from any configuration in each case.

Lemma 2. *From any configuration in Case 1, the system reaches a configuration in* $\mathcal{C}_g$ *in* $O(n \log k)$ *rounds and* $O(kn)$ *moves.*

Proof. Consider configuration C in Case 1. Then, there exists exactly one $M(C)$-robot node. We denote it by v_t. Since v_t continues to keep the biggest number of robots after that, all the other robots clearly gather to v_t. Since each robot moves at most n times, it is clear that the number of total moves is $O(kn)$.

In the following, we consider the number of rounds. The key property we will prove is that the number of robots on v_t is doubled or becomes k in $O(n)$ rounds. This property clearly implies that the number of rounds is $O(n \log k)$.

We show the key property. For simplicity, we assume all the robots are in the one side of the ring (The other cases can be proved similarly). That is, we assume each robot occupies a node in $\{v_t, v_{t+1}, \dots, v_{t+\lceil n/2 \rceil}\}$. Let m be the index such that $\sum_{i=1}^{m-1} N_C(v_{t+i}) < M(C)$ and $\sum_{i=1}^{m} N_C(v_{t+i}) \geq M(C)$. Note that, since robots on v_{t+m} cannot move before the number of robots on $v_{t+1}, \dots, v_{t+m}$ becomes less than the number of robots on v_t, robots on $v_{t+1}, \dots, v_{t+m-1}$ can continue to move until they reach v_t. Consequently all robots on $v_{t+1}, \dots, v_{t+m-1}$ in C reach v_t in $O(n)$ rounds. Then, robots on v_{t+m} can move because the number of robots on $v_{t+1}, \dots, v_{t+m}$ becomes less than the number of robots on v_t. Thus all the robots on v_m in C reach v_t in $O(n)$ rounds. Then, since $\sum_{i=1}^{m} N_C(v_{t+i}) \geq M(C)$ holds initially, the number of robots on v_t is doubled or becomes k in $O(n)$ rounds. □

Lemma 3. *From any configuration in Case 2-1, the system reaches a configuration in $\mathcal{C}_g$ in $O(n \log k)$ rounds and $O(kn)$ moves.*

Proof. At configuration C in Case 2-1, there exists exactly one robot r that can move. Then r is closest to an $M(C)$-robot node, and r moves toward it. Since the distance to the $M(C)$-robot node becomes smaller after r moves, other robots cannot move until r joins the $M(C)$-robot node. Consequently the system reaches a configuration in Case 1 in $O(n)$ rounds and $O(n)$ moves. Therefore, from Lemma 2, the system reaches a configuration in $\mathcal{C}_g$ in $O(n \log k)$ rounds and $O(kn)$ moves. □

Lemma 4. *From any configuration in Case 2-2, the system reaches a configuration in $\mathcal{C}_g$ in $O(n \log k)$ rounds and $O(kn)$ moves with probability at least $1/4$.*

Proof. At configuration C in Case 2-2, exactly one robot r moves with probability at least $1/4$ from Lemma 1. If r resides in a robot node at C and joins its neighboring $M(C)$-robot node, exactly one $(M(C)+1)$-robot node is created after r's movement. That is, the configuration becomes one in Case 1. Otherwise, the robot moves to a free node and gets closer to an $M(C)$-robot node. Then, the configuration becomes one in Case 2-1. From Lemmas 2 and 3, the lemma clearly holds. □

From Lemmas 2 to 4, the system reaches a configuration in $\mathcal{C}_g$ in $O(n \log k)$ rounds and $O(kn)$ moves with at least constant probability from any initial configuration. If the system does not reach the configuration in $O(n \log k)$ rounds and $O(kn)$ moves, robots can continue the algorithm because the configuration is in either Case 1, Case 2-1, or Case 2-2. Consequently, the system reaches a configuration in $\mathcal{C}_g$ in subsequent $O(n \log k)$ rounds and $O(kn)$ moves with at least constant probability. This implies that the system can reach a configuration in $\mathcal{C}_g$ by repeating the behavior constant expected number of times. Therefore, we have the following theorem.

Theorem 3. *The proposed algorithm solves the gathering problem in $O(n \log k)$ expected rounds and $O(kn)$ expected moves from any initial configuration.*

5. An algorithm for the orientation problem

The goal of the orientation problem in this paper is to make a configuration such that 1) it includes exactly one tower node and one 1-robot block of size ℓ, where ℓ ($\ell \leq n-1$) is a given input value, and 2) the tower node and the 1-robot block are neighboring (see Fig. 2). The problem requires that $k \geq \ell + 2$, that is, there should exist enough number of robots to create one tower node and one 1-robot block of size ℓ.

Our orientation algorithm achieves such configuration from the gathering configuration. That is, we first execute the gathering algorithm and then execute the orientation algorithm. Note that, as we describe later, some configurations for the orientation algorithm are the same as those for the gathering algorithm. Since robots are oblivious, it seems to be impossible for robots to recognize which algorithm is executed in the configurations. However, by executing the orientation algorithm preferentially, robots can execute the orientation algorithm after the gathering algorithm. This means that the gathering algorithm is switched to the orientation algorithm as soon as the configuration becomes one included in the orientation algorithm.

The orientation algorithm is divided into two phases. In the first phase, the algorithm achieves the target configuration for $\ell = 1$ or $\ell = 2$, *i.e.*, it extracts one or two robots from the tower node. The third phase achieves the target configuration for any ℓ.

5.1. The first phase

Let $\mathcal{C}_{f1}$ be a set of configurations such that there exist one tower node and one 1-robot node neighboring to the tower node, and let $\mathcal{C}_{f2}$ be a set of configurations such that there exist one tower node and one 1-robot block of size 2 neighboring

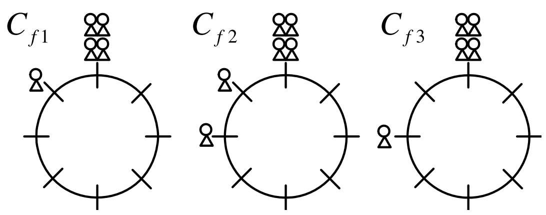

Fig. 4. The first phase of the orientation algorithm.

to the tower node (see Fig. 4). The goal of the first phase is to reach a configuration in $\mathcal{C}_{f1}$ (if $\ell = 1$) or $\mathcal{C}_{f2}$ (if $\ell \geq 2$). If either $\ell = 1$ or $\ell = 2$ holds, the orientation algorithm finishes in the first phase.

First, we show the algorithm under the assumption of $n \geq 4$ and $\ell \geq 2$ for simplicity. This assumption will be removed later.

The algorithm is based on the gathering algorithm, and the behavior of robots is simple. If the configuration is in $\mathcal{C}_g$, each robot moves to its neighboring node with probability $1/(2k)$. This implies that exactly one robot moves with probability at least $1/4$ from Lemma 1. In this case, the configuration becomes one in $\mathcal{C}_{f1}$. If the configuration is in $\mathcal{C}_{f1}$, the robot on the 1-robot node moves to its neighboring free node. After the move, the configuration becomes a configuration in $\mathcal{C}_{f3}$, where $\mathcal{C}_{f3}$ is a set of configurations such that 1) there exist exactly one tower node and exactly one 1-robot node and 2) there exists one free node between the tower node and the 1-robot node. Next, each robot on the tower node moves toward the 1-robot node with probability $1/(2(k-1))$. Exactly one robot moves with probability at least $1/4$, and then the configuration becomes one in $\mathcal{C}_{f2}$. Since robots move probabilistically when the configuration is in $\mathcal{C}_{f1}$ or $\mathcal{C}_{f3}$, the configuration may become one not in the above configuration sets. In this case, the system goes back to a configuration in $\mathcal{C}_g$ by executing the gathering algorithm.

In summary, the algorithm of the first phase is as follows:

- **Case 1:** If the configuration is in $\mathcal{C}_g$, each robot moves to its neighboring node with probability $1/(2k)$.
- **Case 2:** If the configuration is in $\mathcal{C}_{f1}$, the robot on the 1-robot node moves to its neighboring free node.
- **Case 3:** If the configuration is in $\mathcal{C}_{f3}$, each robot on the tower node moves toward the 1-robot node with probability $1/(2(k-1))$.
- **Goal:** If the configuration is in $\mathcal{C}_{f2}$, no robot moves.
- **Other case:** Otherwise, each robot executes the gathering algorithm.

Clearly, from a configuration in $\mathcal{C}_g$, the system reaches a configuration in $\mathcal{C}_{f2}$ if exactly one robot moves in Cases 1 and 3. This happens with probability at least $1/16$ from Lemma 1. If multiple robots move in Cases 1 and 3, the system reaches a configuration in $\mathcal{C}_g$ again in $O(n \log k)$ expected rounds and $O(kn)$ expected moves from Theorem 3. Therefore, from any initial configuration, the system reaches a configuration in $\mathcal{C}_{f2}$ in $O(n \log k)$ expected rounds and $O(kn)$ expected moves.

Lastly, we remove the assumption of $n \geq 4$ and $\ell \geq 2$. Consider case $n \geq 2$ and $\ell = 1$. In this case, the algorithm terminates if the configuration becomes one in $\mathcal{C}_{f1}$. If the configuration is neither in $\mathcal{C}_g$ nor in $\mathcal{C}_{f1}$, each robot executes the gathering algorithm. Next, consider case $n = 3$ and $\ell = 2$. In this case, if the configuration is in $\mathcal{C}_{f1}$, each robot on the tower node moves toward the free node with probability $1/(2(k-1))$. By this behavior exactly one robot moves with probability at least $1/4$, and then the configuration becomes one in $\mathcal{C}_{f2}$. If the configuration is neither in $\mathcal{C}_g$, $\mathcal{C}_{f1}$, nor $\mathcal{C}_{f2}$, each robot executes the gathering algorithm. For both cases, the system reaches a configuration in $\mathcal{C}_{f1}$ (if $\ell = 1$) or in $\mathcal{C}_{f2}$ (if $\ell \geq 2$) after executing the gathering algorithm constant expected number of times.

In summary, we have the following lemma.

Lemma 5. *From any initial configuration, the system reaches a configuration in $\mathcal{C}_{f1}$ (if $\ell = 1$) or in $\mathcal{C}_{f2}$ (if $\ell \geq 2$) in $O(n \log k)$ expected rounds and $O(kn)$ expected moves.*

5.2. The second phase

The goal of the second phase is the same as that of the orientation problem, that is, to construct a configuration such that 1) it includes only one tower node and one 1-robot block of size ℓ, and 2) the tower node and the 1-robot block are neighboring. Since the orientation algorithm finishes in the first phase if $\ell < 3$, we assume $\ell \geq 3$ in the following. In addition, for simplicity we assume $\ell \leq n - 3$. This assumption will be removed later.

First, we list all the possible configurations in this phase. We denote these configurations by regular configurations. Before we define regular configurations, we define some sets of configurations $\mathcal{C}_{r1}^m$, $\mathcal{C}_{r2}^m$, $\mathcal{C}_{r3}^m$, and $\mathcal{C}_{r4}^m$ for each $m \geq 2$ (see Fig. 5).

- Configuration C is in $\mathcal{C}_{r1}^m$ iff C satisfies the following conditions:
  - There exist one 1-robot block of size m, one tower node, and no other robot nodes,

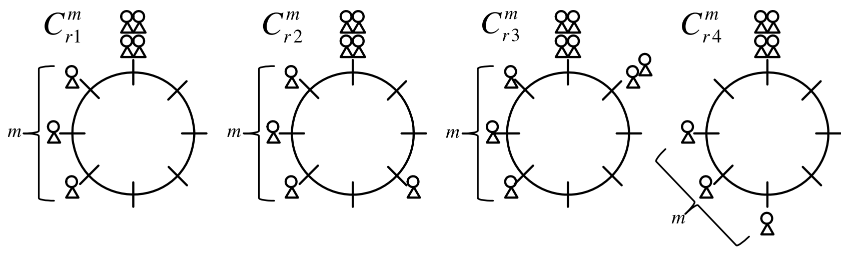

Fig. 5. The second phase of the orientation algorithm.

- and the 1-robot block and the tower node are neighbors.

- Configuration C is in $\mathcal{C}^m_{r2}$ iff C satisfies the following conditions:
  - There exist one 1-robot block of size m, one tower node, one isolated 1-robot node, and no other robot nodes,
  - and the 1-robot block and the tower node are neighbors.
- Configuration C is in $\mathcal{C}^m_{r3}$ iff C satisfies the following conditions:
  - There exist one 1-robot block of size m, two tower nodes, and no other robot nodes,
  - the 1-robot block and a tower node are neighbors,
  - and the two tower nodes are neighbors.
- Configuration C is in $\mathcal{C}^m_{r4}$ iff C satisfies the following conditions:
  - There exist one 1-robot block of size m, one tower node, and no other robot nodes,
  - and there exists exactly one free node between the 1-robot block and the tower node.

We define $\mathcal{C}^m_r = \mathcal{C}^m_{r1} \cup \mathcal{C}^m_{r2} \cup \mathcal{C}^m_{r3} \cup \mathcal{C}^m_{r4}$ and $\mathcal{C}_r = \bigcup_{2\le m\le \ell-1} \mathcal{C}^m_r$. We define configuration C is regular iff $C \in \mathcal{C}_r$ holds. By definition, the goal of the orientation problem is to reach a configuration in $\mathcal{C}_o = \mathcal{C}^\ell_{r1}$. Note that, since we assume $\ell \le n-3$, each regular configuration is not symmetric. Let us confirm this fact. For configurations in $\mathcal{C}^m_{r1}$, $\mathcal{C}^m_{r2}$, or $\mathcal{C}^m_{r3}$, since there exists only one tower node, the axis of symmetry can lie on the tower node. For configurations in $\mathcal{C}^m_{r3}$, since there exist only two tower nodes, the axis of symmetry can lie between the two tower nodes. However, for both cases, the distances from towers to their neighboring robot nodes are different between both directions because, from $m \le \ell - 1 \le n-4$, there exist at least three nodes except for tower nodes and a 1-robot block of size m.

The behavior of each robot for the second phase is given as follows.

- **Case 1:** If $C \in \mathcal{C}^m_{r1}$, each robot on the tower node moves to its neighboring free node with probability $1/(2d)$, where $d = k - m$ is the number of robots on the tower node.
- **Case 2:** If $C \in \mathcal{C}^m_{r2}$, a robot on an isolated 1-robot node moves toward the 1-robot block.
- **Case 3:** If $C \in \mathcal{C}^m_{r3}$, each robot in the tower node that is not a neighbor of the 1-robot block moves to the other tower node.
- **Case 4:** If $C \in \mathcal{C}^m_{r4}$, each robot in the tower node moves to the free node between the 1-robot block and the tower node.
- **Goal:** If $C \in \mathcal{C}_o$, no robot moves.
- **Other case:** If $C \notin \mathcal{C}_r \cup \mathcal{C}_o$, each robot executes the first phase of the orientation algorithm.

Clearly, after some robots move at regular configurations, the configuration is still regular or becomes one in $\mathcal{C}_o$. In the following, we prove the correctness of the algorithm.

Lemma 6. *From any configuration C such that $C \in \mathcal{C}^m_r$ and $2 \le m \le \ell - 1$, the system reaches a configuration in $\mathcal{C}^{m+1}_{r1}$ in $O(n)$ expected rounds and $O(k+n)$ expected moves.*

Proof. Let us consider the behavior at $C \in \mathcal{C}^m_{r1}$. From Lemma 1, exactly one robot on the tower node moves to the free neighbor with probability at least 1/4. Then, the system reaches a configuration in $C \in \mathcal{C}^m_{r2}$. After that, the system reaches a configuration in $\mathcal{C}^{m+1}_{r1}$ in $O(n)$ rounds and $O(n)$ moves. If all the robots except one robot moves at $C \in \mathcal{C}^m_{r1}$, the system reaches $C \in \mathcal{C}^{m+1}_{r1}$ in $O(1)$ rounds and $O(k)$ moves. If multiple or all robots move at $C \in \mathcal{C}^m_{r1}$, the system reaches $C \in \mathcal{C}^m_{r3}$ or $C \in \mathcal{C}^m_{r4}$ in $O(1)$ rounds and $O(k)$ moves respectively.

If the configuration is in $\mathcal{C}^m_{r3}$, the system reaches $\mathcal{C}^m_{r1}$ or $\mathcal{C}^m_{r2}$ in $O(1)$ rounds and $O(k)$ moves. If the configuration is in $\mathcal{C}^m_{r4}$, the system reaches a configuration in $\mathcal{C}^m_{r1}$, $\mathcal{C}^m_{r2}$, or $\mathcal{C}^{m+1}_{r1}$ in $O(1)$ rounds and $O(k)$ moves.

In all the cases, the system reaches $\mathcal{C}^{m+1}_{r1}$ in $O(n)$ rounds and $O(k+n)$ moves, or goes back to $\mathcal{C}^m_{r1}$ in $O(1)$ rounds and $O(k+n)$ moves. Consequently, by repeating the probabilistic move at most constant expected number of times, the system reaches a configuration in $\mathcal{C}^{m+1}_{r1}$. Therefore, the lemma holds. □

From Lemma 6, if the number of nodes in the 1-robot block is less than ℓ, it is increased in $O(n)$ expected rounds and $O(k+n)$ expected moves. Consequently, the following lemma holds.

Lemma 7. *From any configuration in $\mathcal{C}_r$, the system reaches a configuration in $\mathcal{C}_o$ in $O(\ell n)$ expected rounds and $O(\ell(k+n))$ expected moves.*

Since $\mathcal{C}_{f2} = \mathcal{C}_{r1}^2 \subset \mathcal{C}_r$, the proposed algorithm solves the orientation problem for input ℓ such that $\ell \le n-3$ from Lemmas 5 and 7. From $\ell \le \min\{k, n\}$, we have the following lemma.

Lemma 8. *The proposed algorithm solves the orientation problem for input ℓ in $O((\log k + \ell)n)$ expected rounds and $O(kn)$ moves from any initial configuration if both $k \ge \ell+2$ and $\ell \le n-3$ hold.*

Finally we remove the assumption $\ell \le n-3$. Consider case $\ell = n-2$. From the above algorithm, we can construct a configuration in $\mathcal{C}_{r1}^{\ell-1}$. From this configuration, each robot on the tower node moves to its neighboring free node with probability $1/(2d)$, where d is the number of robots on the tower node. By this behavior, exactly one robot moves with probability at least $1/4$ and then the robot can join the 1-robot block. Even if multiple robots move, the system can go back to a configuration in $\mathcal{C}_{r1}^{\ell-1}$ again in $O((\log k + \ell)n)$ expected rounds and $O(kn)$ expected moves. Therefore, by repeating the above behavior constant expected number of times, the system reaches the target configuration for $\ell = n-2$. For the case of $\ell = n-1$, we can construct the algorithm similarly. Therefore, we have the following theorem.

Theorem 4. *The proposed algorithm solves the orientation problem for input ℓ in $O((\log k + \ell)n)$ expected rounds and $O(kn)$ moves from any initial configuration if $k \ge \ell+2$ holds.*

6. Application: an algorithm for the set formation problem

In this section, we show an algorithm for the set formation problem as an application of our gathering and orientation algorithm. The goal of the set formation problem is to form *SET* in Definition 3. If $|SET| = 1$ holds, the gathering algorithm solves the set formation problem, and if $|SET| = n$ holds, the orientation algorithm for $\ell = n-1$ solves the set formation problem. For this reason, we assume $2 \le |SET| \le n-1$ in this section.

Similarly to the previous section, we first execute the orientation algorithm for $\ell = |SET| - 1$ and then execute an algorithm for the set formation problem. That is, we assume that initial configuration C is in $\mathcal{C}_o$. The set formation algorithm is executed preferentially, and thus if the system reaches a configuration in the set formation algorithm during the execution of the gathering or orientation algorithm, the system transits to the phase for the set formation immediately.

Since a configuration in $\mathcal{C}_o$ is not symmetric, all the robots can recognize the ring network in the same direction. Without loss of generality, we assume that v_0 is the tower node and $v_1, v_2, \dots, v_\ell$ form the 1-robot block. We define the direction from v_i to v_{i+1} (resp., from v_{i+1} to v_i) as the forward (resp., backward) direction.

To realize the set formation algorithm, each robot has to decide on the positions that form the target set. In our algorithm, each robot decides on the positions based on a tower node v_0. That is, we assume each robot knows a function $SET(v_0)$ that gives positions that form the target set based on a tower node v_0. For ease of explanation, we assume that $SET(v_0)$ satisfies the following conditions.

1) $v_0 \in SET(v_0)$.
2) $v_{n-1} \notin SET(v_0)$.
3) Let $i_{min} = \min\{i \mid v_i \in SET(v_0) \setminus \{v_0\}\}$ and $i_{max} = \max\{i \mid v_i \in SET(v_0) \setminus \{v_0\}\}$. In other words, $v_{i_{min}}$ and $v_{i_{max}}$ are target nodes closest to v_0 in the forward and backward directions of v_0, respectively. Then, $i_{min} \le n - i_{max}$ holds. That is, $v_{i_{min}}$ is not farther from v_0 than $v_{i_{max}}$.

Note that these conditions do not compromise generality because only a relative set *SET* is given as an input of the set formation problem. So, we can select arbitrarily a target node set in $\mathcal{F}(SET)$. For example, in the case of $n = 8$ and $SET = \{0, 1, 3, 6, 7\}$, $SET(v_0) = \{v_0, v_1, v_2, v_3, v_5\}$ satisfies the above conditions.

The algorithm is very simple: Each robot on the 1-robot block moves to a node in $SET(v_0)$ one by one (see Fig. 6). In more details, let v_a be a free node with maximum index a such that v_a is in $SET(v_0)$. Let v_b be a 1-robot node with maximum index b such that $b < a$. That is, v_b is the robot node closest to v_a in the backward direction of v_a. Then, a robot on v_b moves toward v_a.

Since both $v_0 \in SET(v_0)$ and $v_{n-1} \notin SET(v_0)$ hold, the configuration never becomes symmetric during the execution of the algorithm before a robot on v_1 moves. After the robot on v_1 moves, the configuration never becomes symmetric during the execution because of the third condition of $SET(v_0)$. Note that, even if a configuration that forms $SET(v_0)$ is symmetric, the last configuration becomes symmetric but configurations during the execution are not symmetric. Since at least one robot moves in each round, each robot moves at most n times. Therefore, since the number of moving robots is at most $\ell = |SET| - 1$, the following lemma holds.

Lemma 9. *From any configuration in $\mathcal{C}_o$, the system reaches a configuration that forms SET. This takes at most $O(|SET|n)$ rounds and $O(|SET|n)$ moves.*

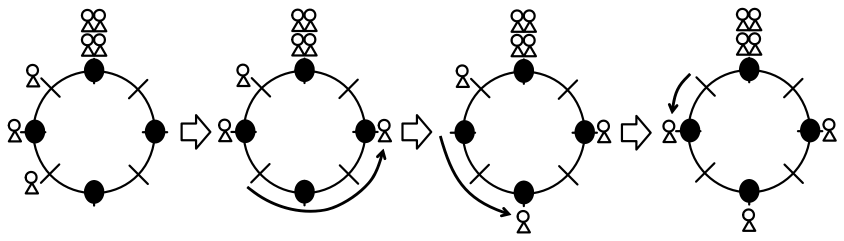
Fig. 6. The set formation algorithm. Black circles denote nodes in $SET(v_0)$.

By combining the previous algorithms, we can obtain a self-stabilizing algorithm for the set formation problem. From Theorem 4 and Lemma 9, we have the following theorem.

Theorem 5. *The proposed algorithm solves the set formation problem for set SET in* $O((\log k + |SET|)n)$ *expected rounds and* $O(kn)$ *expected moves from any initial configuration if* $k \geq |SET| + 1$.

7. Conclusion

In this paper, we investigated self-stabilizing algorithms for robots in ring networks. We have proved that no probabilistic self-stabilizing gathering algorithm exists in the asynchronous (ASYNC) model or if robots are only endowed with global-weak and local-strong multiplicity detection. This means that, contrary to popular belief, robot algorithms are *not* "almost" self-stabilizing, as removing the assumption on the initial global configuration of robot positions implies making very strong system hypotheses to retain problem solvability. On the other hand, we have presented a probabilistic self-stabilizing algorithm for the gathering and orientation problem in the semi-synchronous (SSYNC) model with global-strong multiplicity detection. Since the gathering and orientation algorithms solve the essential difficulties that arise in robot networks, many algorithms can be derived by the proposed algorithms. As an example application of our scheme, we provided a self-stabilizing algorithm for the set formation problem. This means that any static set formation can be realized in a self-stabilizing manner in our model.

An interesting direction for future research is to characterize the problems that are actually solvable in a self-stabilizing setting using weaker hypotheses (*e.g.* ASYNC model, no global-strong multiplicity detection, etc.). Another interesting direction is to consider weaker forms of schedulers. In this paper, we assume a powerful scheduler that adapts to robots' decisions, and obtain worst-case analysis. It is worth investigating the solvability of the problems we considered under an oblivious scheduler, which is a scheduler whose decisions are unrelated to robots algorithms. Such a scheduler may prove useful to assess solution efficiency in more realistic settings.

Acknowledgements

This work is supported in part by JSPS KAKENHI Grant Numbers 24500039, 24650012, 25104516, 26280022, 26330084. This work is supported in part by LINCS.

References

[1] I. Suzuki, M. Yamashita, Distributed anonymous mobile robots: formation of geometric patterns, SIAM J. Comput. 28 (4) (1999) 1347–1363.
[2] Y. Dieudonné, F. Petit, Self-stabilizing gathering with strong multiplicity detection, Theoret. Comput. Sci. 428 (2012) 47–57.
[3] J. Clement, X. Défago, M. Gradinariu, T. Izumi, S. Messika, The cost of probabilistic agreement in oblivious robot networks, Inform. Process. Lett. 110 (11) (2010) 431–438.
[4] T. Izumi, T. Izumi, S. Kamei, F. Ooshita, Feasibility of polynomial-time randomized gathering for oblivious mobile robots, IEEE Trans. Parallel Distrib. Syst. 24 (4) (2013) 716–723.
[5] P. Flocchini, G. Prencipe, N. Santoro, P. Widmayer, Gathering of asynchronous robots with limited visibility, Theoret. Comput. Sci. 337 (2005) 147–168.
[6] C. Agathangelou, C. Georgiou, M. Mavronicolas, A distributed algorithm for gathering many fat mobile robots in the plane, in: Proc. of PODC, 2013, pp. 250–259.
[7] A. Honorat, M. Potop-Butucaru, S. Tixeuil, Gathering fat mobile robots with slim omnidirectional cameras, Theoret. Comput. Sci. 557 (2014) 1–27.
[8] S. Das, P. Flocchini, G. Prencipe, N. Santoro, M. Yamashita, The power of lights: synchronizing asynchronous robots using visible bits, in: Proc. of ICDCS, 2012, pp. 506–515.
[9] G. D'Angelo, G.D. Stefano, R. Klasing, A. Navarra, Gathering of robots on anonymous grids without multiplicity detection, in: Proc. of SIROCCO, 2012, pp. 327–338.
[10] R. Klasing, A. Kosowski, A. Navarra, Taking advantage of symmetries: gathering of many asynchronous oblivious robots on a ring, Theoret. Comput. Sci. 511 (2010) 3235–3246.
[11] R. Klasing, E. Markou, A. Pelc, Gathering asynchronous oblivious mobile robots in a ring, Theoret. Comput. Sci. 390 (2008) 27–39.
[12] G. D'Angelo, G.D. Stefano, A. Navarra, Gathering on rings under the look-compute-move model, Distrib. Comput. 27 (2014) 255–285.
[13] T. Izumi, T. Izumi, S. Kamei, F. Ooshita, Time-optimal gathering algorithm of mobile robots with local weak multiplicity detection in rings, IEICE Trans. Fundam. Electron. Commun. Comput. Sci. E96-A (630) (2013) 1072–1080.
[14] S. Kamei, A. Lamani, F. Ooshita, S. Tixeuil, Asynchronous mobile robot gathering from symmetric configurations without global multiplicity detection, in: Proc. of SIROCCO, 2011, pp. 150–161.

[15] S. Kamei, A. Lamani, F. Ooshita, S. Tixeuil, Gathering an even number of robots in an odd ring without global multiplicity detection, in: Proc. of MFCS, 2012, pp. 542–553.
[16] S. Devismes, F. Petit, S. Tixeuil, Optimal probabilistic ring exploration by semi-synchronous oblivious robots, Theoret. Comput. Sci. 498 (2013) 10–27.
[17] P. Flocchini, D. Ilcinkas, A. Pelc, N. Santoro, Computing without communicating: ring exploration by asynchronous oblivious robots, Algorithmica 65 (2013) 562–583.
[18] A. Lamani, M.G. Potop-Butucaru, S. Tixeuil, Optimal deterministic ring exploration with oblivious asynchronous robots, in: Proc. of SIROCCO, 2010, pp. 183–196.
[19] Y. Elor, A.M. Bruckstein, Uniform multi-agent deployment on a ring, Theoret. Comput. Sci. 412 (2011) 783–795.